The Journal of
Reliability, Maintainability & Supportability
in Systems Engineering

Spring 2013

RMS Reliability, Maintainability and Supportability Partnership

Table of Contents

Introduction

James Rodenkirch

Chris Peterson, the author of a testing article for our Winter 2012 RMSP Journal titled, "Experimentation at Team Corporation, using single and multiple-axis vibration: a comparison in time to failure," has set up a blog site at: http://chamber-queen.blogspot.com. Chris utilizes this blog site to offer up insight(s), via short, informational articles, on a variety of RMS-related topics; e.g., testing, reliability, MIL-STD-810, personal management as well as testing industry updates.

Chris has been associated with the business of testing since 1990 and her credentials are noteworthy. She is on the MIL-STD-810G editing committee, an immediate Past President of IEST (the The Institute of Environmental Sciences and Technology, a Board Member of the RMS Partnership, RAMS® committee, and IEEE's ASTR (Accelerated Stress Testing & Reliability) and an active participant in the ISO/IEC WG50. She teaches testing topics around the world and exhibits a strong passion for helping others to do their testing right—all of that with a goal of improving reliability and safety, while lowering costs - the "bottom line," to her, is to make all parties the winner—a "bottom line" we should always focus on!

I've gleaned a lot from Chris' blog site about testing and, being keenly aware of the vagaries and problems associated with the new Enterprise System(s) architecting and engineering approach, I've discovered some "nuggets" regarding what to expect, from an "interacting-with-others perspective," as we move from a centralized/stove-pipe organizational model to a more decentralized, open and matrix-like society that supports the "Enterprise work environment." Two of her deliberations caught my eye regarding all of that: the post of 12/31/2012, "What Measurements Do You Need to Do Your Best Testing," and her 2/11/2013 offering, "Another Lesson from the Boeing Battery Failure." In, What Measurements Do You Need to Do Your Best Testing, Chris promotes the idea, without actually stating same, that reductionist thinking is in decline while holistic approaches and Systems Thinking—progenitors of top-down and bottom-up approaches to problem solving—are on the rise. Additionally, Chris takes a moment out from reading all of the Monday morning quarterbacking re the Boeing battery failure to coalesce the myriad posts over what should have been done into a synopsis of the four types of "opines"—a) you shoulda came to me a long time ago; b) I've been telling you guys this for years; c) finger pointing (reminds me of my old Navy days—fix the blame before you fix the problem; and, finally, d) a sensible query regarding how any testing was accomplished. Chris' observations on the types of responses generated over a system/component failure are telling and germane because, as our RMS world shrinks and we find ourselves interacting with others across myriad and, perhaps, unfamiliar engineering disciplines—we will be exposed to varied opinions and approaches to problem solving. Our mantra should be…pencil in hand and at the ready?...focus on response "d"—it's the only one that searches out root causes and promotes making everyone winners!

What Measurements Do You Need to Do Your Best Testing

Measurements needed for testing are broken into two general categories—field measurements and measurements during test—the good part is that the same types of sensors can often be employed.

Each of those categories has several subcategories in common such as:

- Climatic (temperature, humidity, rain, ice, solar radiation, and more)
- Dynamic (shock, vibration, acceleration, etc.)
- Electrical (EMC, ESD, voltage fluctuation, etc.)

All of that doesn't look too difficult— it's just a list of things to check for, right? WRONG! Not only do the single environments need to be measured but the combinations of environments as well. If something fails it is rarely because of only one environmental stress. Temperature (1) + Humidity (1) does not necessarily make (2). If both are in the right range they could make conditions just right for fungal growth, if humidity is high it could change the plasticity of materials, if it is low there could be issues with ESD. As temperature changes it creates air movement which could also change the stress level—either making it more or less stressful. These things are much more likely to be seen in the field than in the lab where we can design a test that will REPLICATE the majority of environmental effects but will never DUPLICATE nature.

Here's the plain and simple truth so listen up. IF YOU DON'T KNOW THE END ENVIRONMENT YOU CAN NEVER EXPECT YOUR TESTING TO TEACH YOU WHAT YOU NEED TO KNOW.

Let's take a closer look at climatics. For the sake of discussion, your widget is a piece of electronics equipment that will be mounted on some type of electronics panel in a box that is to be outside. You already feel that you know your widget. Now what? Now it's time for questions:

When it is mounted, what could it be mounted next to?

Could that be high heat emitting and, if so, could the high temp have an adverse effect on your widget?

What kind of box will it go into? What will the material be? (metal, composite, plastic, etc.) Will the box be vented? Where will the box be mounted? (on the ground, on a pole, on a wall, inside a shed)

Will it have any protection? (Weather protected with climatic control, weather protected without climatic control, semi-weather protected or non-protected)

Will it typically be operational or nonoperational? (A redundant unit may typically be nonoperational but will be expected to work as soon as it is activated after a long period of rest—and it will still get environmental stresses.)

Is it for use in one particular climate or is it for worldwide use? (If for worldwide, or if there is a chance that people could use it outside the zone you expect, then you need to remember possible extremes beyond what you are expecting.)

Now let's get down to what to measure for climatics. You can't just say temperature, humidity, sun and rain—you need to look deeper.

Let's start with temperature and break it down. It takes a lot more than a weather report. It is important to know the outdoor temperature but it takes more than that and, remember, just because a general area may have a given temperature doesn't mean that everything in that area is that temperature. (Think about sun vs. shade; being shaded in the summer can be a protection but in the winter can mean much longer at very low temperatures.) Basically, weather is outside that box and we need to know the temperature inside it. There is a specification that I was reading that gave ballpark figures for temperatures inside boxes. However, the temperatures given were in the air away from all heat emitting items where it was considered stable and uniform. Guess what? That is NOT the temperature you need! You need to know where the part is that gives off the most heat and where your widget is in comparison (or if the heat emitted by your widget could lead to the destruction of something else). You also need to know whether everything else will always be turned on, whether the inside of the box will be lit (heat source), whether a "turn on" phase could be the highest heat time, etc.

Here is what you need to measure—your widget. Consider different layouts of the components, whether something hotter could be placed next to it, and what could happen if someone did not allow enough space for airflow around it. That means that you cannot just take the measurements but also need to figure in some margin when you write your test plan.

On to the humidity. Again, weather reports will give a general idea of what is going on outside of the box, but inside can be a completely different story. Let's look at the difference between a vented and unvented box as we think about humidity. An unvented box will hold the heat in which could seem like a good thing—it should bake off the humidity. That's what it might seem like, but there are other things to consider. When the panel was mounted (or maintained) it could have been on a high humidity day and that high moisture content could overcome the best of intentions. Most boxes are not airtight, and there is a "breathing" effect. (An unvented box holding heat in could also lead to overheating conditions.) A vented box, on the other hand, increases the breathing, the chance for condensation, possible rain spray coming in with high winds, etc. MEASURE.

You get the idea. Measure for different circumstances, especially worst case. Then use that information for test. Only then can you…test to be your best.

Another Lesson from the Boeing Battery Failure

I have worked on Boeing and Airbus related projects for a couple of decades now and, being a frequent flyer, take an active interest in aircraft issues. This battery issue has me very interested and I've had mixed emotions when reading responses to it. (The responses below are not from a single person but a mix of types of responses that I've read.) These can happen in your own management or test lab so I thought I'd highlight them as a learning lesson.

Response A: Done with frequent interspersed words in all caps and misspelling. This is the person who thinks that if only Boeing would have come to him, personally, that there would never have been an issue. He publicly offers to do a failure analysis and then proceeds to do it without having full information but only what was gleaned through news reports. (Let me tell you, I've seen the reports and there isn't much information there to glean which means it is mostly speculation.)

Response B: From someone who believes in only accelerated testing. He comes across like, "Sigh, I've been telling you this for years and people still aren't listening. You have to test to failure, fix and test some more. Clearly this wasn't tested for. When will people listen?"

Response C: Exaggerated finger pointing at every single major failure while leaving out the fact that there are fewer major failures in aircraft than in automobiles, portable electronics is, virtually, anything else you can think of.

Let's stop for a minute and see why this is so. If an aircraft has a catastrophic failure it could easily mean hundreds of lives. On top of that, it can kill sales. No

aircraft manufacturer wants to deal with that. There are also agencies, like FAA, that will have rules to be followed. Believe me, no aircraft manufacturer will ever use something without testing it; there is a lot invested and far too much to lose.

Response D: This one made the most sense to me. A man wrote in to the editor of Aviation Week to ask about it. He wasn't pretending to have the answers, but wondered if the testing that was done included going through various altitudes while testing.

This man has the key. Not only does testing need to be done, but testing that will simulate the environments that the unit under test will be likely to see in usage. This means taking measurements so that the actual conditions are known.

Doing accelerated testing without using altitude has limited value for anything going aboard aircraft. Temperature, humidity, vibration, and power cycling can be replicated but changing air pressure can make a huge difference. That includes things like leakage of gases or fluids from gasket-sealed enclosures; deformation, rupture or explosion of sealed containers; change in physical and chemical properties of low-density materials; overheating of materiel due to reduced heat transfer; evaporation of lubricants; and failure of hermetic seals. While some of these may show up under other environments not all of them will.

Kneejerk reactions are a dime a dozen. It's easy to come across as if you can diagnose a problem without completely understanding it; it's even easier to point blame without bothering to find out the whole story. You've got the advantage of "hindsight vision" and can sit back for hours saying, "this, this and that should have been done," but have no clue whether it had been done, or not.

The best way to use this information is to avoid your own catastrophic "battery" issues. Measure the end environments, know how they interact and the effects of synergism, and test to that - not with a one-size-fits-all test but with a test tailored specifically to recreate the same effects as closely as possible. If someone believes that you can just overstress everything and that way you don't need measurements, and by the way you don't even need to use the same stresses, be wary. If someone thinks they can singlehandedly come up with the solutions let them have their fun, but realize that often the wisest solutions come from teams. They definitely need information to be built upon. Value the person that asks, "what if?" Then you can truly…test to be your best.

I hope those two "intro blogs" from Chris' blog site entice you to go to her site, on occasion, to see what's of importance, for the day, to a fellow RMS practitioner. Note: I asked Chris, in advance, if it was ok to purloin two of her blog posts and she was comfortable with me doing that!

Before getting in to my "Intro" regarding the four articles I have some snippets of 'new info' I discovered, related to comments from the Director of DARPA, Arati Prabhakar, which I'd like to share with you. In the latest NDIA Defense Watch weekly e-mail I received in early May, Sandra Erwin, the Editor of the NDIA magazine, focused on several elements of Ms. Prabhakar's new document titled, "Driving Technological Surprise: DARPA's Mission in a Changing World." You can find that document at this URL: *http://www.nationaldefensemagazine.org/blog/Documents/DARPA%20Framework_Embargoed%20until%201500%20ET%204.24.2013.pdf*

Ms. Erwin zeroed in on several points of interest that Ms. Prabhakar talks to in her document:

- DARPA is not going to have the luxury of time and money to pursue its trademark silver-bullet technologies and, instead, will emphasize "layered," "adaptable" and "multifunctional" concepts.
- When we consider future engagements, we can more readily imagine a host of diverse environments and adversaries. In an uncertain world, adaptability is critical. We won't always know exactly what we will need for tomorrow's battle." DARPA will focus on "systems that can be readily upgraded and can adapt in real time to changing surroundings and conditions."
- Budget cuts also have to be factored into DARPA's future and, Ms. Prabhakar is quick to add that "we may be at the beginning of a fundamental shift in how our society allocates resources to the business of national security. And I'm not talking today about the immediate issues around sequestration." "Enduring fiscal pressures," she added, "could shape a different future over the coming years and decades."

It was the budget/costs piece that Ms. Erwin zeroed in on because Ms. Prabhakar states that DARPA is ready to look at how DoD can turn the rising costs of our weapon systems, as our enemies counter them with low-cost technological systems, in to a positive by asking this question: how can we impose more cost on our adversaries and less on ourselves, thereby increasing our deterrent?

In other words, DARPA wants to help "invert the cost equation," as stated by Ms. Prabhakar. She goes on to say that DARPA will be seeking "ways to use innovation not just to nibble at the cost of systems, but really to fundamentally change the cost equation and to inflict much more costs on our adversaries to respond to the solutions that we come up with." The notion of increasing our measure of deterrence by turning the rising costs of systems into a counter to whatever our enemies are doing to

counter our systems smacks of confrontational interoperability, the interoperability of friendly (blue) and adversary (red) forces and the subject of several Measure of Merit discussions I enjoin my students to participate in during my Complex Systems Architecting course at Southern Methodist University. We all understand "good interoperability," referred to as cooperative interoperability but…confrontational?? Perhaps, not so much. In a confrontational operational scenario or process, one set of systems tries to achieve advantage or effectiveness over the remaining systems; many military processes, such as time critical targeting, psychological operations, or defensive counter air are inherently confrontational. Ms. Prabhakar uses cyber warfare as a case in point. She said. "Cyber is an extraordinary example of the importance of changing the cost equation. In this new domain, threats range from self-trained individuals, who can sometimes go up against costly, sophisticated systems, to the concerted efforts of nation states." She continues with this; "We can readily imagine a future in which cyber warfare is fully integrated with kinetic warfare—and there is no doubt that our potential adversaries can see this future as well." From an RMS perspective, I believe that if you haven't been that involved in discussions surrounding those two interoperability types, perhaps now is the time to initiate a little research into them. One can expect RMS considerations for system functions that support confrontational interoperability capabilities will have a different flavor(s), terminology and measurements than MTBF or MTTR or…pick any current RMS measurement applied to cooperative interoperability.

OK—on to my "Intro" regarding the article submissions for our Spring 2013 Journal.

John O. Clark authored an article on System of Systems Engineering (SoSE) and Family of Systems Engineering (FoSE), back in 2009 that appeared in an IEEE International Conference on System of Systems Engineering. We received approval from IEEE, courtesy of John's "support," to re-publish this article and hope you enjoy its focus—addressing two of the least well-understood SE disciplines, SoSE and FoSE! John provides us with his knowledge of the SE standards, the V-Model, and particularly the 3-dimensional Dual-V Model, as he goes about aiding our understanding of the relationship(s) between SE, SoSE, and FoSE. Additionally, I was fortunate to "bump in to" an article by James N. Martin on the Seven Samurai of Systems Engineering. Mr. Martin takes an Enterprise-wide approach to understanding the seven different systems that must be acknowledged and understood by those who purport to do systems engineering. Since all of us "work" within the world of Systems Engineering, I thought this paper was germane and salient so I approached Mr. Martin and asked for his "ok" to re-publish his article; this article deserves a "read" by us all. He gave me the head nod so here it is for all to enjoy. In short, these two articles comprise our semi-annual foray into the world(s) of Systems Engineering, from an Enterprise perspective, courtesy of Mr. Martin and Mr. Clark.

The other articles, "Legacy Sustainment: Slashing the costs of COTS Obsolescence Management" by Kaye Porter and "Sustaining COTS: Is the problem "a square peg in a round hole" or an attempt to put the right peg in the wrong hole?" by Anthony E. Trovato round out our "reading assignment" with more traditional foci on COTS which, it turns out, has a direct link(s) to FoSE and the Intervention System(s), part of the foci from Mr. Martin and Mr. Clark!

So, two diverse sets of articles for your reading pleasure—all four can be linked, easily, with each other if one puts on his or her "Enterprise Architecture" thinking cap. Good reading!

System of Systems Engineering and Family of Systems Engineering from a Standards, V-Model, and Dual-V Model Perspective

John O. Clark

Based on "System of Systems Engineering and Family of Systems Engineering from a Standards Perspective," by John O. Clark which appeared in the IEEE International Conference on System of Systems Engineering, 2008. SoSE '08.

Abstract

System of Systems Engineering (SoSE) and Family of Systems Engineering (FoSE) continue to be two of the least well-understood SE disciplines. Knowledge of the SE standards, the V-Model, and particularly the 3-dimensional Dual-V Model, significantly aid this understanding, including the relationship between SE, SoSE, and FoSE.

The goals of this paper are to: 1) define SoS, SoSE, and FoSE from an SE standards perspective; 2) describe the original V-Model and the Dual-V Model; 3) show how to apply these SE standards and V-Models to a system, to SoSs, and to FoSs; and 4) encourage and challenge the participants to understand, select, tailor, and apply these SE standards and V-Models to complex SoSs and FoSs. Individuals may have an understanding of portions of SE, SoSE, and FoSE based on other sources. The SE standards, V-Model, and Dual-V Model provide a more complete and common understanding.

Keywords: System of Systems (SoS), System of Systems Engineering (SoSE), Family of Systems (FoS), Family of Systems Engineering (FoSE), Complex Systems, Complex Systems Engineering, V-Model, Dual V-Model.

1 Introduction

The subject of SoSE versus SE is currently debated in the literature and at conferences. The question is asked: "Is engineering a system of systems really any different from engineering an ordinary system?" [1]. Some believe that SoSE is "different" from SE, the SE processes are inadequate or insufficient for SoSE, and additional processes are needed. Others, like me, believe the SE processes as documented in the SE standards: IEEE 1220, EIA/IS-632, EIA-632, ISO 15288 [2-8], and the guide: ISO TR 19760 [9], are a necessary and sufficient set of processes for SoSE, and no additional processes are needed.

The above SE standards and guide are referenced in this paper and used in the presentation that accompanies this paper. However, because they are copyrighted by the publishing organizations, material from them cannot be reproduced in this paper or in softcopies of the presentation. Refer to these SE standards and guide for this information.

In my opinion (based on reading, comparing, understanding, teaching, revising, tailoring, and applying the SE standards), there is only one classical SE process as shown in Figure 1 [5].

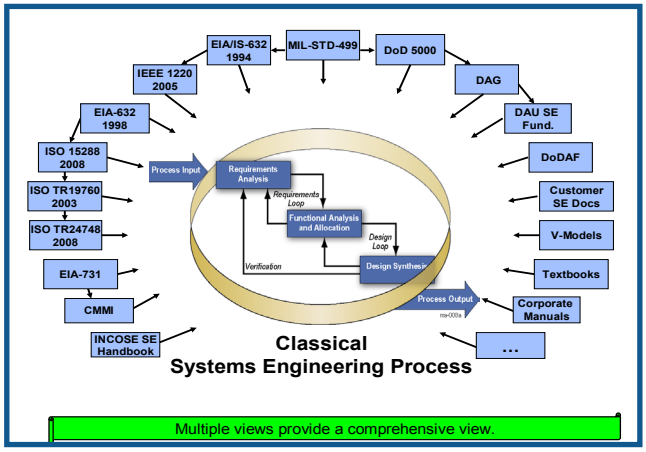


FIGURE 1 – SYSTEMS ENGINEERING VIEWS

Each SE standard presents a slightly different view of this one classical SE process. By understanding each SE standard, and looking at each standard's view, a systems engineer can get a comprehensive view of this one classical SE process and apply it to SoSE and FoSE. This principle also applies to the guides, manuals, handbooks, etc, shown in Figure 1.

Systems engineers may struggle with applying SE to FoSE. However, FoSE is simply SE applied to a FoS. By family, we mean a product-line or domain, wherein some assets are re-used un-modified; some assets are modified, used, and re-used later; and some assets are developed new, used, and re-used later. Product-lines are the result.

This paper addresses SoSE and FoSE from the SE standards, V-Model, and Dual V-Model perspective. In my opinion, this information is sorely needed to meet the challenges of complex SoSE and FoSE.

2 What is Different about SoSE and FoSE from SE?

In my opinion, SoSE and FoSE are an acquisition management problem, not a technical problem. The technical problem is solvable using the SE Standards and V-Models, but the acquisition management problem has not been solved. A few key management issues are:

- There is no god (no overall Program Manager) of a SoS or FoS
- Acquisitions are stovepipes (single systems, not SoS or FoS)
- Systems are directed to "integrate" with other systems, often after fielding

- Suppliers don't cooperate with each other in FoSE (they believe it's not in their best interest)
- Acquirers don't cooperate with each other for the same reason
- FoSE costs more up-front to develop for re-use (but saves much more later)

There are several key challenges to SoSE [10]. For example, SoS and FoS may consist of new, modified, or unmodified systems; and some systems may be evolving and their future unpredictable. To mitigate the risks inherent in these challenges, focus should be placed on developing and controlling the interfaces between system elements and external systems. Developing and controlling interfaces correctly is what integration and interoperability are about.

3 The Building Block

For a system or a SoS, the SE standards apply the Building Block concept. A system or SoS Building Block consists of Products, Processes, and People (some standards call these three items Elements) as shown in Figure 2.

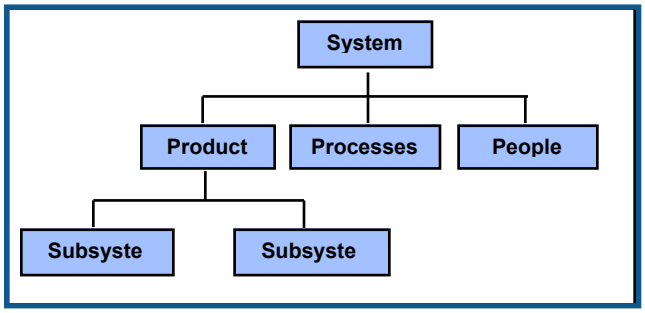


FIGURE 2 – SYSTEM BUILDING BLOCK

Next, the SE standards construct a system or a SoS using these Building Blocks as shown in Figure 3. A system or SoS can be decomposed from the top down, composed from the bottom up, or a combination of the two.

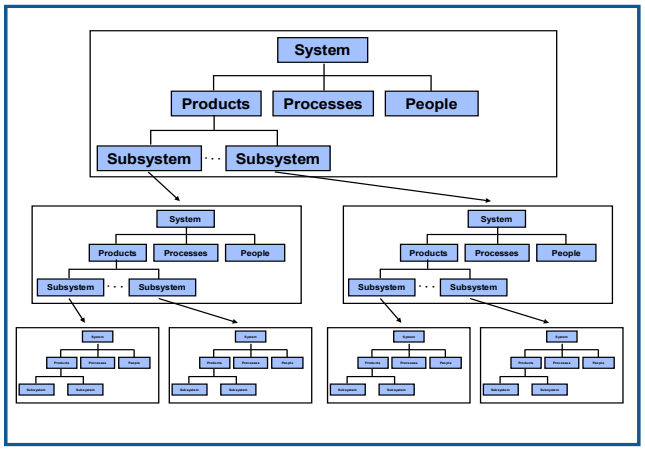


FIGURE 3 – SYSTEM OF SYSTEMS BUILDING BLOCKS

Each subsystem of the system or the SoS is treated as a system in its own right. For top-down SoSE, the Building Block structure continues on down the System Breakdown Structure (SBS) to the leaf-level that is needed to describe the SoS. For bottom-up SoSE, the opposite occurs.

Other structures are determined from the SBS such as the Work Breakdown Structure (WBS), the Specification Tree, and the Integrated Product Team (IPT) organization.

4 Simple Definitions of SoS and FoS

Following are simple definitions of SoS and FoS:

- SoS: The sum of the whole is greater than the sum of the individual parts:
  - The parts are integrated (i.e., have interfaces)
  - The parts may or may not be members of a common domain (such as a product line, for example: surface ship radars)
- FoS: The sum of the whole is equal to the sum of the individual parts:
  - The parts are not integrated
  - The parts are members of a common domain (such as a product line)

Integrating systems could result in the whole being less than the sum of the individual parts, but I assume that's not the case if they are integrated correctly!

5 The U.S. Department of Defense's Definitions of SoS and FoS

Per the DoD Defense Acquisition Guidebook (DAG) 2006 version [11], SoSE:

- Deals with planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into a SoS capability greater than the sum of the capabilities of the constituent parts.
- SoSs should be treated and managed as a system in their own right, and should therefore be subject to the same systems engineering processes and best practices as applied to individual systems.
- Differs from the engineering of a single system. The considerations should include the following factors or attributes:
  - Larger scope and greater complexity of integration efforts;
  - Collaborative and dynamic engineering;
  - Engineering under the condition of uncertainty;
  - Emphasis on design optimization;
  - Continuing architectural reconfiguration;
  - Simultaneous modeling and simulation of emergent system of systems behavior; and
  - Rigorous interface design and management.

Per the DAG 2006 version, a FoS:

- Is not considered to be a system per se.
- Does not create capability beyond the additive sum of the individual capabilities of its member systems.
- Basically a grouping of systems having some common characteristic(s). For example, each system in a FoS may belong to a domain or product lines (e.g., a family of missiles or aircraft).
- Lacks the synergy of a SoS.
- Does not acquire qualitatively new properties as a result of the grouping. In fact, the member systems may not be connected into a whole.

Per the DoD SE Guide to Systems of Systems [12]:

- As defined in the Defense Acquisition Guidebook (DAG) 2008 version, a SoS is "a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities." (Note by J. Clark: The DAG 2008 version has not been published as yet, but was anticipated to contain this definition.)
- An SoS is defined as a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities [DAG 2004]. Both individual systems and SoS conform to the accepted definition of a system in that each consists of parts, relationships, and a whole that is greater than the sum of the parts; however, although an SoS is a system, not all systems are SoS. (Note by J. Clark: The DAG 2004 version was superseded by the DAG 2006 version referenced above, and the DAG 2008 definition of SoS was anticipated to revert back to this DAG 2004 definition.
- A family of systems (FoS) is defined as a set of systems that provide similar capabilities through different approaches to achieve similar or complementary effects [CJCS 2007(1)]. For instance, the war fighter may need the capability to track moving targets. The FoS that provides this capability could include unmanned or manned aerial vehicles with appropriate sensors, a space-based sensor platform, or a special operations capability. Each can provide the ability to track moving targets but with differing characteristics of persistence, accuracy, timeliness, etc." This definition is included for completeness. FoS are fundamentally different from SoS because, as CJCSI goes on to say, a family of systems lacks the synergy of a system of systems. The family of systems does not acquire qualitatively new properties as a result of the grouping. In fact, the member systems may not be connected into a whole. This guide specifically addresses SoS, but some of its contents may apply to FoS.
- SoS systems engineering deals with planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into an SoS capability greater than the sum of the capabilities of the constituent parts [DAG 2004]. Consistent with the DoD transformation vision and enabling net-centric operations (NCO), SoS may deliver capabilities by combining multiple collaborative and autonomous-yet-interacting systems. The mix of systems may include existing, partially developed, and yet-to be designed independent systems. (Note by J. Clark: As noted above, the DAG 2004 version was superseded by the DAG 2006 version.)

The SE Guide to SoS identifies 3 new SoS SE "roles":

- Translating Capability Objectives
- Understanding Systems & Relationships
- Monitoring & Assessing Changes

It is unclear why these three SoS SE roles are really "new." In my opinion they are included in the 16 technical and technical management processes defined in the DAG chapter 4, and are included in the SE Standards, V-Model, and Dual-V Model on which the DAG chapter 4 is based.

6 INCOSE's Definitions of System and SoS

Per the INCOSE SE Handbook [10]:

- A system is a combination of interacting elements organized to achieve one or more stated purposes.
- System of systems applies to a system-of-interest whose system elements are themselves systems; typically these entail large scale inter-disciplinary problems with multiple, heterogeneous, distributed systems.

Personally, I like this simple definition of SoS and would simplify it even further as follows:

- System of systems applies to a system whose system elements are themselves systems.

I like the above definition because it does not distinguish between dependent or independent systems, or any other characteristics of systems or SoS, and it supports my ideas about systems thinking discussed later in this paper.

7 The V-Model

Although not a SE standard, the V-Model is a very popular model of the SE process. The original V-Model [13] is shown in Figure 4. For top-down SE (i.e., forward engineering), the process starts on the upper left and goes to the upper right. For bottom-up (i.e., reverse engineering), it starts on the upper right and goes to the upper left.

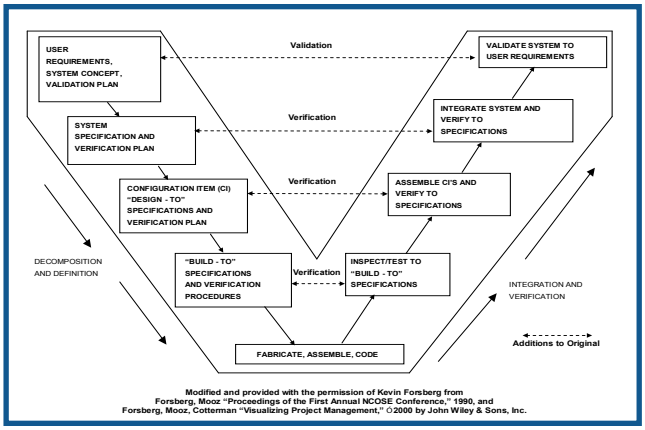


FIGURE 4 - ORIGINAL V-MODEL

The application of the original V-Model to a system or SoS is shown in Figure 5. This application is similar to the Building Block in which the Vs are repeated at each level of the SBS.

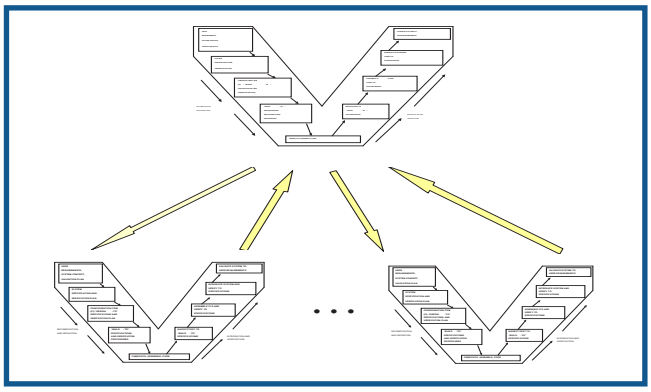


FIGURE 5 - SYSTEM OR SOS V-MODEL

An example of the detailed application of the V-Model to a system or a SoS is presented in Figure 6, the Dual-V Model [14].

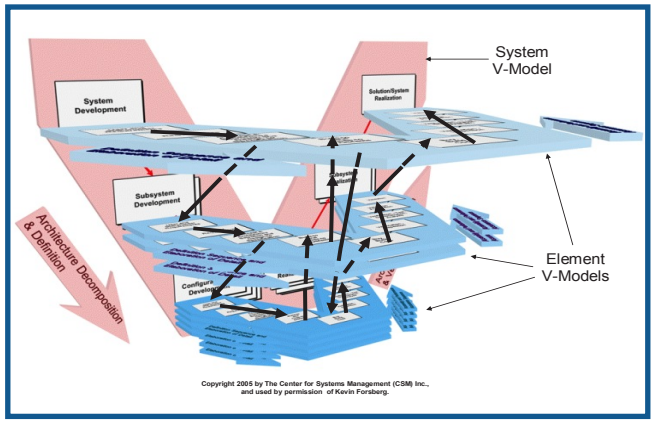


FIGURE 6 - DUAL V-MODEL

In this example in Figure 6 there are 1 system, 2 subsystems, and 4 Lowest Configuration Items (LCIs). The vertical backplane is the System-V and the horizontal planes are the Element-Vs. Each Element-V is the same as Figure 4 and is applied at each level of the System-V. A SoS-V would be depicted by adding the SoS in the backplane above multiple systems. Parts of an LCI would be depicted by adding the parts in the backplane below the LCI.

For top-down SE, in Figure 6, system requirements are allocated down to subsystems from the system "design-to" (i.e., requirements) specification on the left side of the System Element V. Each Subsystem Element V begins at its requirements process, passes its "build-to" (i.e., design) spec up to the system "build-to" spec process of the System Element V, ends at its validation process, and returns the result to the "fabricate, assemble, code" process at the bottom of the System Element V.

Similarly, subsystem requirements are allocated down to LCIs from the subsystem "design-to" specifications on the left side of the Subsystem Element V. Each LCI Element V begins at its requirements process, passes its "build-to" spec up to the subsystem "build-to" spec process of the Subsystem V, commences its "fabricate, assemble, code" process at the bottom of the LCI Element V, ends at its validation process, and returns the result to the "fabricate, assemble, code" process at the bottom of the Subsystem Element V.

The application of the original V-Model to a FoS is shown in Figure 7. Here, the Vs are sequentially nested into the page, signifying that for subsequent systems, some prior-system V assets are re-used un-modified; some assets are modified, used, and re-used later; and some assets are developed new, used, and re-used later. If a member of the FoS is a SoS, then the Vs continue down to the next lower-level as was shown in Figures 5 and 6.

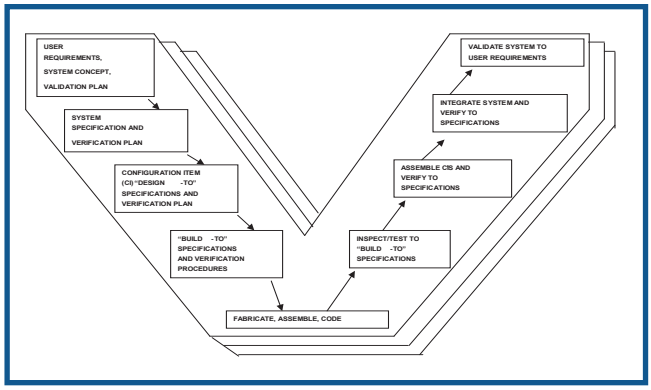


FIGURE 7 - FOS V-MODEL

8 Technical Baselines, Documents, Reviews, and Audits

An example of Technical Baselines, Documents, Reviews, and Audits for a system is shown in Figure 8 (the acronyms should be self-explanatory).

The top group shows the full menu of Technical Baselines, Documents, Reviews, and Audits from which the systems engineer selects (tailors) the appropriate ones for the system, subsystem, and LCI levels.

Requirements, functions, and preliminary design are shown on the left side. For top-down SE, these flow down from the system-level. System requirements allocated to a subsystem (system

allocated baseline) become the requirements baseline for that subsystem. Subsystem requirements allocated to a LCI (subsystem allocated baseline) become the requirements baseline for that LCI. From these requirements baselines come the functional, allocated, and product baselines for that level of the SBS.

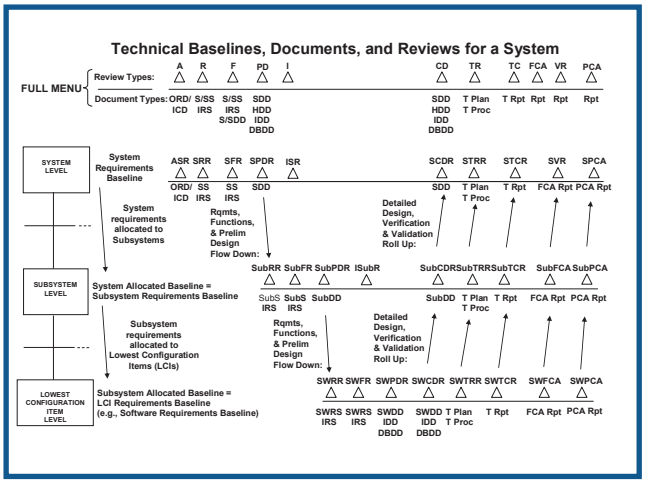


FIGURE 8 - TECHNICAL BASELINES, DOCUMENTS, REVIEWS, AND AUDITS FOR A SYSTEM

System requirements reviews precede subsystem requirements reviews that precede LCI requirements reviews. The same sequence applies to functional and preliminary design reviews.

Critical design, verification (e.g., test), validation, and audits are shown on the right side of Figure 8. These flow up to the system-level. LCI critical design reviews precede subsystem critical design reviews that precede system critical design reviews. The same sequence applies to verification and validation reviews and audits.

Extending Figure 8 to a SoS results in Figure 9. The same sequence of technical baselines, documents, reviews, and audits applies. A SoS is just another system, albeit more complex. Per the Defense Acquisition Guidebook: "SoSs should be treated and managed as a system in their own right, and should therefore be subject to the same systems engineering processes and best practices as applied to individual systems."

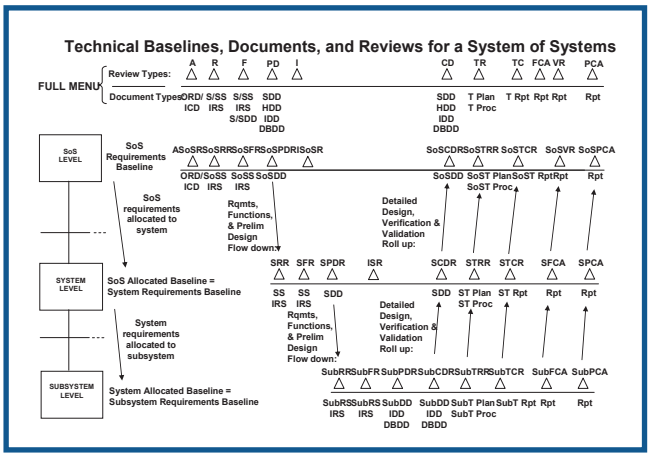


FIGURE 9 - TECHNICAL BASELINES, DOCUMENTS, REVIEWS, AND AUDITS FOR A SYSTEM OF SYSTEMS

9 Systems Thinking

In my opinion, systems-thinking involves the following:

- Everything and everyone (from the universe to the nucleus of an atom) is a system, a SoS, and a subsystem of a higher-order system
- Everything and everyone that exists/existed (things, people, thoughts, sayings, writings, actions, etc.) uses/used the systems engineering process
- You see everything and everyone as a system, a SoS, and a subsystem of a higher-order system
- You "Stand on the standards"
- You have "The Knack"

Every SoS is a system, every system is a SoS, and every system is a subsystem of a higher-order system, from the universe to the nucleus of an atom. It's a matter of degree of complexity. The universe is a system, a SoS, and a subsystem of a higher-order system (ponder which higher-order system); the milky way galaxy is a subsystem of the universe; the solar system is a subsystem of the milky way galaxy; the earth is a subsystem of the solar system; the weather system, the ecosystem, and the human system are subsystems of the earth; the digestive system is a subsystem of the human system; the stomach is a subsystem of the digestive system; the cell is a subsystem of the stomach; the molecule is a subsystem of the cell; the atom is a subsystem of the molecule; and the nucleus is a subsystem of the atom.

To me, the best example of a SoS is our human body. We are an SoS. We have a digestive system, a circulatory system, a respiratory system, a lymph system, a muscular system, a skeletal system, a reproductive system, a nervous system (sometimes I wish mine weren't so active!), etc. These systems have an interface, are integrated, and are interoperable. How fearfully and wonderfully made we are!

To me, the best example of a FoS is brothers and sisters. When brothers and sisters are separated (not interfaced), they are a FoS (a product line of their parents!). However, when they come together, have an interface, and are integrated, they become a SoS…hopefully they are interoperable!

Knowledge of the SE standards, the V-Model, and particularly the 3-dimensional Dual-V Model, will significantly aid systems thinking and its application to a system, a SoS, a subsystem, and a FoS.

Systems-thinking is really weird. Your thinking is transformed and your mind is renewed. You find yourself being transformed by the renewing of your mind! Enjoy systems thinking!

10 Complex SoSE and FoSE

So, how does all this apply to complex SoSE and FoSE? SoSE and FoSE address:

- Bounding and defining problem context
- Solution methods and techniques

- Solution tools
- Strategies for efficient solutions
- Various applications

This paper aims at providing and communicating focused solutions and propositions to the problems being encountered in complex SoS and FoS situations. Situations that form around complex SoS and FoS are characterized by lack of clarity, uncertainty, ambiguity, and limited understanding—stretching capabilities to manage and engineer effective responses.

SoSs and FoSs can be very complex. Now days, in evolutionary acquisition, many systems are using spiral development and thus their future behavior is unknown. (Not incremental development wherein their future behavior is known, and is just parceled-out into increments.) How do we integrate these evolutionary systems that are using spiral development into a SoS? Everyone is spiraling!

This is a complex situation. The approach to complex SoSE and FoSE is to surround and conquer. Apply a combination of top-down SE (forward engineering) and bottom-up SE (reverse engineering) using the Building Blocks and/or Vs. Apply the good 'ole SE standards to the Building Blocks and/or Vs. Treat each Building Block and/or V as a system, and vice versa. Focus on developing and controlling the interfaces between system elements and external systems. This is what integration and interoperability are all about.

Put the interfaces under formal Configuration Management (CM). Form Interface Control Working Groups (ICWGs). Consider assigning the ICWG the overall responsibility for CM of the system elements and the external systems, not just their interfaces. Document, baseline, and CM what you currently know about the interfaces. Exchange that information with all sides of the interfaces. Control what you don't know by exception. For example, if unpredictable behavior occurs and all sides of the interfaces agree, either alert the operator, record the data, apply artificial intelligence, or ignore the behavior.

Eventually, in time, the evolutionary behavior will settle down, become predictable, and quit spiraling. Until then, read and understand the SE standards and V-Models, apply good 'ole SE as evidenced in the SE standards and V-Models, and "Stand on the standards."

11 Conclusion

Is engineering a SoS really any different from engineering an ordinary system? Some believe that SoSE is "different" from SE, the SE processes are inadequate or insufficient for SoSE, and additional processes are needed. Others, like me, believe the SE processes as documented in the SE standards, and as illustrated in the V-Model and Dual-V Model, are a necessary and sufficient set of processes for SoSE, and no additional processes are needed. If you disagree, please get involved in the SE standards working groups and help us fix them.

12 References

1. Sheard, S. 2006. Is Systems Engineering for "Systems of Systems" Really Any Different? INCOSE Insight, Volume 9 Issue 1, October 2006.
2. IEEE 1220. 1994. IEEE Trial-Use Standard for Application and Management of the Systems Engineering Process. Copyright IEEE. http://shop.ieee.org/ieeestore
3. IEEE 1220. 1998 and 2005. IEEE Standard for Application and Management of the Systems Engineering Process. Copyright IEEE. http://shop.ieee.org/ieeestore
4. EIA/IS-632. 1994. Systems Engineering. Copyright © 1994, Government Electronics and Information Technology Association a Sector of the Electronic Industries Alliance. http://geia.org
5. EIA-632. 1998. Processes for Engineering a System. Copyright © 1999, Government Electronics and Information Technology Association a Sector of the Electronic Industries Alliance. http://geia.org
6. ISO/IEC 15288. 2002. Systems engineering – System life cycle processes. Copyright International Organization for Standardization (ISO), American National Standards Institute, 25 West 43rd Street, New York, NY 10036. (212) 642-4900, http://webstore.ansi.org.
7. ISO/IEC 15288. 2008. Systems and software engineering – System life cycle processes. Copyright International Organization for Standardization (ISO), American National Standards Institute, 25 West 43rd Street, New York, NY 10036. (212) 642-4900, http://webstore.ansi.org.
8. ISO/IEC TR 19760. 2003. Systems engineering – A guide for the application of ISO/IEC 15288 (System life cycle processes). Copyright International Organization for Standardization (ISO), American National Standards Institute, 25 West 43rd Street, New York, NY 10036. (212) 642-4900, http://webstore.ansi.org.
9. International Council of Systems Engineering (INCOSE) Systems Engineering Handbook, Version 3.1, Section 2.4. August 2007, http://www.incose.org.
10. Defense Acquisition Guidebook (DAG). 2006. U.S. Department of Defense (DoD).
11. Systems Engineering Guide for Systems of Systems. 2008, U.S. Department of Defense (DoD)
12. Forsberg, K. and Mooz, H. 1990. "Proceedings of the First Annual NCOSE Conference;" and Forsberg, K. Mooz, H. and Cotterman, H. 2000 "Visualizing Project Management," John Wiley & Sons, Inc., provided with permission.
13. The Center for Systems Management (CSM), Inc., www.csm.com, provided with permission.

13 Biography

John O. Clark is a Chief Engineer in the Information Systems Sector of Northrop Grumman. He is located at the Warfare Systems Engineering Department in Virginia Beach, Virginia. John currently supports the Information Systems Sector Directors of Process Management (SE Process) and Human Resources (SE Training). He led the development of and is the lead instructor for the INCOSE Certified Systems Engineering Professional (CSEP) course, and is an INCOSE CSEP. John has over 42 years experience applying SE and software engineering to the acquisition, development, verification/testing, operations, and support/maintenance of military command, control, communications, computer, intelligence, radar, sonar, electronic warfare, identification, weapon, network, scientific, and information systems. He is an active member of several Northrop Grumman Corporate Systems Engineering Advisory Group (SEAG) Working Groups and Communities of Practice; the Director of Education and Training of the INCOSE Hampton Roads Area Chapter; a member of the IEEE 1220 working group; a member of the EIA-632A GEIA G-47 SE committee; a co-lead of the GEIA G-47 Technical Reviews and Audits committee; and a member of the review teams for ISO/IEC 15288, 12207, TR 19760, TR 24748, and the INCOSE SE Handbook. He is an internationally recognized speaker and Subject Matter Expert in SE and teaches SE tutorials at major SE symposia. John earned a BS in Electrical Engineering from the Pennsylvania State University and an MS in Electrical Engineering from the State University of New York. He is an adjunct instructor in the MSSE curriculum at Old Dominion University in Norfolk, Virginia, and will be an instructor at Rutgers University.

The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems

James N. Martin

Abstract

There are seven different systems that must be acknowledged and understood by those who purport to do systems engineering. The main system to be engineered is the Intervention System that will be designed to solve a real or perceived problem. The Intervention System will be placed in a Context System and must be developed and deployed using a Realization System. The Intervention, when installed in the Context, becomes the Deployed System which is often different in substantial ways from the original intent of the Intervention. This Deployed System will interact with Collaborating Systems to accomplish its own functions. A Sustainment System provides services and materials to keep the Deployed System operational. Finally, there are one or more Competing Systems that may also solve the original problem and will compete for resources with your Deployed System. All seven systems must be properly reckoned with when engineering a system.

Introduction

The Analogy

"Shichinin No Samurai," the 1954 film classic directed by Akira Kursawa, is an apt illustration for the plight of the systems engineer. The Seven Samurai were the mighty warriors who became the seven national heroes of a small town. A poor village under attack by bandits recruits seven unemployed samurai to help them defend themselves. The notion of the "seven samurai" described in this paper illustrates the seven systems that are underemployed in the classical practice of systems engineering. When these 7 Samurai are employed with proper consideration and enthusiasm, they will become the seven national heroes of your small town (the system development project).

The Context System

Let us examine the first of seven systems—the Context System (S1). Context is "the set of facts or circumstances that surround a situation or event." (WordWeb) It is the set of "interrelated conditions in which something exists or occurs." (Webster's New Collegiate) Context also goes by the name "Environment" which means the "circumstances, objects, or conditions by which one is surrounded." (ibid) Context originally meant the "weaving together of words" and leads us to the more common connotation of the term: "the parts of a discourse that surround a word or passage and can throw light on its meaning."

Linguistics?

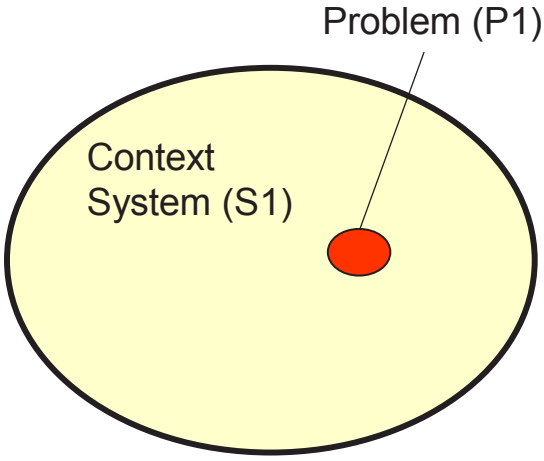
Now why would we systems engineers bother with the linguistic aspects of the word Context? Precisely because systems engineering is very much about finding the correct words to describe the problem to be solved by the engineering solutions we intend to create. In the words of Jack Ring, the systems engineer's job is to "language the project." [Ring et al 2000]

The Problem System

The Context is where the Problem P1 resides. Aspects of Context can be, and often must be, reverse engineered to discover the constituents of the problem's environment. We must understand the relationships of the constituents to each other and to the problem itself. Is there something in the context that is causing the problem? If we solve the "problem" but do not address the cause(s), will the problem merely evolve into something more dreadful? Is the initial statement of the problem really the problem or merely a symptom of the real problem?

Object Oriented Thinking

Using the object oriented approach, the relevant items in the environment can be identified as "objects." The objects are identified as either types or instances. These object types (or classes) and instances can be depicted using a Class Diagram. An older, but still useful, technique for this contextual analysis is the ERA approach. This involves identifying the relevant Entities, the Relationships between those entities, and the Attributes of each entity or relationship. Below is an illustration of an ERA diagram developed during the context analysis phase of a project to develop the Observing System Architecture for NOAA (National Oceanic and Atmospheric Administration). [Martin 2003]



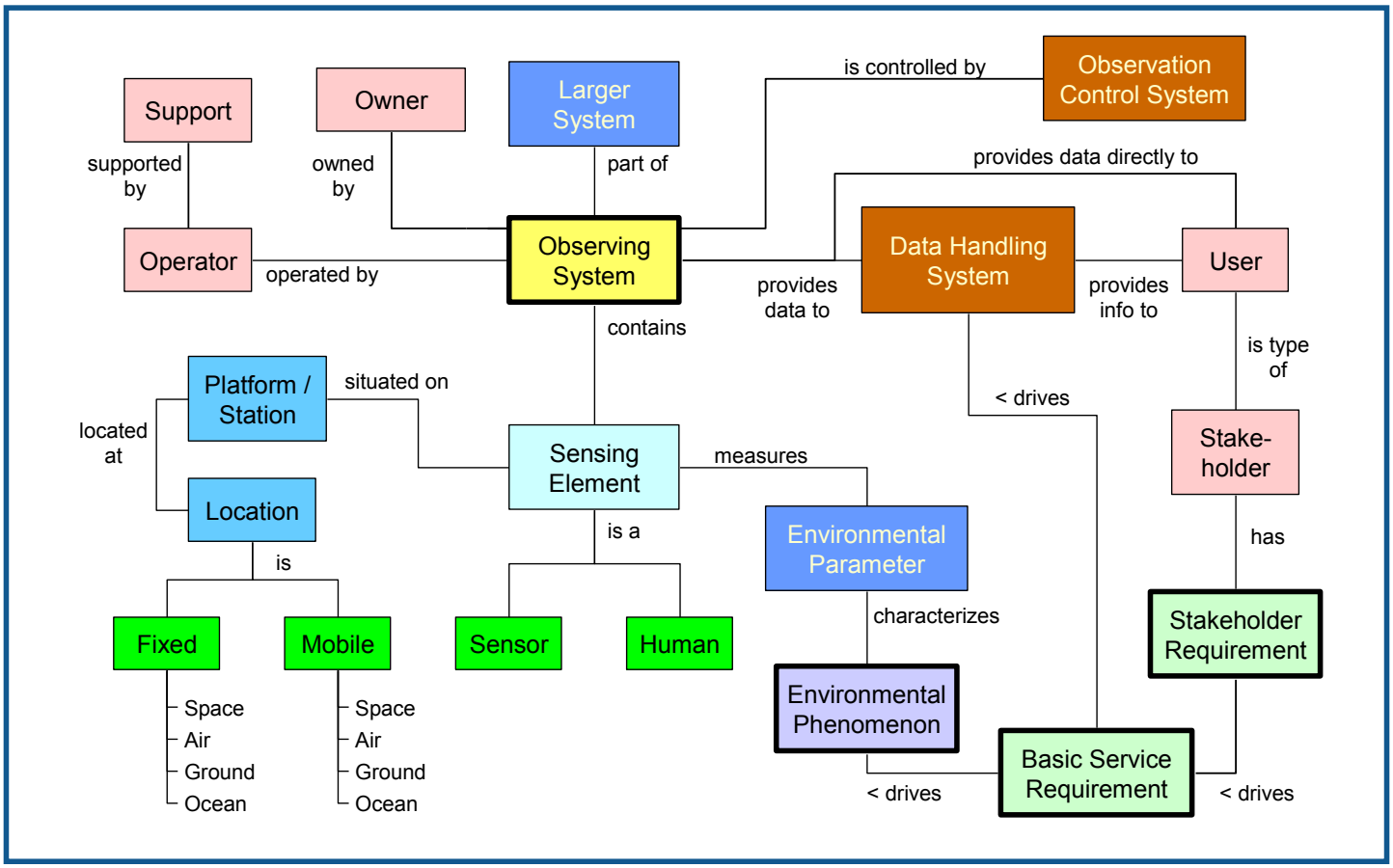


FIGURE 1 – ERA DIAGRAM FOR THE NOAA OBSERVING SYSTEM ARCHITECTURE

The ERA diagram above does not illustrate the attributes, so it is more correct to call this an ER diagram. Sometimes the attributes of each entity are listed inside each entity box. In the case above only the entity type names are shown. Whether you use a Class Diagram or Entity Relationship Diagram, you are really defining the "scope" of the problem to be solved

Metamodeling

The basic structure of any problem can be captured in a "metamodel." Often the metamodel you need to use for your problem of interest is already captured in your favorite tool or methodology (e.g., UML or IDEF0). The problem P1 for NOAA was to identify the deficiencies and excess capacities of the 100 different observing system types owned or operated by NOAA. The ERA diagram above is a depiction of the metamodel for the NOAA problem situation. A good description of the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a metamodel are given at [metamodel.com]:

A meta-model is an explicit model of the constructs and rules needed to build specific models within a domain of interest. A valid meta-model is an ontology, but not all ontologies are modeled explicitly as meta-models. A meta-model can be viewed from three different perspectives:

1. *as a set of building blocks and rules used to build models*
2. *as a model of a domain of interest, and [emphasis added]*
3. *as an instance of another model.*

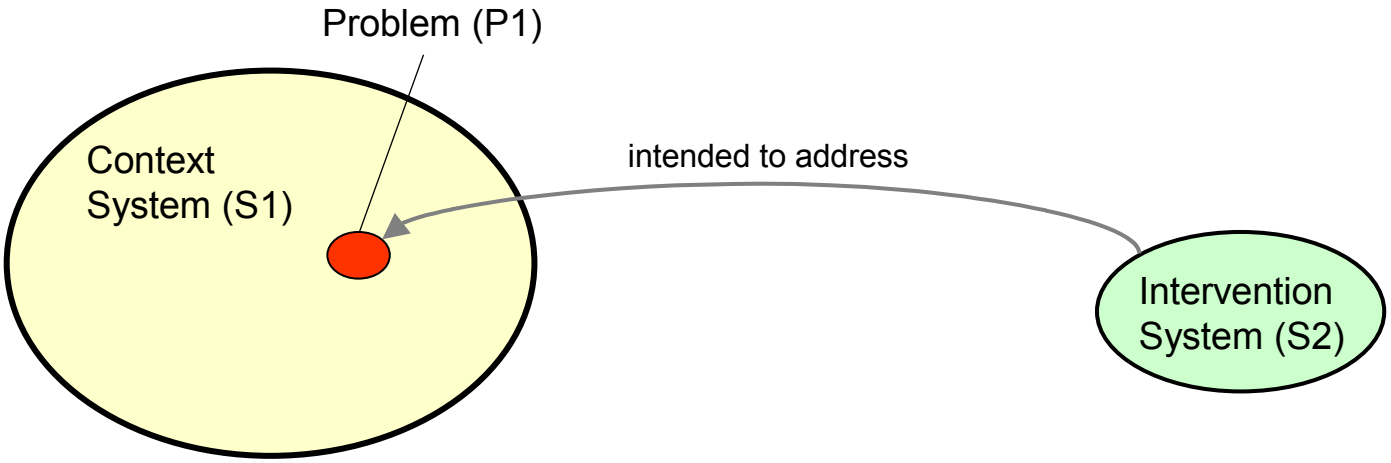
When comparing meta-models to ontologies, we are talking about meta-models as models (perspective 2).

Note: Meta-modeling as a domain of interest can have its own ontology. For example, the CDIF Family of Standards, which contains the CDIF Meta-meta-model along with rules for modeling and extensibility and transfer format, is such an ontology. When modelers use a modeling tool to construct models, they are making a commitment to use the ontology implemented in the modeling tool. This model making ontology is usually called a meta-model, with "model making" as its domain of interest.

The Intervention System

Now we must look for a solution to the problem. Let us call this intended "solution" the Intervention System (S2). The Intervention System is intended to address the Problem P1. It is the system to be engineered using the systems engineering process, methods, and tools. This is the central focus for the development project that is established to be a profitable venture for systems development companies. But to ensure that the so-called "requirements" for this system are valid and complete, full and proper consideration must be given to all seven "samurai."

These samurai will bring misery to all if left loose to roam at will across the countryside.



Preventing the Undesirable

Intervention is "action affecting another's affairs: an action undertaken in order to change what is happening or might happen in another's affairs, especially in order to prevent something undesirable" (dictionaries.com) Intervention can be seen as a sort of perturbation of the Context, as a form of engagement with the evils of the world. Intervention has two types: intermediation and mediation. Mediation is a form of negotiation to "resolve differences conducted by some impartial party." (WordWeb) Intermediation is acting "between parties with a view to reconciling differences." (ibid)

Achieving Reconciliation

What are these differences to be reconciled? There will be people in the Context that would like the situation to be different, better somehow. The systems engineer should devise an Intervention System that settles the differences between the way things are now, the "as-is" situation, and the desired state of affairs after intervention, the "should-be" situation. It is important to recognize that the systems engineer must be an unprejudiced, third party to this situation. When a systems engineer is "involved" in the situation, it is difficult to be impartial and just when deciding how best to "solve" the problem.

Systems Architecting

The conceptual nature of the Intervention System is often understood through the efforts of "architecting." [Maier and Rechtin 2000] Bear in mind that S2 may include 'mod kits' to the Problem System and the Context System. A depiction of the system architecture is created by development of an architectural model which uses the metamodel's foundational building blocks—the element types and structures discovered in the Context during analysis of S1. Architecture can be thought of as "an arrangement of feature and function that accomplishes some objective." [Ring 2001]

The Realization System

For the Intervention System to come about, it must be brought into being by a Realization System (S3). The Realization System consists of all the resources to be applied in causing the Intervention System to be fully conceived, developed, produced, tested, and deployed.

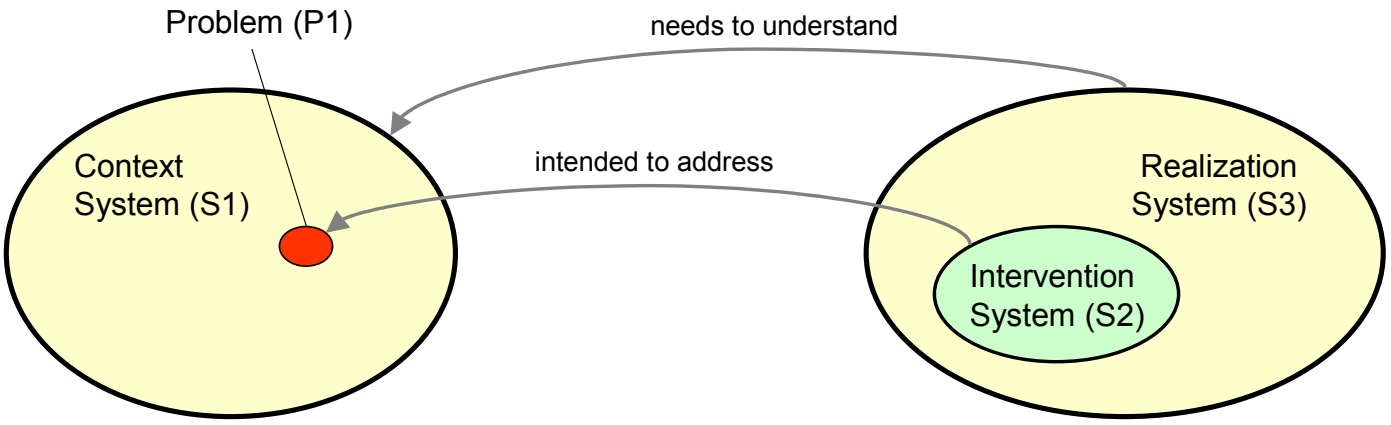
The Realization System will consist of a wide variety of things, some tangible and some not:

a) people & organizations

b) facilities & equipment

c) materials & supplies

d) services & utilities



e) processes & methods

f) tools & techniques

g) policies & procedures

h) data & information

i) knowledge & wisdom

j) and so on

All of these things interact in complex ways to bring about a solution to the real or perceived problem. The Realization System needs to "understand" the Context and the Problem contained therein. How can this be? How can a system have understanding? Well, people and organizations have understanding and they are an intimate part of the Realization System. Understanding is also captured in policies and procedures, and in knowledge and wisdom. This is the reason that knowledge management has become so important for better execution of the systems engineering process. A good way to model and understand the Realization System is through knowledge modeling. [Lillehagen et al. 2003]

Enterprise Architecture

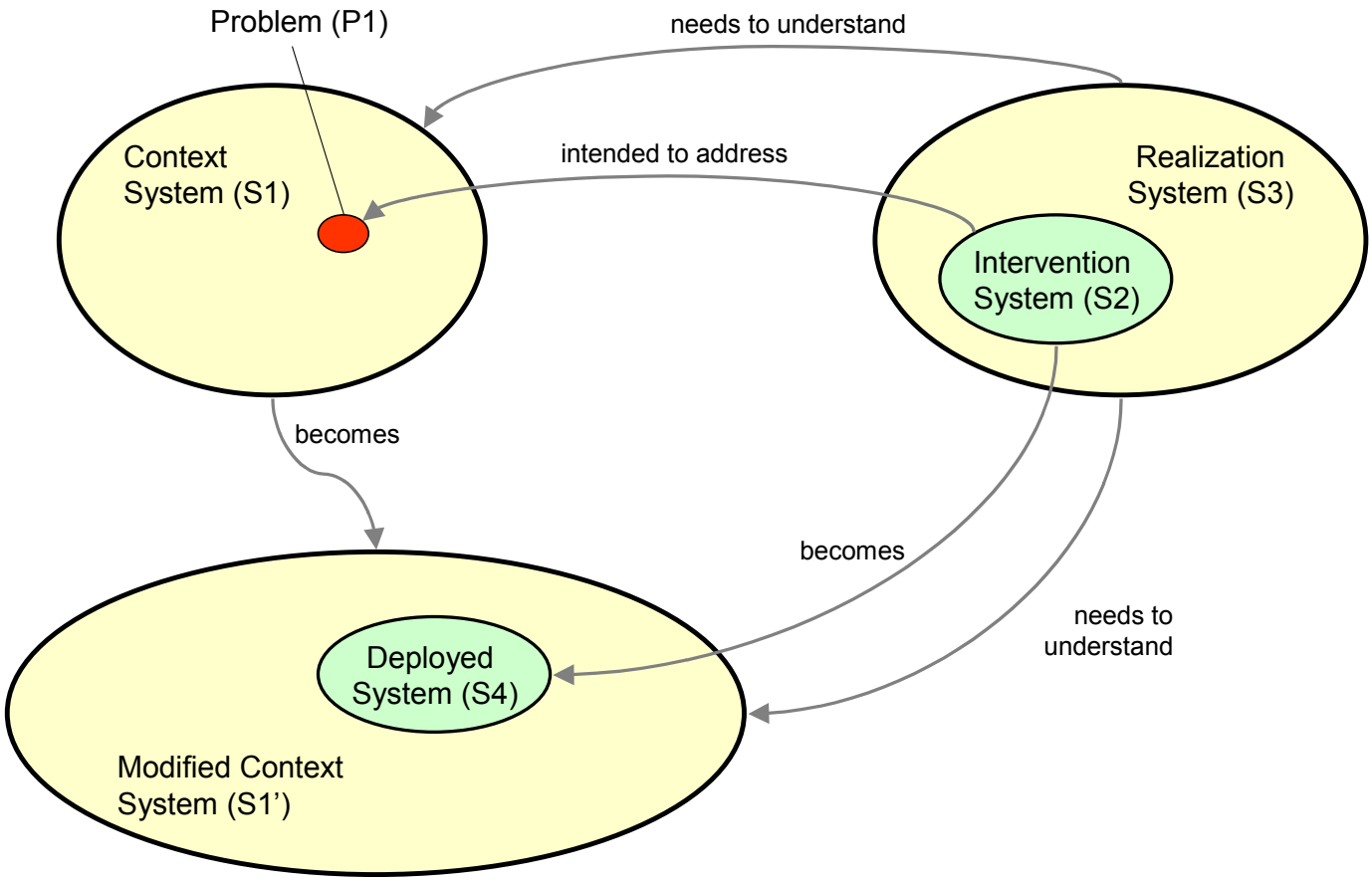
Often this Realization System is known as an Enterprise. An Enterprise is a purposeful or industrious undertaking (especially one that requires effort or boldness). It usually involves many organizations that contribute their resources to the "owning" organization of that enterprise. The organizational resources can be either tangible (e.g., funding and people) or intangible (e.g., goodwill and enthusiasm). Enterprise architecting is a relatively new field of endeavor but is gaining popularity as the complexity of current ventures (and adventures) becomes more recognized. A good description of enterprise modeling can be found in [Vernadat 1996].

The Deployed System

Even though we have the best of intentions, the system we design, develop, and build will often morph into something else once it is transitioned to its final destination. This Deployed System (S4) is intended to be the same as S2, but variability often occurs due to malicious intent, inadvertent errors, performance degradation, deployment pressures, interaction between the new system and its environment (S1/S4 coupling), and so on.

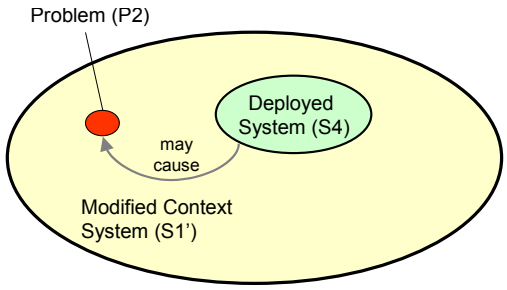
Modified Context

The new system will often change the original Context into a Modified Context (S1') in ways that are sometimes beneficial, but more often than we would like this change is to the detriment of those we were trying to help. Furthermore, several years may have passed since the original analysis of the Context was conducted and when the Intervention System was ready to deploy. The world changes without asking our permission. The original "customer" has often moved on. The people we interviewed to assess the situation may have already solved their problem through other means.



Unintended Consequences

Notice that the Realization System also needs to understand the Modified Context. The systems engineers must be cognizant of how their proposed solution might change the original Context, and perhaps even become worse than the original problematic situation. Never forget the Law of Unintended Consequences. [Norton]



The law of unintended consequences, often cited but rarely defined, is that actions of people—and especially of government—always have effects that are unanticipated or "unintended." Economists and other social scientists have heeded its power for centuries; for just as long, politicians and popular opinion have largely ignored it.

The concept of unintended consequences is one of the building blocks of economics. Adam Smith's "invisible hand," the most famous metaphor in social science, is an example of a positive unintended consequence. Smith maintained that each individual, seeking only his own gain, "is led by an invisible hand to promote an end which was no part of his intention," that end being the public interest. "It is not from the benevolence of the butcher, or the baker, that we expect our dinner," Smith wrote, "but from regard to their own self interest."

This Law clearly applies in the economic domain, but is equally applicable, if not more so, in the domain of systems engineering. We must heed this Law if we are to be successful in engineering systems that are appropriate for Context Systems that are complex and adaptive. [Holland 1998] The best situation is where the proposed solution is adaptive to changes in the environment to compensate for environmental changes. [Holland 1995]

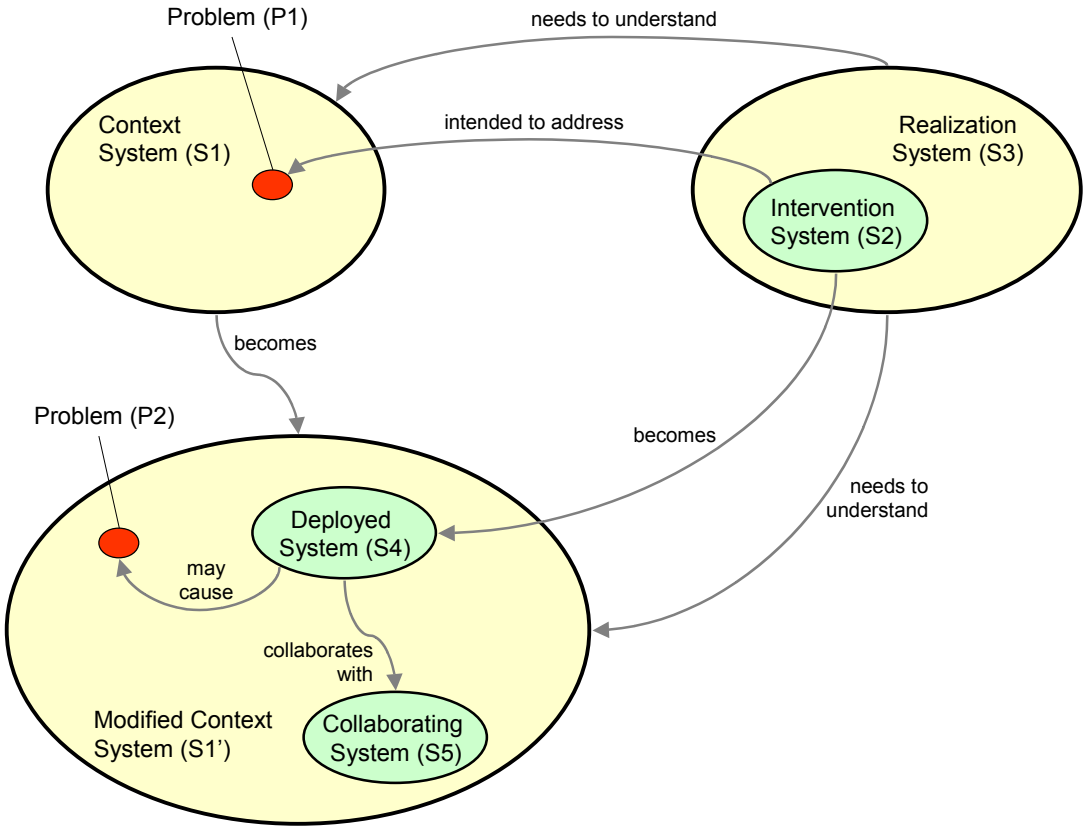
A New Problem

Not only has the original Context been modified, but our newly deployed system often causes a new Problem (P2). More work for the unemployed, you say. Yes, but your company might go out of business due to litigation or bankruptcy before you have a chance to rid the streets of the homeless.

One reason for the change in Context is that it contains people. People are highly complex and adaptive. Therefore you can expect your system "solution" to be used improperly, controverted, damaged (sometimes even unintentionally), bypassed, and so on. People are good at finding things to do with your system that were not part of your original intent. Hence, be forewarned—your solutions can sometimes cause more problems than they solve. As system development progresses, it is essential to be cognizant of mutations in the Context and adjust the development goals accordingly.

The Collaborating System

When we designed our Intervention System, we may have realized that we had access to certain resources that could solve



only part of the problem. What to do? We made agreements with industry partners, or we decided to make our system modular so that it fits into someone else's platform. We may have decided to incorporate standard interfaces so our system will work with other systems in a synergistic fashion. This can be a win-win situation. But there are times when this can backfire due to "emergent" properties that are undesirable. Why, our system worked with that other system in our integration lab—why doesn't it work out in the field?

Unintended Collaborations

Don't forget that the Collaborating Systems also interact with the Context (the Modified Context, really) and these changes in the environment could affect how your system interacts with its intended collaborators. And then there are the Collaborating Systems that you never intended to interact with. Someone else can come along and "plug in" to your system. This could be great since it could make your system much more valuable to the customer, more indispensable. Or this could be bad since this new Collaborating System could be performing some of the functions of your systems (those that are perhaps not quite as efficient or effective as they could be).

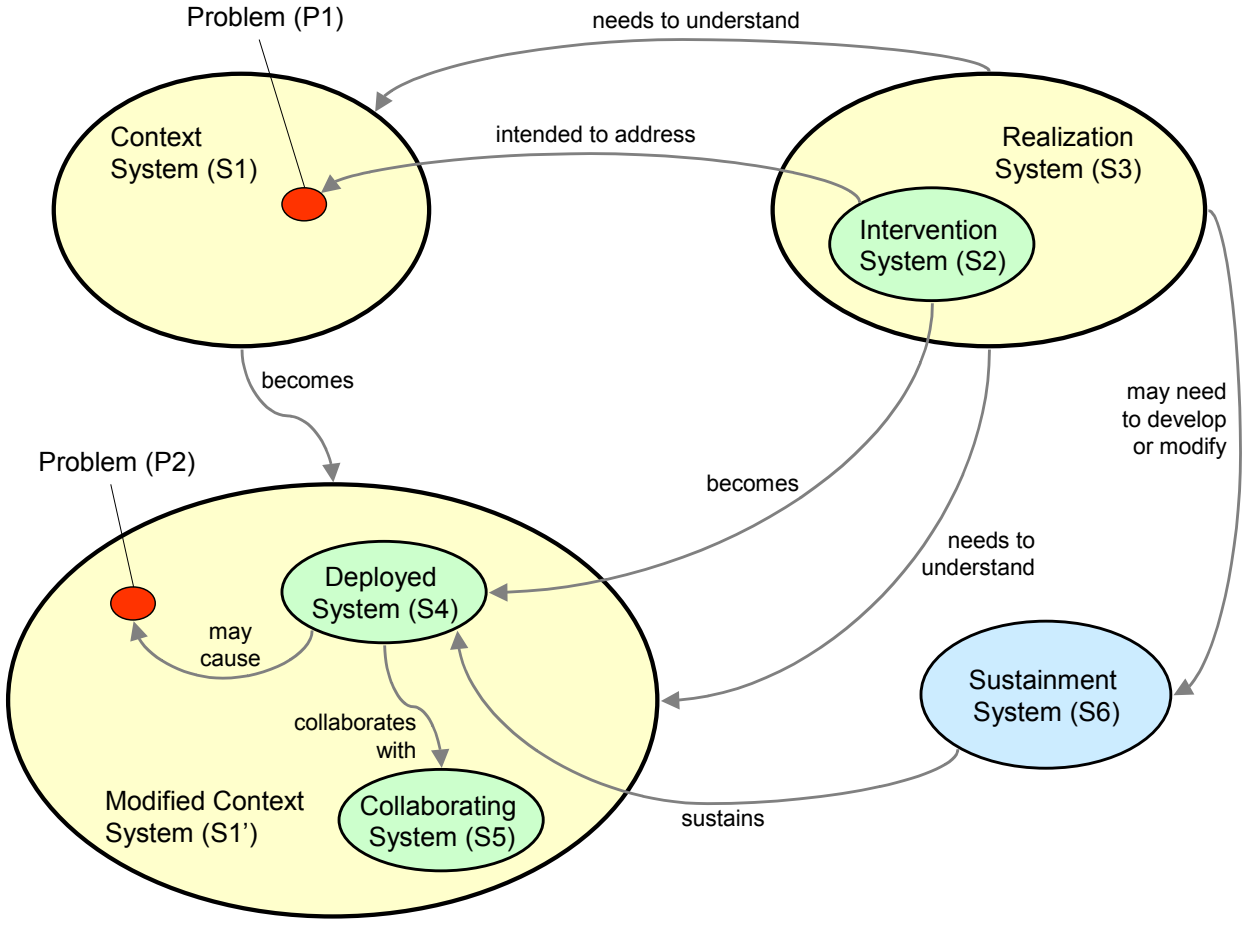
The Sustainment System

Now we come to the Sustainment System (S6) that provides the necessities and support such as fuel, energy, spare parts, training, customer hotline, maintenance, waste removal, refurbishment, retirement, and so on. It is quite important for the Intervention System to take into account the capabilities and limitations of the Sustainment System. In many cases, the Realization System may need to modify (or even develop parts of) the Sustainment System.

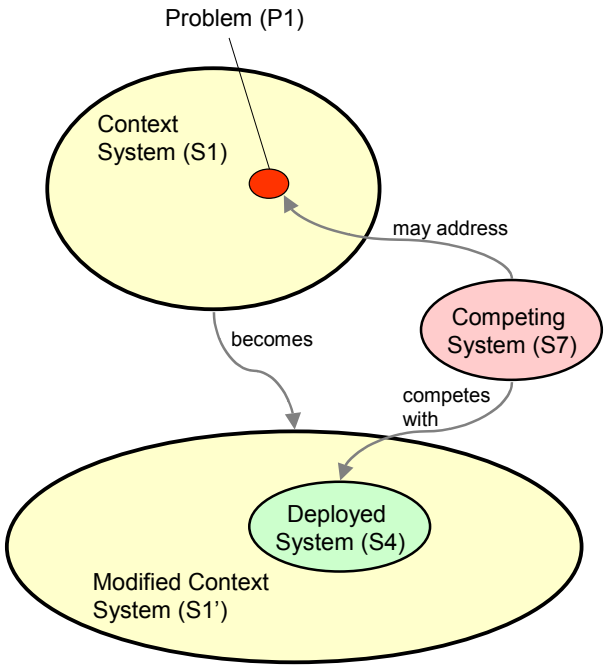
The Sustainment System is often thought to be under the purview of the logistics support engineer. Logistics is a relatively mature discipline that can address most of the concerns related to sustainment. (See [Blanchard 1998] for a good summary of logistics support tools and techniques.) But the systems engineering team needs to work with logistics early in the game to ensure these issues are addressed before "unsupportable" features and functions are captured in the solution concept. The sustainment costs are typically ten to twenty times the cost of development. Therefore, it is worthwhile to spend considerable effort in understanding the sustainment issues before proceeding too far along the path of system development.

The Competing System

Now if life were not complicated enough already as a systems engineer, we must also deal with the Competing System(s) (S7) that may also address all or parts of the original Problem P1. It may provide similar or identical features and functions as your proposed System solution. The Competing System also competes for resources used by the Deployed System. Furthermore, you need



to avoid being blindsided by concurrent developments or advances in technologies that might render the Deployed System obsolete.



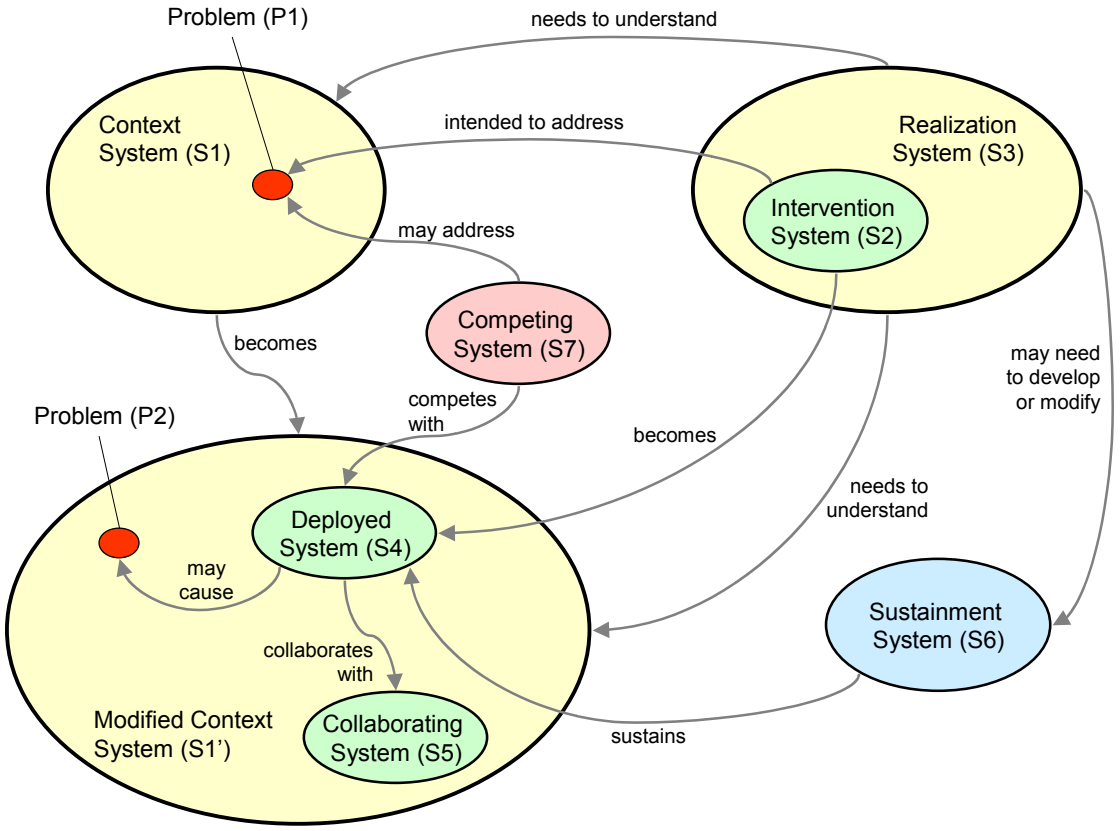
Summary

We can now summarize the interactions between these seven samurai systems:

1. Context System (S1) contains a Problem (P1)
2. Intervention System (S2) is intended to address P1
3. Realization System (S3) brings S2 into being
4. S2 is a constituent of S3
5. S3 needs to understand S1
6. S3 needs to understand the Modified Context System (S1')
7. S3 may need to develop or modify the Sustainment System (S6)
8. Intervention System (S2) becomes Deployed System (S4)
9. S1 becomes the Modified Context System (S1')
10. S4 is contained in S1'
11. S4 collaborates with one or more Collaborating Systems (S5)
12. S4 is sustained by Sustainment System (S6)
13. S4 may cause new Problem (P2)
14. Competing System(s) (S7) may address the original Problem (P1)
15. S7 competes with S4 for resources and for the attention of users and operators

Holistic Systems Thinking

By understanding these fifteen interactions, we now have a better chance of understanding the "whole picture." We need to model all aspects of the entire situation to ensure our system solution is indeed the best way to solve the problem. The essential holistic view is illustrated below.



Is it any wonder why Systems Engineering (SE) is so difficult? For decades we have not explicitly acknowledged nor understood the various systems that must be addressed when engineering a solution for a complex, adaptive situation.

This new paradigm of the "Seven Samurai" must be considered in the application of SE process, methods, tools, and standards if we expect SE to address the increasingly complex problems of the 21st century.

References

1. Blanchard, Benjamin S., Logistics Engineering and Management, Prentice Hall, 1998.
2. Holland, John N., Hidden Order: How Adaptation Builds Complexity, Addison Wesley, Reading MA, 1995.
3. Holland, John N., Emergence: From Chaos to Order, Addison Wesley, Reading MA, 1998.
4. Lillehagen, Frank and Solheim, Helge G., "The foundations of AKM technology." Concurrent Engineering Conference, 2003.
5. Maier, Mark W. and Rechtin, Eberhardt, The Art of Systems Architecting. CRC Press, 2000.
6. Martin, James, "On the Use of Knowledge Modeling Tools and Techniques to Characterize the NOAA Observing System Architecture." Proceedings of the INCOSE Symposium, 2003.
7. Norton, Rob, "Unintended Consequences," The Library of Economics and Liberty. http://www.econlib.org/library/Enc/UnintendedConsequences.html
8. Ring, Jack, and A. Wayne Wymore, "Concept of Operations (ConOps) of a Systems Engineering Education Community," Proceedings of the INCOSE Symposium, 2000.
9. Ring, Jack, "Discovering the Architecture for Product X," Proceedings of the INCOSE Symposium, 2001.
10. Vernadat, Francois, Enterprise Modeling and Integration: Principles and Applications. Kluwer Academic Publishers, 1996.

Biography

James Martin is a systems architect and engineer at The Aerospace Corporation developing solutions for information systems and space systems. Mr. Martin led the working group responsible for developing ANSI/EIA 632, a US national standard that defines the processes for engineering a system. He previously worked for Raytheon Systems Company as a lead systems engineer and architect on airborne and satellite communications networks. He has also worked at AT&T Bell Labs on wireless telecommunications products and underwater fiber optic transmission products. His book, Systems Engineering Guidebook, was published by CRC Press. Mr. Martin is an INCOSE Fellow and leader of the Standards Technical Committee. *James.Martin@incose.org*

Legacy Sustainment: Slashing the costs of COTS Obsolescence Management

Kaye Porter

Abstract

Today's advanced technology is lasting longer than predicted and rapidly becoming more complicated to sustain. When 80% of a budget is in sustainment, program managers are forced to take another look at commercial off-the-shelf, end-of-life-cycle costs—and how to avoid them. The reality is, current understanding of obsolescence management needs a refresh just as much as many of our defense systems.

Introduction

In 2003, Senator Joseph A. Lieberman[1] reported that "DoD and intelligence agencies will need to be first adopters of the most advanced integrated circuits, and will be increasingly dependent on such chips for a defense and intelligence edge." In the time since his report, the embedded industry has seen the evolution of this prediction, not only in defense industry technology and the need for interoperability, but also in the growing challenges sustainment teams face when it comes to reliably supporting legacy technology.

Today, many of our defense programs are seeing life-cycles lasting longer than originally intended. By the time a ship or plane is decommissioned, it has probably been in use longer than the original life expectancy. It has also seen longer exposure to harsh environments, extreme temperatures, hard usage, and in some cases direct combat. If the reliability of warfighter platforms and other systems is impacted by unreliable or unfunded requests for spares and repairs, our troops aren't getting the support they need.

As defense technology ages, the realities and costs of sustaining it get worse and no one can afford to have a specialized engineer for every problem. Embedded systems are far more complex than at the time of the B-52 or even the more "recent" USS Mount Whitney, launched in 1970—whose life-cycle is now projected out until 2039. The F-22 saw four redesigns before the program was ultimately shut down. The F-35 is currently facing similar concerns having been designed in the '90s, with the first planes starting testing ten years later. In a climate of PBLs, budget cuts, commercial off-the-shelf (COTS) obsolescence, and increased counterfeit risks; *our current understanding of obsolescence management needs a refresh just as much as many of our defense systems.*

(See Figure 1: In 2012 the Air Force Science and Technology Board estimated that Weapon System Sustainment (WSS) was nearly 32 percent of the Air Force O&M FY2012 budget.)

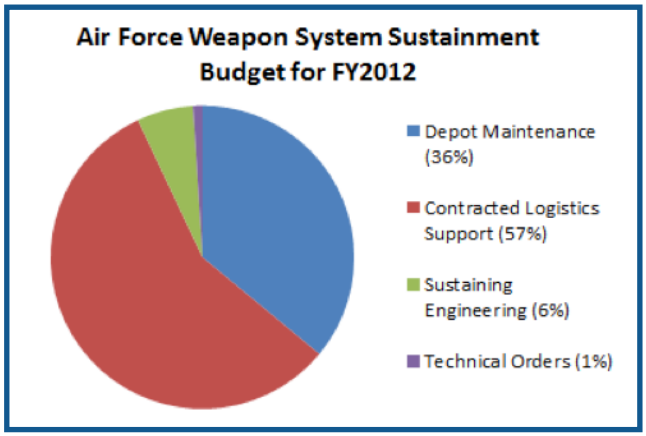


FIGURE 1

Legacy systems power much of our military capability and the average defense program with a life-cycle of 30 years will spend 60%-80% of its life-cycle and total ownership costs in sustainment.[2] A 2011 study published by the Air Force Science and Technology Board and sponsored by the National Research Council (NRC) estimated that 65 percent or more of Weapon System Sustainment (WSS) came from supporting the system, and in 2012 was nearly 32 percent of the Air Force O&M FY2012 budget.[3] (See Figure 2: The average defense program with a lifecycle of 30 years will spend 60%-80% of its lifecycle and total ownership costs in sustainment. These costs go up as technology evolves and systems become more and more complex.

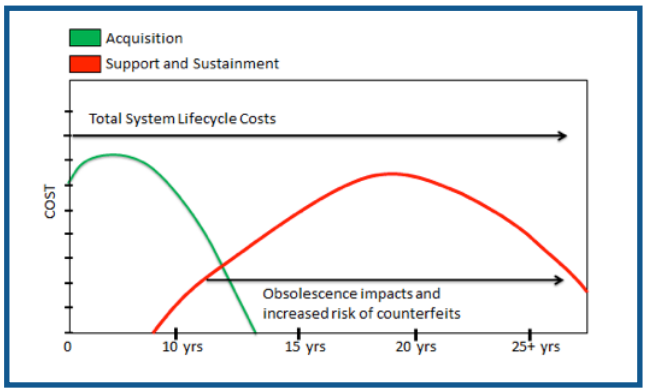


FIGURE 1

Additionally, with the growing budget cuts and the National Defense Authorization Act for FY2012 (NDAA for FY2012) anti-counterfeiting laws, the defense industry more than

1 J. Lieberman, "White Paper: National Security Aspects of the Global Migration of the U.S. Semiconductor Industry," Ranking Member Airland Subcommittee United States Senate Armed Services Committee (2003).

2 Presentation by DAU "DoD Life Cycle Management (LCM) & Product Support Manager (PSM) Rapid Deployment Training"

3 Department of the Air Force, FY 2012 Budget Estimates, Operations and Maintenance, Vol. II.

ever requires aggressive solutions. Beyond growing costs of authentication and counterfeit avoidance, defense legacy programs demand predictable forecasts as products age in order to continue to receive enough advance funding.

In an ideal world, long-lasting critical equipment would be easy to support, repair, have advance EOL/LTB notifications, and relatively stable and predictable long-term costs. However even if last-time-buy (LTB) and end-of-life (EOL) notices were easily available, in the face of sequestration, a typical 3- to 6-month notification won't provide proactive funding visibility for the parts long-lasting systems require. Add to that, that system sustainment costs are rarely predictable, because costs increase with system complexity and age—and often aren't planned for until sustainment and obsolescence issues have already become a problem. Without a proactive legacy management plan to predict sustainment requirements, it is common for logistics and engineering teams to be left hunting for parts long after a LTB or EOL event, leaving critical military programs vulnerable to the costs of obsolescence.

A Shift in Perspective: From EOL Management to Legacy Management

The side effects of COTS embedded obsolescence, at least four, are well known to the defense community; (1) Components that have been pre-stocked may not have been stored or cared for properly, resulting in batches of defective materials that fall-out…often discovered only after spares have been assembled. (2) Counterfeit components enter the supply chain, resulting in failure risks, testing costs, and evolving regulations around authentication and traceability. (3) Ongoing repair and sourcing replacements push out timelines and can be tricky without the original IP data and test specifications. (4) Re-engineering also requires recertification, bringing with it additional costs and timeline delays, etc…

When preparing for total life-cycle sustainment, obsolescence is a reality, but to effectively manage it, the work can't start at the point of EOL. Currently the performance of some sustainment teams is measured against their resolution of "obsolescence mitigation incidents", rather than incident prevention. This process leaves critical components at risk because no action is taken until there is a red flag raised.

One way to estimate reactive cost impacts on an embedded board is by counting ASICs. Each obsolete ASIC could cost up to $1.5 million per component, not including recertification fees, and costs of downtime and repair. When a global product integrator discovered the $15-$30 ASIC components they relied on to support an in-flight navigation system were no longer available, it created an abrupt transition of focus for the program. Facing component shortages and without a lifetime buy option, follow-on support became impossible without either a single component redesign or a "proactive" $3.3M system redesign, which was never budgeted on a $750 board.

In order to secure legacy systems, the goal is obsolescence prevention or "Legacy Sustainment" and a collaborative approach across the supply chain needs to be part of the process. Total life-cycle sustainment needs to be a focus from the outset of a defense program. Bills of materials (BOM) and critical components need to be monitored throughout the process with ongoing health/risk analysis. In the case of custom systems, a defense contractor or depot may have access to the BOM, however, with COTS CCAs/SBCs and an embedded system, collaboration with the OEM is critical because the BOM will not be available.

Accountability for sustainment needs to happen up-front and that accountability needs to have follow-through; even if the original program manager retires before the sustainment goes into effect. Sustainment collaboration needs to happen across the warfighter supply chain, from acquisition, to logistics, to design and engineering teams. Forward thinking processes involve the best thinking from key supply chain players along a program's life-cycle.

While it is important to already have these processes in place for any long-life program, it is fundamental for programs migrating to a firm-fixed or PBL agreement. Otherwise a defense suppliers and government operations and sustainment teams will likely face the reality that, with each contract cycle, their options for affordable support diminish as the system ages…and parts, adequate funding, and the brain trust all fade away.

A System and Application Level Strategy

Ideally both obsolescence management and legacy sustainment would result in keeping long-lasting, critical technology healthy and active. But unfortunately no matter how "proactive" you're trying to be, end-of-life triggered obsolescence management at a part by part approach becomes a rigid and reactive process, resulting in unplanned budget impacts and last-minute redesign. (See Figure 3: if you wait until EOL to start obsolescence management, you are already in a reactive place and several years behind on the parts necessary to sustain critical systems.

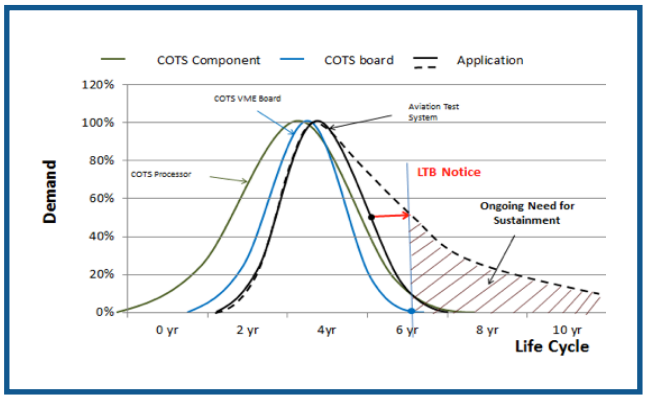


FIGURE 3

A proactive perspective on the entire system helps predict important events such as technical refresh and part sourcing. The

proactive assessment then allows sustainment and DMSMS teams to better prepare for the required time and funding in advance. Ideally, an onsite sustainment team has these solutions outlined during the design phase, in order to prevent reactive obsolescence mitigation. However, that isn't always the case, and an experienced extended service provider can help close critical gaps.

A high-level picture of the entire system life-cycle also needs to include both COTS and custom technology. A life-cycle assessment can be done at any point in a system's life-cycle, and includes information like: current rate of usage and repair, historic usage in different environments (if applicable), contracted use hours (flight hours/run hours), commonalities across the board and component level, length of expected life-cycles, and current stock or materials on hand.

Proactive planning allows for:

- Usage and life expectancy based on active contract periods, with a sustainment plan that expands beyond current contract timeframes.
- Health assessment of the system, especially on high-priority and critical parts.
- Funding forecasts to sustain the system over time.
- Options for repairs and replacements of embedded systems.

In the end, it is critical to acknowledge obsolescence isn't an event that might happen once in an application's life cycle; rather, it's inevitable for long-life systems. To support an application through its end-of-life, programs require ongoing planning and lifecycle analysis to determine tasks, accountability, procedures, and maintenance schedules. Proactive legacy sustainment approaches, like GDCA's PLM+ Solutions provide a forward-thinking, flexible framework where the realities of COTS board-level obsolescence can be planned and budgeted for in advance. With this mindset, EOL is simply something that happens along the way, as opposed to a driving event.

Biography

Kaye Porter is Marketing Manager for GDCA. She has written several articles on collaborative, proactive obsolescence management, and spoken at conferences including ERAI, CALCE, and DMSMS on counterfeit avoidance and sustaining long-term system-level embedded technology for both the medical and military industries. Kaye believes that while obsolescence is a reality, end-of-life (EOL) doesn't have to be. Because of this, her mission is to challenge the assumption that a product's lifecycle ends with EOL, instead of when customers are ready for phase-out. For more information about GDCA's Proactive Legacy Management (PLM+) solutions, please visit http://www.gdca.com. Kaye can be reached at marketing@gdca.com.

Sustaining COTS: is the problem "a square peg in a round hole" or an attempt to put the right peg in the wrong hole?

Anthony E. Trovato

Introduction

This article provides a brief look at the introduction of COTS equipment into the engineering design of new equipment and its impact(s) on sustainment after deployment. It does so at the highest level, since there have been many articles written in depth that describe the issues associated with the use of COTS as components in the design of IT systems. A short discussion follows on the potential solution to the dilemma, recognizing that the issue may not be the use of COTS products but the inability of the support infrastructure to adapt to the new environment and products. Finally we can arrive at the point where we can characterize the solution as one of polishing the diamond to fit into the round hole rather than bemoaning the difficulties of trying to drill a new hole for the rough cut stone.

The Way It All Developed

A few years back, more than I want to remember, COTS was widely hailed as the solution to high development costs. Today the pendulum has shifted as we begin to see increased costs in sustainment, in selected areas of COTS products, due to the high technology turnover rate and the short life expectancy in the areas of "out of production" and "out of support." In short, COTS is no longer the darling of the product acquisition and sustainment world.

What happened to the savings? For the most part they evaporated in sustainment of these selected products. Baseline COTS products in the areas of motor vehicles, communication gear, and aviation have not been similarly impacted. Only in the areas of computers and computer operating systems have we seen this disruption.

If we look at the problem from a 35,000-foot level the disruption in the sustainment can be seen as a problem in both technology and in the processes we use for support and sustainment, rather than just a problem with the technology.

We have, while working with "built to need" end items over the years, developed a fairly rigid process for maintenance and supply support to meet our needs over a wide variety of locations and conditions. These have served us well in sustaining and supporting the troops around the world. We successfully integrated COTS into these processes for products, which had a close alignment with the development process and had stable configurations, along with a wide base of suppliers to meet the diverse user and maintainer base in a manner similar to the one we had in the military. We had alternative supply chains for selected products where instead of assigning a national stock number we relied on the vendor ID and Part number for ordering, and that seemed to work for selected products where we had a stable configuration and a parts catalogue that contained the correct information.

But something changed when we were not looking at the product line details. Providing a support solution to the use of COTS as IT components was not as simple as following the old rules. It required a different way of thinking if we wanted to sustain the product after deployment.

The technology we are using in the military, starting with the OEM and representatives, is widely used in the commercial world and finds support where needed through a variety of sources. But the COTS network extends to a multiplicity of third party vendors and suppliers. Technology changes occur outside the control of the OEM and in many cases are driven by changes in technology in the supplier base. The biggest change occurs in the total lack of control that the OEM exerts over the third party vendors and the part suppliers. When an OEM for an item of government equipment purchases COTS IT equipment they are less than 1% of most product lines from the supplier. The probability is high that the purchase was from a distributor, not the manufacturer; thus, performance specifications are used to select from a variety of parts and manufacturers to meet the current need, there is limited maintenance data available and almost no design data. By the time the COTS IT equipment is actually installed, tested and fielded the COTS components are both out of production, and most likely out of support from the initial sources. When this happens we can source the market for alternative equipment. However, that is not a continuing issue early in the program. It is not until the equipment is aged a bit that the need for spare parts arises and, by then, the market and the engineering base has long since changed.

We see the alternative support strategy in the automotive product line where the talented owners do their own maintenance and third party repair facilities, with no connection to the OEM, replace the dealer shops and spares are provided

through a variety of sources and after market manufacturers. Specialty add on items are designed by alternative sources and added without any consultation with the OEM. Should a dealer maintenance facility need one of these outside channel parts they have the flexibility to purchase it without restrictions.

In the IT world items like hard drives are provided by a variety of vendors and through a variety of suppliers, based on their individual business models. The system works because the personnel who provide the maintenance and support are not trained along the rigid product lines and the associated provisioned spare parts that the military uses. Rather, they are trained in the technology and have the freedom to select from any currently available part that meets the general technological profile of the component in the end item.

The provisioned supplier base for the military has a baseline in the development of provisioning data and technical documentation. For COTS products the end item is usually the one that is provisioned, since the government does not buy the engineering baseline of technical data packages below that level. Internal configurations are not well documented and, even if the provider does some sustainment analysis and opens the cases, the second and subsequent products may or may not have the same components.

The "Difficulty" Conundrum

So is the difficulty supporting IT COTS products with the products or the way we maintain, provision and sustain the baseline?

Albeit a bit tongue in cheek, in many ways the current furor over COTS is a bit like the dinosaurs roaming the earth and refusing to recognize climate change. Let's face it, the COTS application is here to stay and all the moaning and gnashing of teeth will not change that. Making the supportability requirement a part of the systems engineering world has done nothing to improve the process. System engineers, often, do not understand supportability; they are not trained for it and have for the most part, never experienced it in the field. Supportability in design needs to revert to a Logistics responsibility with mandatory sign off by the logistics community at all key design reviews. But before this can happen the logistician of the future will have to embark on an educational and visionary course of critical thinking, to allow them to adjust on the fly to a new way of thinking.

We are going to have to increase the reliance on the logistics community to help by changing the support process and the infrastructure in the military to successfully integrate COTS IT product support at all levels. Some, like the automotive products are fine while others that use high IT content are going to be an increasing challenge. We either accept the challenge or keep the IT products as repairable but at an OEM or third party level.

So far the discussion has focused on the continuing support of a basic product line with a static configuration. Compounding the issues of course are the desires of the operators and their supervisors in the field to have the latest and greatest equipment. Their desire to upgrade components and/or new software demands new programs all the time; making that happen will require additional engineering support - support that is often overlooked or not planned for as new programs are introduced.

Long-Term Solutions Will Demand a New Way of Thinking

Maintenance training will have to shift and allow more flexibility in the maintainer, making them more adept than the technician of today. Supply support will have to change and allow repair parts to be ordered based on the technicians' assessment of the current market availability, not the contents of a parts manual. I don't know about you but this sounds like the most radical approach I have ever heard but the current system is not working for us and must be changed to fit the product demands. It can work at the depot level with their ability to locally purchase parts outside the restraints of the current "system" but it does not work in the field. Who in Afghanistan has repair parts available immediately for their IT systems?

In summary, shifts in philosophy and practice will be needed to keep it all operating.

Biography

Mr. Trovato, currently, is the President of Full Spectrum Logistics, a supportability solutions consulting firm focused on human capital development and assisting corporations in the development of supportability solutions for their products and customers. Mr. Trovato retired from Raytheon after a 25 year career where he was member of the instructional team working with program managers in the design and delivery of professional development programs for both the RTSC staff and all of the prime divisions in Raytheon. Prior to this he was a Senior Manager at Raytheon Technical Services Company. His assignments included a multiplicity of Air Defense designs and Air Traffic Management programs where he was focused on long-term supportability solutions. He joined then Hughes Aircraft Company after a 22-year career in the US Army covering radar maintenance and operations, as well as supply support and maintenance at the Joint Staff, Corps and Unit levels.

He is a Certified Professional Logistician as designated by SOLE-The International Society of Logistics and has served as the Director of Education, Vice President of Member Services and is a Past President of the society.

He is an adjunct associate professor at the University of Portsmouth in the UK. Tony continues to teach and lecture on multiple logistics subjects and has articles published in multiple international professional journals covering various areas of logistics (including Performance Based Logistics, and Logistics Standards), Program Management, and Supply Chain Management.

The Journal of Reliability, Maintainability, & Supportability *in* Systems Engineering

Editor-in-Chief: *James Rodenkirch*
Managing Editor: *Russell A. Vacante, Ph.D.*
Production Editor: *Phillip Hess*

Office of Publication: *Post Office Box 244, Frederick, MD 21705*
ISSN 1931-681x

Instructions for Potential Authors

The Journal of Reliability, Maintainability and Supportability in Systems Engineering is an electronic publication provided under the auspices of the RMS Partnership, Inc. on a semi-annual basis. It is a refereed journal dedicated to providing an early-on, holistic perspective regarding the role that reliability, maintainability, and supportability (logistics) provide during the total life cycle of equipment and systems. All articles are reviewed by representative experts from industry, academia, and government whose primary interest is applied engineering and technology. The editorial board of the RMS Partnership has exclusive authority to publish or not publish an article submitted by one or more authors. Payment for articles by the RMS Partnership, the editors, or the staff is prohibited. Advertising in the journals is not accepted; however, advertising on the RMS Partnership web site, when appropriate, is acceptable.

All articles and accompanying material submitted to the RMS Partnership for consideration become the property of the RMS Partnership and will not be returned. The RMS Partnership reserves the rights to edit articles for clarity, style, and length. The edited copy is cleared with the author before publication. The technical merit and accuracy of the articles contained in this journal are not attributable to the RMS Partnership and should be evaluated independently by each reader.

Permission to reproduce by photocopy or other means is at the discretion of the RMS Partnership. Requests to copy a particular article are to be addressed to the Managing Editor, Russell Vacante at russv@comcast.net.

Publication Guidlines

Articles should be submitted as Microsoft Word files. Articles should be 2,000 to 3,000 words in length. Please use ONE space after periods for ease of formating for the final publication. Article photos and graphics should be submitted as individual files (not embedded into the article or all into the same file) with references provided in the article to their location. Charts and graphics should be submitted as PowerPoint files or in JPEG, TIFF, or GIF format. Photos should be submitted in JPEG, TIFF, or GIF format. All captions should be clearly labeled and all material, photos included, used from other than the original source should be provided with a release statement. All JPEG, TIFF, or GIF files must be sized to print at approximately 3 inches x 5 inches with a minimum resolution of 300 pixels per inch. Please also submit a 100-125 word author biography and a portrait if available. Contact the editor-in-chief, James Rodenkirch at rodenkirch_llc@msn.com for additional guidance.

Please submit proposed articles by November 1 for the Spring/Summer issue of the following year and May 1 for the Fall/Winter issue of the same year.