Cost Effective Systems Engineering for Companies that Build-on-Speculation

V. A. Lentz, Bruce Lerner
Otis Elevator, a UTC Company
5 Farm Springs
Farmington, Connecticut, USA
860.676.5287 / 860.676.6149
Virginia.Lentz@otis.com
Bruce.Lerner@otis.com

Abstract. The System Engineering community has long been looking for a Systems Engineering solution for 'Commercial' Systems Engineering. Our recent initiatives in the cost effective practice of System Engineering for one commercial enterprise leads us to believe that the differentiation between the two types of System Engineering might be in the business strategy. Does the company build-to-specification (BTS) or build-on-speculation (BOS)? That is the question. The answer is still evolving. However, the insight from more than 5 years of trying to apply the Mil-Aero SE Process to engineering for a commercial enterprise that enters new product development on speculation, leads us to believe there are indeed differences. Continuing to focus on making the case for the value of Mil-Aero style system engineering to the 'commercial' enterprise doesn't get us to the cost effective solution build-on-speculation businesses need. We offer some thoughts about the differences, and the allowances that need to be made. We provide some directions for organizing the SE methods to cost effectively provide the advantages of Systems Engineering to an organization that builds-on-speculation. Note: We borrowed the term 'Build on Speculation' from the construction industry. We are using it to identify a company that brings new product to market without a specific buyer or contract.

Background. In (Adifon et al. 2001), we described an activity to validate a commercial architecture. That initiative brought us to the conclusion that the architecture of a legacy product based on domain and derived functions is pretty stable across product lines and generations. The physical instantiation of the architectural elements might vary over time with technological innovation, but the architecture is stable.

- So stable that many parts can and do evolve independently of other elements in the system. Some of the parts are, for geo-political reasons, best developed in the context of a local product development. Understanding the differences and architecting a solution to cost effectively address these differences are described by (Adifon 2002).
- The functional part of the requirements statement (the What) is also stable. The how well and under what conditions varies by the size, performance and maintenance strategy.

Differing implementation functions (functions added by design decisions) drive the variations in physical solutions. These implementation functions should be maintained separately from domain and derived requirements since their stability and impact of change are different. (Lentz and Lerner 2002) provides additional description.

For a decade now we have tried to apply the SE Process shown in Figure 1. Analyze the requirements→ analyze the functions → create the functional architecture → transform that to a physical architecture→ apply all that systems analysis and control throughout, and then do the integration & testing. We get done with all of that about the same time the product gets

delivered?? What are we doing wrong??

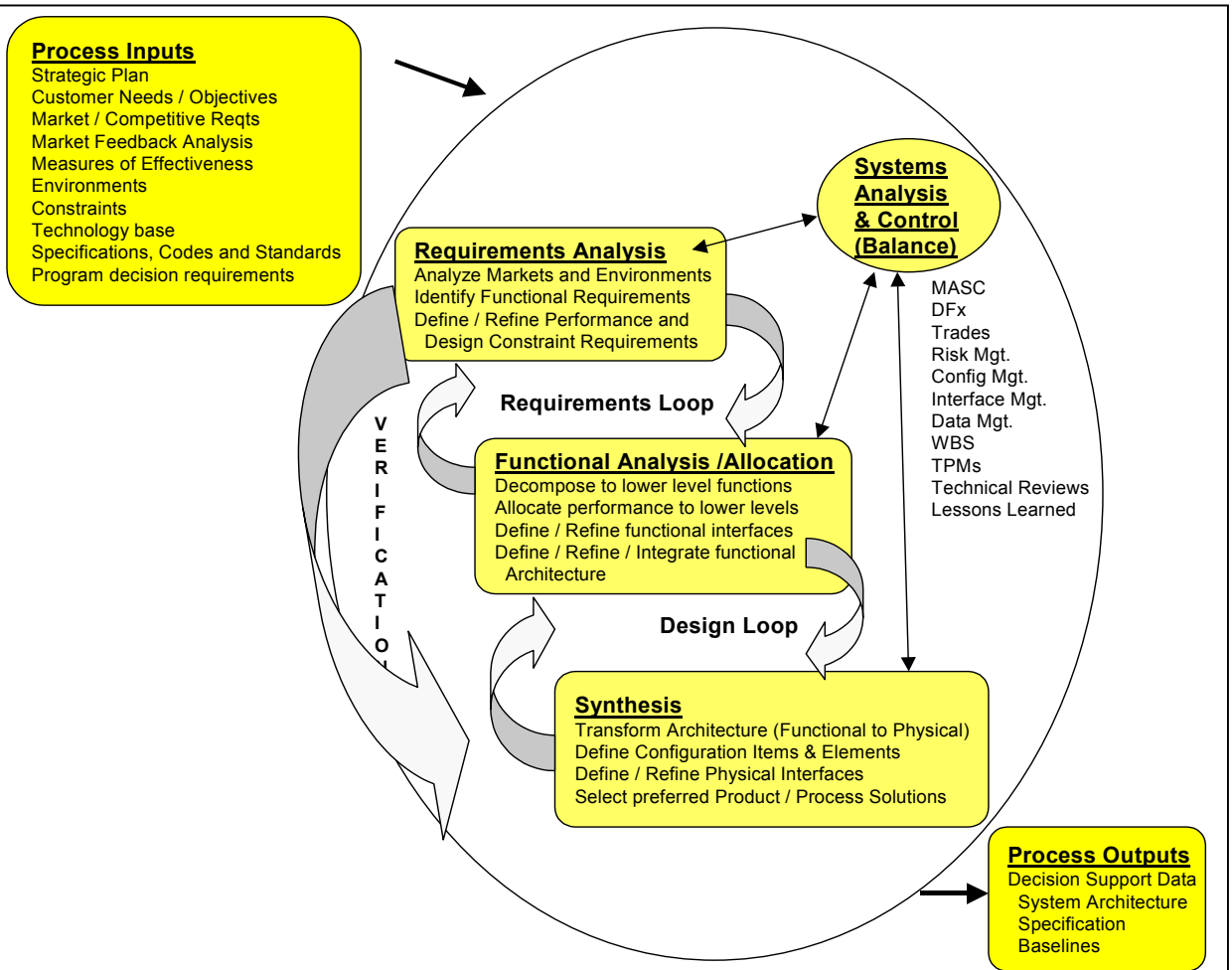

Figure 1 Traditional Mil-Aero Style SE Process

The bosses ask, "Why does it take so long to get the program defined and schedule commitments identified?" The answer 'because we are writing the requirements' produces a confused glance. The bosses know, as well as we do, what is really changing in the product: a little speed, a little weight, a new look. So what is there to 'writing the requirements? And what is different this time? What's different is that different people are assigned to write the requirements, and each has a different understanding of what a requirement is. The tool is still the same: Word - works for the top level, originating requirements. Then we get a little further along, and the component folks say they need the requirements - what is going on here? They know better than the system folk what their component must do. What they don't know is the form factor for this solution, the allocation of the performance factors or their allowed contribution to emergent properties. But we all keep this in the brain, where we also keep (in the brain) the list of lessons we learned the last time. There is a lack of documented requirements history. While the Mil-Aero community is frequently funded to create / recreate the requirements, the build-on-speculation community has a history of translating the requirements to drawings etc. Reuse is through people. See Figure 2.

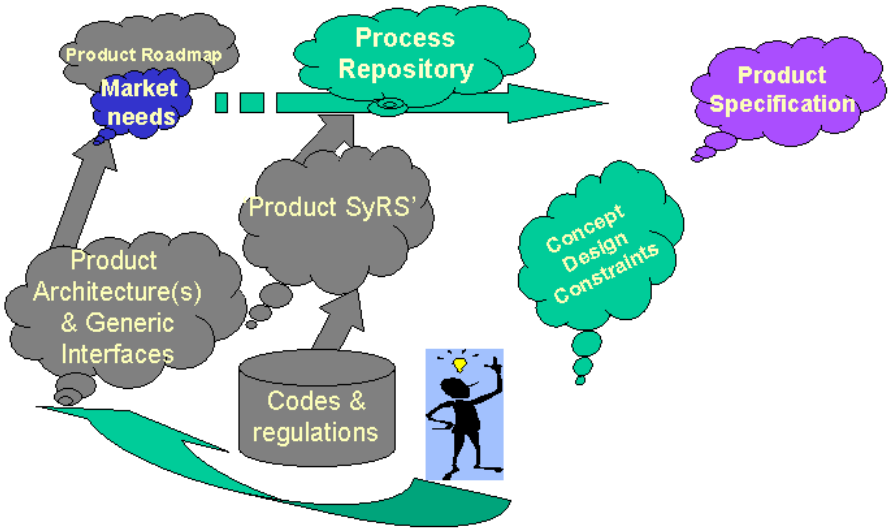

Figure 2. Reuse through People

So one starts asking the questions, what do we know, where is that knowledge, and why does it take all this time to get the requirements ready? At the end of several product cycles, we are in a good position corroborate other research that describes hurdles to documenting good requirements throughout industry. Some of our recommendations are based on observations of the gaps in the requirements in the various product cycles, how might we have identified them faster, gotten agreement on them faster, and thus reduced the rework during the system integration phase of the development effort?

There were problems beside the amount of time to gather requirements. The folks trying to fill in the system requirements specification template asked:

- What is the difference between a function and an operation? They used one or the other not both. They typically chose the functional requirements as easiest to understand.
- Does anyone really want more than 'x's in the cell? Allocating the how well part of requirements has a lot of redundancy and is difficult, but necessary.
- Why do I need a formal requirements review, it is time consuming? So it didn't happen. Instead, the requirements we discussed one on one. The aggregate time was longer, but without metrics, there was not definitive data.
- Why am I not getting design level requirements? The folks to whom requirements were allocated were looking for the build-to level of requirements they were used to writing, not the design-to level they should have expected as input. This is a problem in managing expectations.

These questions made for some interesting discussions and provided lots of insight to what isn't working.

The most difficult requirements were left to last, and we often ran out of time before they were detailed. Often these were the secondary effects driven by technology changes in the

system. While the system capabilities might be the same, and most of the physical parts remain very similar at the architectural level. The physical arrangements of the large-scale architectural elements occasionally change. These changes may drive a behavioral change **in the operational scenarios for how the system will be used or maintained. These are the drivers for the new requirements, and for any reallocation of functions to physical parts.** This is the area that will drive the largest changes in the allocation of requirements to the lower level parts of the system. Therefore understanding these differences should be the focus of any requirements activity, and should be performed first. This is the start of reducing cycle time and rework for product families. The effect of this stability on the requirements is shown in Figure 3.

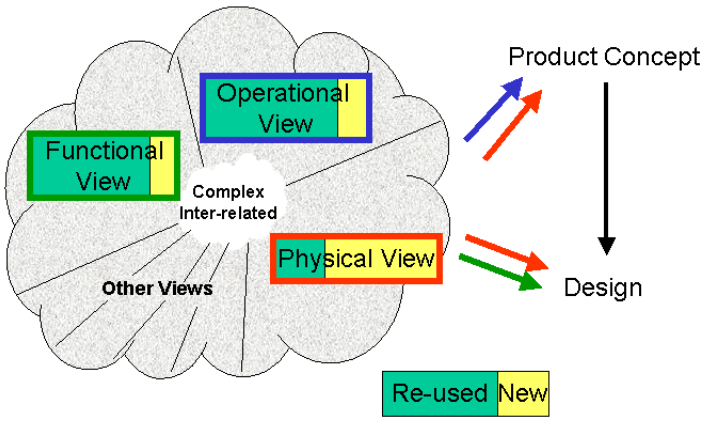

Figure 3. Requirements Stability in Product Families

Relatively infrequently, there will be an architectural change in the product family. Most of these changes are driven by new technology that allows the opportunity for significant product improvements in the areas of performance and environmental application. Sometimes technology obsolescence drives the change. These larger architectural changes will be harder to execute and may take us back to the process shown in Fig 6.

Below are some recommendations for becoming more effective (cost and cycle-time) in the application of SE for the build-on-speculation community. We recognize there are at least two types of companies in the build-on-speculation community, those who create products that are meant to be maintained over the product life (e. g. an automobiles) and those that build commodity type products that only require simple repair or replace, even under warranty (e. g. a cellular phone). Our recommendations may apply to both types of businesses; our experience is with the former.

What does the BOS enterprise need? We propose a modification to the earlier SE process diagram that still reuses the knowledge of the engineers, but does not require steps that are unnecessary, simply because that is the way it is done in the build to specification community. The BOS community has a different environment for the development process. Based on the originating requirements,

1. We first need to understand the changes in the operational concept for use of the product, and for maintenance of the product. Changes in the way the system is used, will make a difference to the requirements, and perhaps to the allocation of functions among the parts of the product.
2. We then need to understand if product functions are changing.
3. Last, we must assess whether the insertion of a new technology might cause a reallocation of functions.

This sequence is where we need to focus. Once the changes to operational behavior and functions are understood, we can:

1. Add the performance data that gives us the requirements,
2. Check the sources of non-functional requirements for new items,
3. Readily update the previous set of requirements for this new product.

The requirements, once documented and reviewed, can be reused, whether they are stored in Word or a requirements tool. Figure 4 is a graphic of this minimalist process.

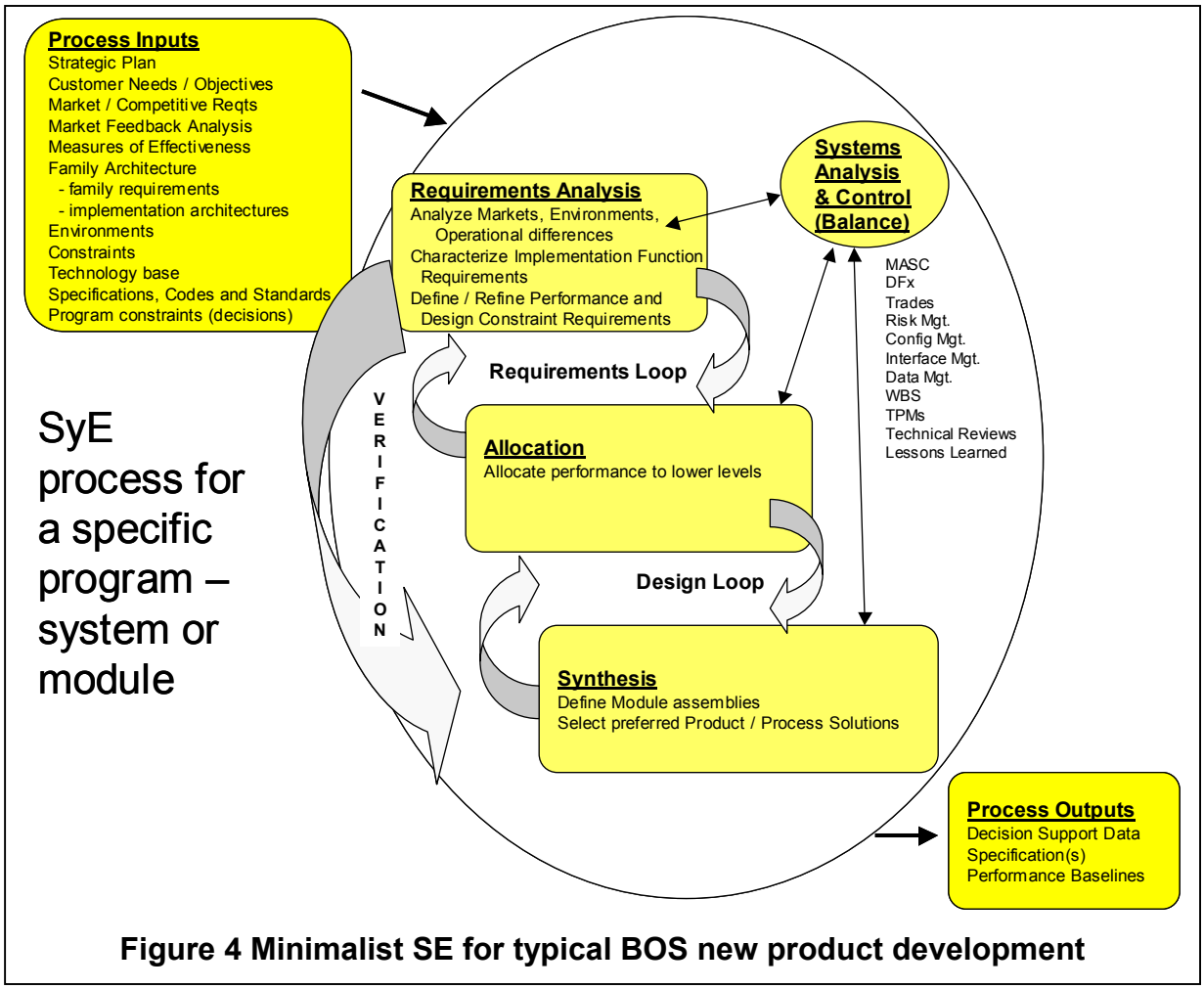

Figure 4 Minimalist SE for typical BOS new product development

Since this is the minimalist approach is it sufficient to tweak the requirements? Is that enough to build a system? No. Only when the 'Family Architecture' in small print in the process inputs box is available, do we have a total solution that supports thinking of the changes as "only

minor requirements changes". All "family" artifacts that support higher-level decisions than the specific one(s) being made for this development can be re-used decreasing development effort and validating the architecture. Other artifacts of the prior design processes are key to making the minimalist approach effective include:

- Similar FMEA analysis & reports,
- Risk analysis (what residual risks remain from the prior product)?
- Test plans & procedures ('higher' level ones do not change)

Additional details for this SE process and the reuse artifacts are shown in Figure 5.

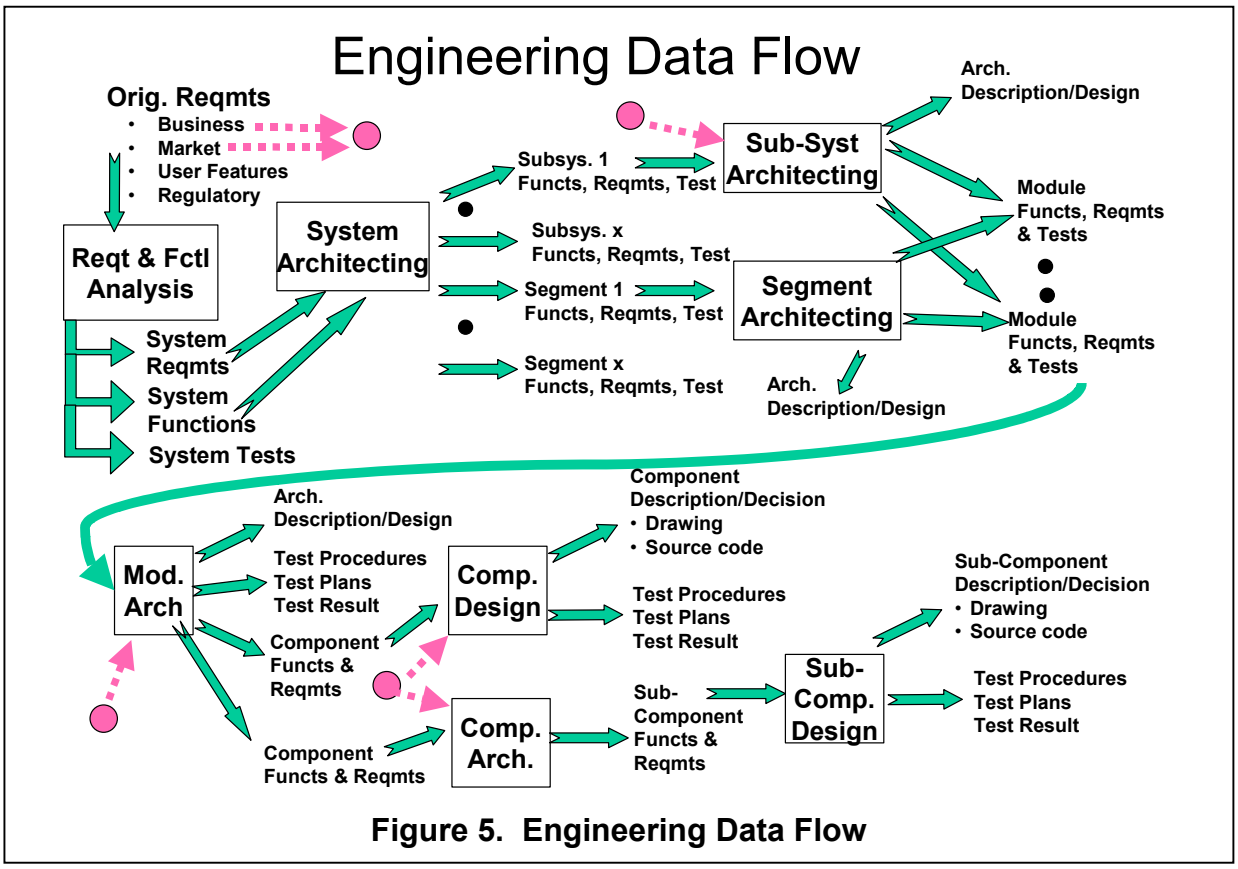

Figure 5. Engineering Data Flow

What are the risks in this minimalist approach? There is always the risk that you will miss something. But you have that risk even with the most comprehensive application of the SE process. While we often debate whether SE is an art or a science, it has attributes of both and there is a value to experience on the part of the lead engineer and the program manger.

There are probably more risks in the independent evolution of the parts, planning for subsequent integration in a system. The disciplined approach to modular systems, and rigorous change management for interfaces are key to making this minimalist approach successful.

What conditions drive a BOS enterprise to use a more traditional process?

The conditions under which you might use the more traditional process include:

- Insertion of technology that affects more than one of the larger elements.
- Planned change in a majority of the elements where the unruly emergent properties are likely to be a risk.

One example is the transition from microprocessor-based implementation of the control features to a more complex computer based solution. Since the controls for a product typically have interfaces with many other parts of the system, the complexity of any changes in the control architecture, might have a major impact on the total system.

Another major change by companies today is the move to 'built in' diagnostics and the smart product that calls the service center when something goes wrong (remember the IBM 4300?). Another example is the 'check engine light'. In the early days, the hacker quickly found out how to simultaneously press 5-10 buttons on the dash and get a diagnostic code that could then be communicated to the service person. Today household equipment can call the 'shop' and report the need for maintenance. This change adds or modifies operational scenarios, which affects functions in many parts of the product: the function of self-diagnosis & reporting. In all cases this will affect derived and implementation functions, and require changes in the parts, the product, and the service infrastructure as well. This change and the requirements and functional allocation that are associated with it need to be analyzed, designed, and implemented from the enterprise system perspective, since the maintenance infrastructure becomes a part of the system! New scenarios have to be developed for using the information to schedule customer contact or visits from a mechanic. The more extensive process is shown in Figure 6.

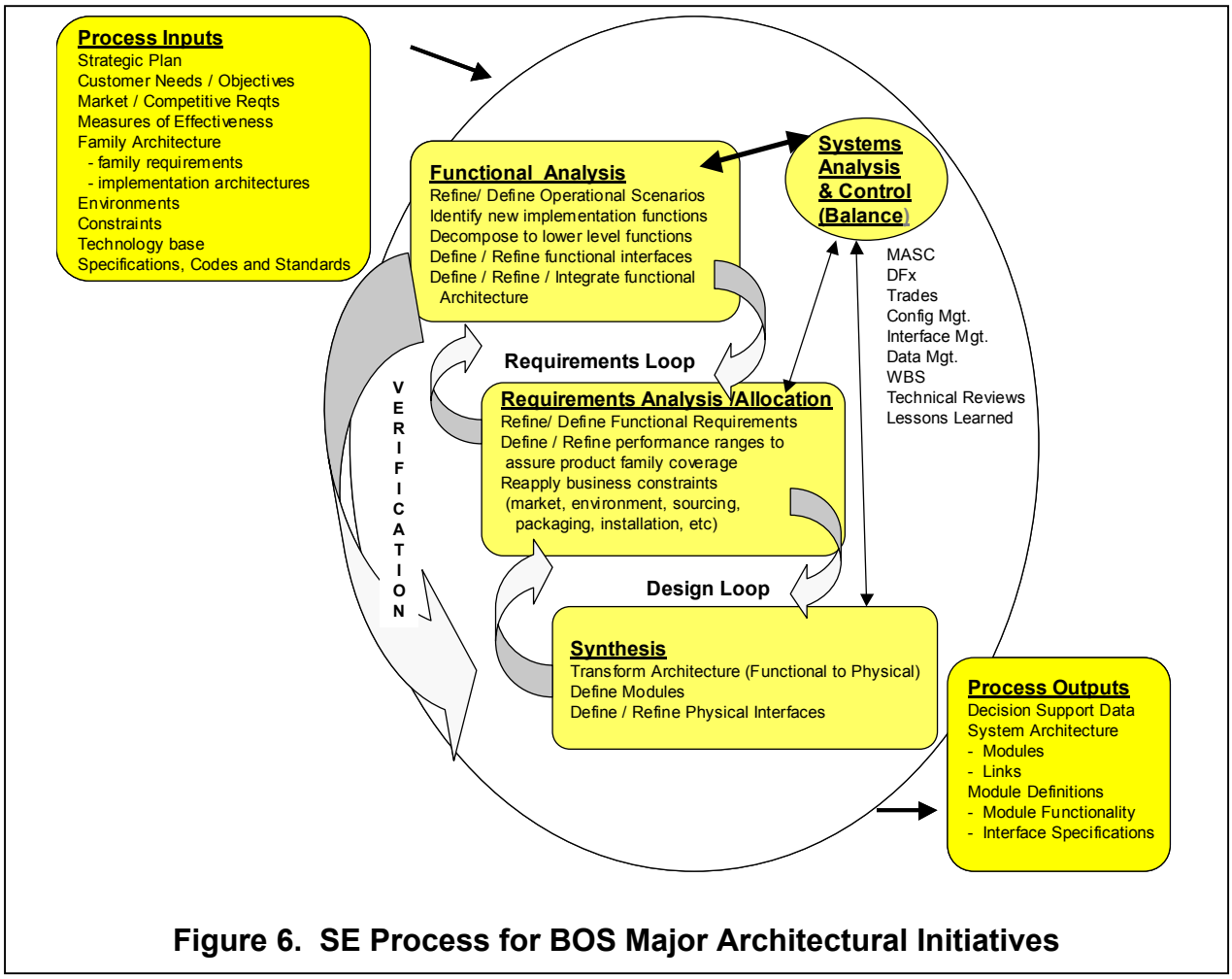

Figure 6. SE Process for BOS Major Architectural Initiatives

Notice that we now start with the functional analysis loop rather than the requirements analysis block. This is because, even with a radically new instantiation of a legacy product, the main and derived system requirements and functions are stable. The FRAT process and procedures are well documented by (Mar 1992, Mar and Morias 2002).

What is the problem with reuse through people?

Reuse through people requires the following conditions:

- Small team guided by strong Chief Engineer / Village wise man (Lentz 2000)
- Stable, geographically localized product development team, always improving the product incrementally

Today, with the trend to globalization, companies are beginning to recognize that exporting the company culture, process and design guidance, does not export the relationships that make reuse through people happen. One transition enabler is standard process & product definitions that can help set expectations and provide consistency while the relationships are built. This is particularly important if the product components are designed in a number of locations around the globe and built in factories that might be attached to a design location, but more likely are located in yet another part of the world. The cycle time and iterations needed for system and part requirements and design reviews, are exacerbated by the distribution of the engineering and manufacturing. While there are advantages to 24x7 product engineering, it needs more coordination and infrastructure and patience on the part of the design engineers trying to find overlapping working hours for team meetings.

What does it take to make cost effective SE happen for the BOS enterprise?

Make an Investment. Invest in understanding & validating the product architecture. This will require an investment in the analysis of the functions, and the separation of the domain, derived & implementation functions (Lentz and Lerner 2002). Invest in understanding the scenarios that describe how the product is used, and maintained, (and recycled?). Invest in capturing a reusable set of requirements, organized for the things that are known to change from one product development effort to another. Invest in the modeling & analysis to identify the best-fit allocation of requirements that manage the result of emergent properties. Invest in a tool for capturing the requirements, so reports can be generated for an individual product, rather than requiring long requirements writing activity. And then remember, this is the repository for the typical product improvement or cost reduction product development initiative. Revolutionary product design initiatives may cause us to reevaluate much of our previous thinking. Folks must be trained in the fundamental SE methods and be experienced in applying the methods as both Requirements→ Functions → Answer → Test and Functions → Requirements → Answer → Test, They need to recognize that both the RFAT and the FRAT are correct when applied to suitable design initiatives. The decision of which to use is based on the type of program, and the baseline information available at the start of the activity. With this infrastructure in place, the time from concept to ‘requirements-defined-and-allocated’ will be commensurate with the changes in the product and the expectations of management.

When introduced to the methods of System Engineering, many recognize the potential advantages. In the build on speculation business a subsequent question is: ‘how do I justify this

cost as adding value to my product?' One measure is the reductions in cycle time provided by reuse of the product parts, the product development artifacts. Planning the work for a new initiative by using the differences from one generation or product line to another decreases program risk. Other metrics are decreased duplication of effort, increased reliability, increased quality and decreased testing. Reuse in this environment does not eliminate the need for a strong engineering team and leadership. It does however free them to focus on the new and creative work, and not on the regeneration of a series of artifacts, whether drawings, software code, or documents that already exist.

REFERENCES

Adifon, Leandre, Lentz, V. A., Lerner, Bruce and Marvin, Daryl, "Validating a Commercial Product Architecture." *Proceedings of the 11th Annual International Symposium of INCOSE*, Melbourne, Australia, 2001.

Adifon, Leandre, 'The Global Design Challenge', UTC MIT Project Report 2002.

Lentz, V. A. "Five Realities for Systems Engineering in Commercial Enterprises" *Proceedings of the 10th Annual International Symposium of INCOSE*, Minneapolis, MN, USA, 2000)

Lentz, V. A. and Lerner, Bruce, "Functional Analysis for Existing Products: a Detailed Procedure." *Proceedings of the 12th Annual International Symposium of INCOSE*, Las Vegas, NV, USA, 2002.

Mar, Brian W., "Back to Basics", *Proceedings of NCOSE*, 1992.

Mar, Brian W. and Morias, Bernard G., "Maximizing the Systems Aspect of Systems Engineering" Professional Tutorial delivered at the INCOSE 2002 Annual International Symposium.

BIOGRAPHIES

V. A. Lentz is responsible for Program Management for New Product Development for Otis Elevator Company. She has been with United Technologies since 1996. Her favorite topic is the use of Systems Engineering in commercial enterprises. Prior to UTC, she spent 30 years at IBM Federal Systems, LORAL, Lockheed Martin building large, unprecedented computer-based systems such as Global Positioning System Control Segment. She was also responsible for Systems Engineering Technology, Process and Training. Ms. Lentz was President of INCOSE in 1996, a recipient of the Founders Award in 1999, and represents UTC on the INCOSE Corporate Advisory Board.

Bruce Lerner is a Principal Systems Engineer with Otis Elevator, a United Technologies Company. He has been at Otis since 1986, managing Software development for embedded communications and control, leading Software Process Improvement activities and spearheading the deployment of Systems Engineering. Mr. Lerner is a member of INCOSE (Constitution Chapter Treasurer) and the IEEE Computer Society.