Using Value Engineering to Propel Cyber-Physical Systems Acquisition

Alfred R. Schenker, ars@sei.cmu.edu, (412) 389-8484

Nickolas H. Guertin, nhguertin@sei.cmu.edu, (703) 350-1061

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM21-0335

Abstract

The DoD's approach to building and deploying software-intensive systems is constantly under revision. In parallel, the tools and methods to model and test architecture representations of candidate products have also evolved. We will investigate the adaptation of Value Engineering (VE) methods into the acquisition of software-intensive weapon systems where the candidate product architecture can be modeled and used to guide implementations throughout the lifecycle. Aligning model-based engineering with VE will accelerate innovation in the development process (Vogt, 2019). When used with a process performance baseline, this method can establish a comparison framework for cost/benefit analyses of alternative approaches.

VE is used to evaluate different approaches or processes to improve the end product cost (including future, unaccounted for, cost overruns), performance or quality. We believe that VE can be applied to the development of software intensive weapons systems as well. To date, the cost to perform system modeling and virtual integration in early product development stages has been traded off against showing early results. The benefits of virtual integration are significant, especially in the long-term value of being able to evaluate system upgrades and changes both while in development and after the system transitions to sustainment.

Introduction

The intent of this paper is to stimulate out-of-the-box thinking about what we can do to improve the status quo of acquiring cyber-physical systems (CPS). We will explore how to incorporate VE principles as a means to stimulate new ways of developing and sustaining the embedded computing resources of a complex CPS. VE is a structured, multi-disciplinary approach to improve contract performance (INCOSE, 2015). In short, a traditional VE approach would have the U.S. government (USG) and the Department of Defense (DoD) provide additional funds to a contractor as an investment in their product development process. The investment is recouped over the life of the contract in one of several ways, e.g., reduced manufacturing cost, higher quality, etc.

When we use the term "VE principles," we mean:

1. to use VE as a means to motivate changes to the workflow of organizations that develop computing platforms for CPS
2. to reward innovations that reduce the acquisition risk of embedded computing systems in DoD CPS
3. that we need to reward innovations based on a new way to count value that identifies and mitigates "showstopper" defects early in the lifecycle as a leading indicator of traditional VE values, e.g., cost, schedule

The basic structure of this paper is as follows:

- We begin by characterizing the status quo for the acquisition of these types of embedded computing resources, which seems to be in need of improvement.
- We then make the point that if we continue to do things the same way, we should expect the same (or similar) results.
- We introduce Architecture Centric Virtual Integration (ACVI), a proven means for identifying issues in embedded computing resources, as a way to try a different approach.

- We emphasize the need to reward the "right kinds of behavior" to try to get the ACVI process integrated into the contractor's workflow—to reward honest effort as opposed to checking the box.
- We describe how important it is for the contractor to establish a measurement program focused on early identification (and resolution) of "showstopper" issues.
- We construct a notional means of financing the costs associated with the additional modeling effort (the value).
- We point out that we do not know enough about the contractor's operations to be prescriptive, but we have general guidance.
- We describe the need to create an environment that allows the contractors to fail without penalty (current status quo), yet rewards the right behavior.
- We conclude by providing some examples of how contracts could be written to provide the right kinds of incentives, and some ideas for how contractors could integrate ACVI into their workflow.
- Finally, we make some recommendations for acquirers and contractors.

The Status Quo

In the development of software-intensive systems, more specifically CPS employing embedded computing resources, the authors are not aware of the construct of VE ever having been used. One limitation imposed by a VE approach is the inherent lack of accuracy in software cost estimation on which to make VE trades. A key element of VE is to recoup an investment, and it is hard for a contractor to commit to a reduction in cost if they can't predict the cost with confidence.

Other formal approaches, such as those found in the Society of American Value Engineers (SAVE) or the International Council of Systems Engineers (INCOSE) have a standard set of activities (SAVE, 2021) We will not attempt to map those process steps precisely to this study, but to reflect on the strategy of capturing value in making product and process improvements as a mechanism for managing the lifecycle of a software-intensive system (DIB, 2019). These are VE principles, not the actual wording of the VE contract language.

Unlike hardware systems, software systems are produced as they are designed and are continuously updated and changed over time. Even products that do not require functional change have to be updated to manage vulnerabilities in the products that make up the overall software system, or through managing technology obsolescence. As such, the context of managing the classic hardware lifecycle cost does not precisely map to software-intensive systems. What is worth considering are the major functional elements of VE, e.g., establishing a baseline, developing a plan for the process improvement, and monitoring the process to identify the opportunity for incentive as they apply to embedded computing systems.

Model Based Engineering (MBE) and digital analysis are well-developed practices for physical products. For example, decades-old technology exists for modeling and analyzing physical properties (thermal, mass, structural, dynamic, electrical) and electro-magnetic interference. These models and analyses have demonstrated their value many times over and, to some extent, these techniques are widely accessible and used by all. However, rigorous equivalent application of MBE for cyber-physical/embedded systems with integrated computing has not achieved widespread adoption. Abstracting and modeling inherently abstract designs like embedded computing systems can lead to over-application of functional or system decomposition. In order to advance the state of the practice for modeling and analysis of embedded computing systems, there must be more process rigor employed when developing and using models and analyses. Models of the system components that interact with each other can virtually verify integration issues and intra/inter-product interoperability properties. To establish this practice will require both new requirements and new incentives to drive innovative, alternative behaviors.

One author was recently a judge for a middle/high school regional science fair for the category of engineering. Several students used a combination of physical modeling and additive manufacturing to fabricate components for their entries or to do virtual prototypes to model their designs before full implementation.

Let's think about the introduction of microwave cooking technology. When first introduced, homeowners did not fully know how to use it nor what it would be best applied to do, but the potential was clear. To take the union of VE and MBE all the way through the development cycle, there is a need to develop methods and techniques that take advantage of this potential. To use an architecture model as a pivotal piece of design that is established and nurtured throughout the lifecycle of a product has its uses, but it is not going to apply to everything all at once, at least not at first (Clements, 2002). Professionals who have seen development fads come and go are reasonably skeptical at the beginning. Such was also the case with microwave cooking technology. When microwaves were first introduced in the 1970s there were people that thought that placing a microwave radiation emitter in their kitchen would cause harm. Juxtaposed to those concerns were the staunch advocates that thought it could be used for everything. Fifty years later, nearly everyone has a microwave oven in the home; we know how to use it, and we are comfortable with it. Who could have predicted then that most of America today would end up using a pre-sealed pouch to make popcorn? Equally important to adopting a new approach is that we also learn when not to use a new tool. Many of us have subsequently found, for example, that a microwave is not optimal for reheating pizza.

Effective use of tools, such as MBE, architecture analysis, and digital twins will reduce acquisition risk and are not minor adjustments to work flows, but are fundamental changes to what we do and how we work.

Changing the Results

Over the past 20-plus years, few would argue that the track record of the USG's acquisition of CPS has met expectations. For example, on page 16 of the recent GAO report (GAO-21-226) on the F-35, "In 2020, the F-35 program resolved 33 of the deficiencies it had identified in developmental and operational testing but it continues to find more. Of the 872 open deficiencies, the program characterizes 11 as category 1 (critical in nature and could jeopardize safety, security, or another requirement), and 861 as category 2 (could impede or constrain successful mission accomplishment). This represents two more open category 1 deficiencies than we reported in May 2020. According to DOT&E officials, additional new discoveries are due to quality problems with the F-35 software, resulting in a high rate of deficiency discoveries during operational testing and by pilots in the field. According to program officials, seven of these open category 1 deficiencies will be resolved prior to the completion of operational testing. Four will not be addressed until the third quarter of 2021." The report goes on to say, "In recent years, program officials did not identify nearly a quarter of all defects until they were already delivered to test aircraft. **Ideally, according to the program office, the contractor would identify defects in the software lab or before the software is fielded to the developmental test aircraft** (emphasis added). However, a November 2020 analysis conducted by a third-party consulting firm on behalf of the program office found that between December 2017 and September 2020, 656 software defects (or 23 percent of all software defects) were identified after the software was delivered to the test aircraft."

As illustrated above, the heart of the problem of acquiring CPS are issues identified late in the program lifecycle (GAO, 2019). These issues arise when physical devices are provided for integration, and the embedded computing systems do not behave as expected (AVSI, 2019). In addition, as a result of the late changes needed to make the system work, efforts to attain certification (safety, airworthiness, security, etc.) can be delayed. These delays increase the overall program schedule, which most often is tied to program budget. The objective is to avoid being on the long list of acquisition programs that have significantly exceeded schedules and budgets, or have even failed to deliver, such as Future Combat System, Ground Combat Vehicle, Advanced Amphibious Assault Vehicle, and Comanche. Other Services have had their impact felt in the dramatic reduction of the number of items produced, like F-22 and DDG-1000 (Rodriquez, 2014). The impact of delays, overruns, and cancellations hurt the taxpayer with higher costs to maintain the national defense.

Figure 1 is a graphic that illustrates the issue that this paper seeks to address. The x- and y-axes represent the schedule and cost budgets of the program. The blue area represents the expected capability of the platform as it evolves. In CPS acquisitions it is typically the case that we get to the end of the schedule, having spent all the money, but the system does not work as expected. Additional funding and schedule are needed to complete the platform. One leading cause for this is that the program is funded assuming "normal" integration issues will be found. As the program progresses and gets to the point where the program should be finished, unexpected issues cause delays, and the acquirer must expend more funds in order to finish the program. The program may also choose to sacrifice planned capability, or attempt to do a balance of both an increase cost and a reduction in delivered capability. The value we seek to capture, e.g., the reduction or possible elimination of unforeseen future costs, is illustrated with the bullseye. The extra funding is a direct hit to the DoD budget, and impacts other programs, as this additional funding is unplanned. The extra schedule is a direct hit to the warfighter, as the delay impacts when the platform can actually be deployed.

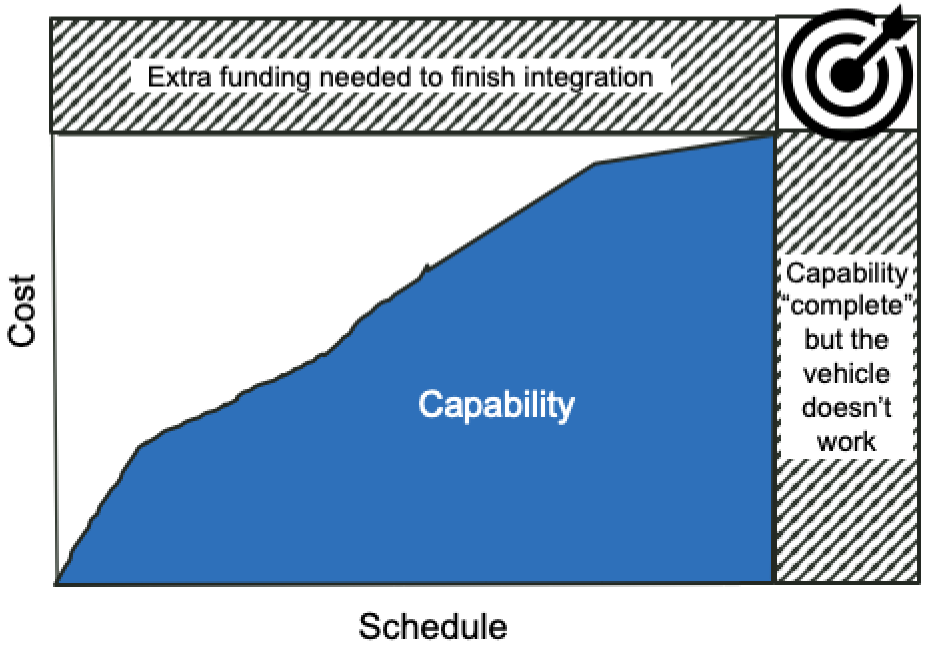
Extra funding needed to finish integration
Cost
Capability
Capability "complete" but the vehicle doesn't work
Schedule

Figure 1
Identifying the Opportunity, the Value Bullseye

However, it does not have to be that way. The accepted colloquial definition for a risk is "an issue that has not yet happened." Integration issues are going to happen, so at this point they can be characterized as risks that are highly likely to occur, more specifically as embedded computing system integration risks. Software system integration issues are the largest single factor on recent new aircraft systems (DSB, 2018). Also, 50% of the system development effort goes to software rework and 80% of the software issues in system development are found in integration (Hansson, 2018). This means that status quo approaches do not effectively find these issues, leaving programs exposed to these risks if they do not address them overtly and early in the life cycle. These embedded computing system integration risks apply to the acquisition of all CPS programs. The acquirer can choose how to mitigate these risks. One way that has potential is the Architecture Centric Virtual Integration Process (ACVIP).

ACVIP: A Different Approach

ACVIP focuses on analysis of embedded software computing systems, using architecture artifacts instead of a fully implemented design. ACVIP provides methods and tools to address product development and integration challenges where computing execution runtime sensitivity, safety, and cybersecurity are critical. A major source of system development schedule growth in these systems is the increasing complexity of interactions between the application software, the embedded computer system, and interfaces with physical systems. ACVIP provides a *virtual* integration environment that enables the detection of defects not typically found until much later. This is frequently accomplished by employing the Architecture Analysis and Design Language (AADL), continuous verification throughout the entire development lifecycle, and a single model that supports the analysis domains (e.g., safety, security, resources) that impact architecture and integration.

ACVIP begins with requirements verification and proceeds through actual system integration with the goal of identifying defects as early as possible. By incorporating virtual integration throughout the lifecycle, system and software engineers are able to prevent these defects from propagating (including side effects of change), and removal reduces future defect insertions. The early discovery of defects improves quality, reduces schedule (and cost) for the development and qualification of these systems, and ultimately enables fielding capabilities earlier. Even for defects that escape ACVIP analysis in the design phase, when physical integration issues arise, the design specification crafted in AADL speeds root cause analysis and predicts side effects of resolutions, enabling efficient resolution rather than multiple "test and fix" cycles.

The use of ACVIP has shown utility and benefit from previous U.S. DoD and non-DoD projects. It is certain that the additional effort to build the product models (to enable ACVIP) will be higher than to not make that investment. However, since as much as 80% of the defects are not found until physical integration when using traditional lethods, early discovery and remediation will result in signficantly lower overall cost as a result of using ACVIP. Additionally, establishing and maintaining the virtual environment that is required to use ACVIP will reduce the cost of future block upgrades to the platforms.

Reducing risk likelihood is generally achieved through a set of engineering and management activities in a step-wise progression that does not reduce the risk consequence. By incorporating an ACVIP approach, the program also reduces the consequence of problems that can impact cost to complete or reduction in delivered functionality. Figure 2 depicts how virtual integration and related analyses could impact both the likelihood of integration risk and late-stage discovery. The governance and workflow integration of ACVIP also reduces the consequence when those risks are realized by finding them early and working through design and integration problems well before the design is finalized.

ACVIP works for capability changes injected into the program throughout the lifecycle as well. Once a product is matured through initial fielding or Minimum Viable Capability Release, ACVIP continues to positively impact the program—as long as the models and analyses are kept up to date with the fielded implementation. We show this transition from metaphysical certainty and dire impact (i.e. it happens to nearly every program) to moderate likelihood and manageable consequence. It would be irresponsible to assert low likelihood or consequence of these risks until these practices have been put into place and quantitative analysis on affected programs can be performed.

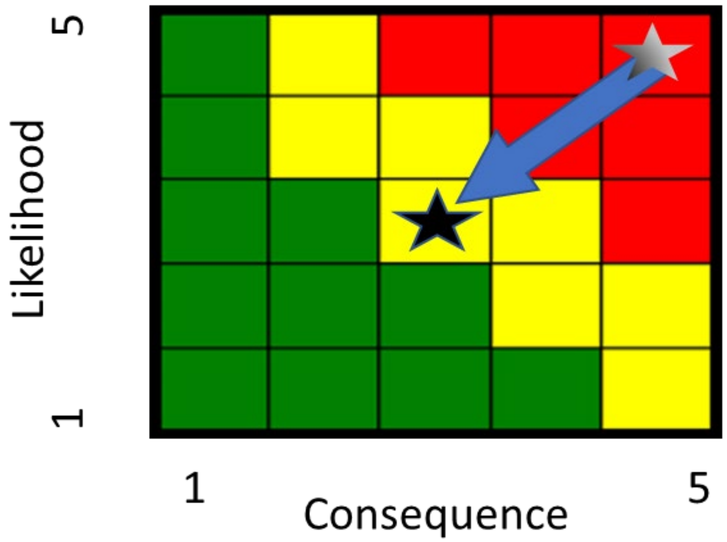

Figure 2
Value Engineering Goals for ACVIP

ACVIP augments and enhances model-based systems engineering (MBSE) along with SysML as a systems engineering practice. While SysML applies broadly to the entire system, ACVIP applies primarily to the runtime analysis of embedded computing systems. ACVIP uses model entities described in SysML together with model components expressed in AADL, and augments them with the structure and additional properties that result in the specification of the software and hardware system. This formal representation of the architecture allows a precise and repeatable analysis capability that will verify a set of system quality attributes expressed as requirements, especially those about an implemented software deployed onto one or many computing platforms. The analysis capability spans a number of crosscutting system qualities that include safety, reliability, cyber security, resources, and performance in terms of signal latency and computing resource utilization, among others. The associated MBSE SysML models, along with other models that have been analyzed and updated because of performing ACVIP analyses will have higher quality and will integrate with fewer issues or defects.

One of the ways that ACVIP would benefit a systems engineer include the representation and analysis of interface descriptions that trace down to the run-time level. The ACVIP analysis helps the systems engineer to understand the dependencies between systems and subsystems. Failing to identify interfaces, or failing to precisely specify their properties, is a common reason for products to fail and not to meet stakeholder expectations. Missing or incorrectly defined interfaces are a major cause of cost overruns and product failures (SAVI, 2015).

Similar issues caused the stability control system of the Lexus SUV to not slow the vehicle when it entered a turn at a high speed. In another example, the explosion of the Ariane 5 rocket’s first test occurred because of a malfunction in the control software. This is an example of a failure to represent interfaces consistently between two pieces of software. The cause was a data conversion / representation issue among two software components. This resulted in the system detecting an out-of-bounds pressure on the rocket, causing the rocket to self-destruct. (Lions, J. 1996)

These are examples of interface defect types that can be detected with ACVIP. Analysis of the architecture model identifies interface inconsistencies that include: data type, data format, data update, data number, data range, and data units. These types of errors are detected continuously beginning with the construction of the initial model, throughout the entire model as pieces become integrated and ensures consistency at software and hardware integration time.

Where's the Value?

Figure 1 showed a bullseye, representing the time and money that we put on contract at the end of the project to finish the project. For the F-35 program, there were billions of dollars of cost, and years in schedule borne by the Government (GAO-21-226). Of course, we cannot ascribe all of the cost overruns and schedule delays to embedded computing system issues. There will also be late changes to the scope (or late changes to the design), to the components, and possibly a reallocation of functionality.

However, significant value to the acquisition stakeholder will come from the early identification and elimination of the "showstoppers" that arise late in lifecycle. Showstoppers are best characterized as issues that force a rewrite of the software, most likely due to a safety concern or a certification issue. Figure 3 summarizes issues from several safety critical systems. "Despite best build-then-test practices, system-level faults due to software have increasingly dominated the rework effort for faults discovered during system integration and acceptance testing. Several studies of safety-critical systems show that 70% of errors in embedded safety-critical software are introduced in the requirements (35%) and architecture design phases (35%). At the same time, 80% of all errors are not discovered until system integration or later. The rework effort to correct a problem in later phases can be as high as 300-1,000 times the cost of in-phase correction. Therefore, it is desirable to discover such problems earlier in the lifecycle and increase the chance of tests passing the first time around." (Feiler et al., 2013)

Our claim is that the extra time and money (the bullseye) we spend at the end of the contract to "finish" the system should be planned for and managed as part of the acquisition risk. Instead of doing emergency appropriations for money in future years, to the detriment of other programs, program managers should avoid risk realization through virtual integration. "We believe that fully embracing an ACVIP approach is effective risk mitigation for the inevitable unknowns that surface late in all cyber-physical system development projects" (Boydston, 2019).

Find and Resolve Showstoppers Early

The acquisition community in general, and programs that build highly interoperable CPSs in particular, should acknowledge the near-certainty of schedule and cost overruns and build in reward mechanisms that incentivize behavior that avoids them. If, for example, we are able to identify and address 90% of the showstoppers during system development, allowing us to achieve the baseline schedule with no increase in funding or compromise in capability, we should reward that behavior, significantly. For example, if we were to avoid a $100 million cost overrun, we should be able to perform a risk-sharing initiative that would harvest some smaller portion of that (e.g. $25 million) as an incentive. Cheap at twice the price.

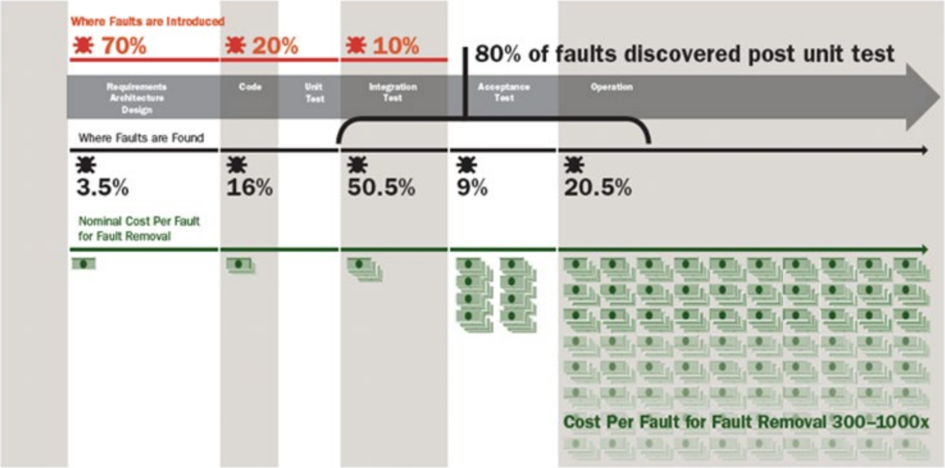

Figure 3
Cost Impact of Late-Stage Discovery (Feiler, 2013).

Figure 3 shows an analysis of the status quo of CPS development (Feiler, 2013). As much as 80% of issues are detected after unit test. It is hard to argue that this is the kind of result that we should be satisfied with when it comes to identifying and resolving problems early in lifecycle (when they are affordable to fix). As a result, we wind up paying many times more than what was estimated to resolve these issues. There is little direct financial incentive for the contractor to change associated behaviors, because the government will make up the difference in total cost to get the project completed. A cynic would argue that the contractor is actually motivated to not change their development process. A comparison could be made to the acceptable quality level of an automobile manufactured in the U.S. up to the 1980s. Consumers would routinely accept substandard product quality, and get their identified issues repaired after the initial purchase and hopefully during the warranty period. There was no incentive for the U.S. automotive manufacturer to update their process, and there was no public outcry for better cars. We just accepted the status quo. By contrast, Japanese companies during that time showed the world how to manufacture automobiles without quality issues and were rewarded with large changes in market share. This forced a change to everything in the U.S. automotive manufacturing industry. The expectation of the consumer been changed. In order to get that kind of change in the environment for CPS, we need a different kind of forcing function through DoD policy changes to improve engineering processes and to insist on better outcomes through VE. A related change in the Defense Industrial Base is needed to encourage risk taking as well as rewarding attempts to improve.

So, in the context of VE for ACVIP, the U.S. government can incentivize the contractors to adopt ACVIP as part of their process (i.e., as a means to identify showstopper issues earlier in lifecycle). If the modeling and analysis is implemented in the right way, the contractor will avoid a significant number of showstoppers late in the lifecycle, which can be subject to risk-sharing incentives that benefit them and the Government (Wrubel and Gross, 2015). This translates into reduction of acquisition risk. This will raise the bar for CPS developers.

Reward the "Right Kinds" of Behavior

We believe that contractors will not be motivated to change their process without incentive. We believe that there is plenty of opportunity to improve the status quo. This is what brings us back to the original concept of VE. We need to stimulate innovation within the contractor community by incentivizing changes to their workflow that take advantage of these novel methods. We want the incentive to reward the right behavior, that is, if the contractor is successful in avoiding showstoppers late in the lifecycle, there should be a significant reward. We are not attempting to prescribe anything other than guiding principles, as we do not yet know enough to be prescriptive. We do know that we are talking about potentially hundreds of millions of dollars, and a significant change to the development process. There are many ways that the rewards and incentives could be realized, which is a topic for future research.

Bridging the Systems vs. Software Divide

A long-standing hallmark in DoD system development is the mismatch between systems development activities and software development activities (Sheard et al, 2019). While we do not think that adopting ACVIP will break the cultural clash, we do think that it is a step in the right direction. MBSE is an established discipline. Potentially critical flaws in the system design can be found much earlier using a similar model-based approach for the design and implementation of the embedded computing system, allowing much more time for the evaluation of alternative approaches.

While the adoption of ACVIP by itself will not force an improved collaboration, the development of an architecture-centric virtual integration model will lead to the potential for a common ground for systems and software engineers to reach agreement. We refer to this as the "authoritative source of truth (ASOT)." Assumptions, constraints, and dependencies are all things that need to be represented and incorporated into the model, which will be fed by both the systems engineers and the software engineers. The ASOT will serve as a clearing house for the design issues. As the model evolves, with increasing fidelity over time, the model will serve as a fundamental resource for both groups, providing an environment that facilitates the rapid resolution of the showstoppers.

Contracting and Managing Incentives for ACVIP

One of the most effective communication tools the USG has with industry is through its requests for information (RFI) and requests for proposals (RFP) and the resulting contracts. The structure of the requirements, evaluation criteria, and incentives sends a strong message about what is important enough to be the difference between winning and losing. One challenge of managing a program is to balance the drive to perform as soon as possible with the hard decisions of increasing near-term costs for long-term gains. For the purpose of this discussion, there will be a focus on cost-type contracts, which are best suited for development of cyber-physical products where there is an inherent lack of precision on which the outcomes can be predicted (Kendall, 2013).

The projected future cost of any contract is analyzed up-front through mechanisms like the Government's Independent Cost Estimate (ICE) that is developed prior to releasing a RFP. These projections are, in large measure, validated by the range of responses provided by industry to those solicitations. The cost structure of those bids goes through rigorous cost modeling and price justification internally before being presented as a part of the company's response to the RFP. These activities could set the benchmark from which incremental and final VE incentives can be set. The appropriation to fund VE incentives would need to be added to the program budget and included with the overall funding appropriation. Anecdotally, we have seen cost overruns that exceed 50% of the original ICE (GAO 20-439, 2020). It seems reasonable to set an initial threshold for the incentives at 25% of that. Notionally, if the idea works, the program would save 75% of their overrun.

There are a variety of contract award and incentive structures applicable to the activities needed to manage the technical and business objectives of building major weapon systems (DoD, 2013). Development of new products is, by its nature, uncertain with respect to the input requirements and ultimate design synthesis. As such, the guidance is focused on cost-plus contracts and structures associated with awards that are aligned to the type of work being performed. The recommendations associated with integration contracts are especially valuable to consider.

While any new RFP released by the DoD is a communication tool, the evaluation criteria for the contract award is the part of the vehicle that speaks the loudest. To make ACVIP and VE effective in driving positive outcomes across the contract period, evaluation criteria must show that performing virtual integration will be rated very highly as a discriminator for award. In fact, typical evaluation factors include cost and performance. It is likely that ACVIP could be a significant contributor to both of these.

To ensure that industry remains engaged in using ACVIP, contract incentives should be in place for continuing to reward the right behavior: identifying showstoppers, and reducing acquisition risk. Incentive Fee type cost contracts are suitable, but Award Fee types may provide the most utility. In such contract types there is usually an award fee board that can adjust the criteria to match the state of the program and the types of performance improvements that are most sought after by program management. The best carrot should be at the end of the contract period, where the VE improvements captured during the contract period can be evaluated in the near hindsight of contract closure to the mutual benefit of both parties.

Well-established patterns of schedule slippage and cost overruns in software programs can also be mitigated through incentive structures. Metrics generated from performing ACVIP during early stages of the design process can be used to inform contractor fees, incentives, and terms. Programs can use these early design and synthesis stages to reduce program execution risk, and the contractor can benefit financially through the incentive structures—a win-win environment of transparency and collaboration.

There is a great degree of creativity in this space where customer and supplier can seek to deliver the greatest product for the money. However, the thrust of ACVIP is primarily focused on avoiding future unforeseen cost, not for reducing the profit available to industry. Juxtaposing total cost management with reasonable profit motives require thoughtful attention in order to avoid unintended consequences of artificially inflated starting bids or short-sighted cost cutting measures, both of which ultimately hurt program execution and reduce taxpayer value.

Integrating ACVI into Workflow

The introduction of a new technology generally provides opportunity for success and for failure. We learn from our mistakes, and many failures are often experienced before success can be claimed. In the context of the acquisition of CPS, ACVIP has been applied with mixed results. One significant lesson learned is that the ACVIP modeling and analysis has often been applied too late in the development process to have had a significant impact on the outcome. ACVIP is performed primarily with embedded computing systems because these systems require attention to CPS interaction, are timing constrained, have specific security and safety requirements, and require resource and performance analysis throughout their development. ACVIP specifically addresses these qualities. We believe the early surfacing of issues in these areas will translate into a more predictable and easier product development process.

Recommendations for Acquirers and Contractors

The context of the application of ACVIP strongly influences the planned workflow. Consider a contractor-oriented scenario where ACVIP analyzes process threads that are loaded on to a processor to optimize throughput and assess potential component deployment options. Compare this to a use case for ACVIP being applied by the USG to support an acquisition. For example, a potential acquisition scenario would be to acquire a new and improved widget (e.g., a new global positioning system device) for integration into a legacy fleet. In the acquisition scenario, ACVIP could be applied by the acquirer prior to the contract award to assess the likelihood that there will be problems during system integration and test. Contractors and acquirers will need to determine how to apply ACVIP to their existing development or acquisition processes, which are highly context dependent.

ACVIP practices are not widely adopted by the DoD’s CPS contractors. There is an awareness of the benefits of ACVIP, and some contractors have more experience than others, but for the most part the level of ACVIP expertise within the contractor community is low.

As contractors think about how to modify their workflow to take advantage of ACVIP, they will need to consider what (and how) they plan to measure the impact of ACVIP on their process.

Both for their own purposes, and for their government customer, industry will need to measure the effect of a change to their workflow on their process baseline. It should be noted that actual, validated process performance baselines that would support this type of analysis are rare within the contractor community. Instrumentation and metrics are essential in order to provide the opportunity to analyze the anticipated workflow for the project and strategically insert the ACVIP practices as early as possible in the product lifecycle to try to achieve maximum effect.

A popular way to facilitate process instrumentation is to write down questions that one would want to be able to answer with the collected data. This is the key activity in the Goal Question Metric Approach (Basili et al, 1994), as it ties the measured data, to the achievement of a business goal. As the questions are elaborated, it becomes clear to the developer whether or not their existing process instrumentation provides the data needed to answer the questions. This might generate an entire activity by the developer to establish a baseline for their process, the required first step before embarking on any process improvement. In the case of ACVIP, the key benefit is the ability to identify showstopper defects and issues earlier in lifecycle, and with an established baseline, some possible questions might be:

- Were the defects identified by the new workflow not identifiable by previous methods (i.e., not normally detectable until physical devices were present)?
- Should the defects that escaped the new workflow (i.e., not detected until physical devices were present) have been detected by ACVIP? If so, why weren't they?

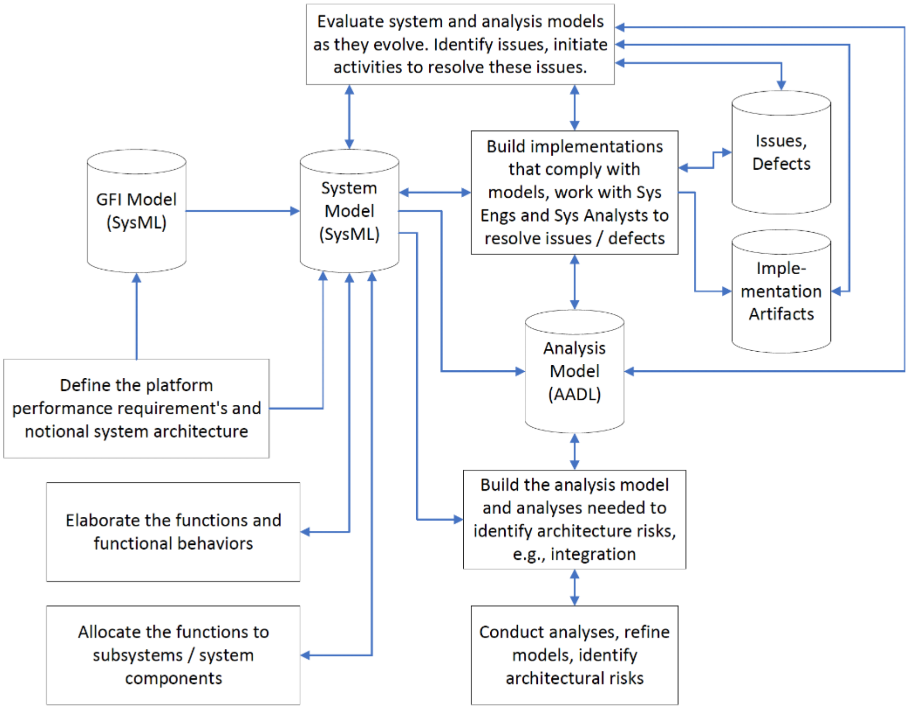
Evaluate system and analysis models as they evolve. Identify issues, initiate activities to resolve these issues.
Issues, Defects
Build implementations that comply with models, work with Sys Engs and Sys Analysts to resolve issues / defects
GFI Model (SysML)
System Model (SysML)
Imple-mentation Artifacts
Analysis Model (AADL)
Define the platform performance requirement's and notional system architecture
Build the analysis model and analyses needed to identify architecture risks, e.g., integration
Elaborate the functions and functional behaviors
Allocate the functions to subsystems / system components
Conduct analyses, refine models, identify architectural risks

Figure 4
Workflow Activities and Repositories

Figure 4 shows the key activities that we expect would benefit from ACVIP, and where the data comes from (or goes) for each activity. There is an intentional open aperture for delineation between organization (U.S. government, contractor, sub-contractor), or by role (e.g. system engineer, software engineer, hardware engineer, test engineer) in order to open up flexibility to alternative implementations and evolving practices. So, there are a lot of options for how to actually integrate these practices, or modify them to take advantage of the new process capability. There are non-trivial issues to consider, such as

- product line considerations
- proprietary data
- Government Furnished Information (GFI) detail
- modular open system approach (MOSA) requirements

Road to ACVIP Adoption

Adoption of any new approach must have the enabling environment set and be mature enough to be ready for full use. The enabling environment is in place:

- The Army and Carnegie Mellon University's Software Engineering Institute (CMU, SEI) have evolved the concept and practice of ACVIP.
- The SAE International® has nurtured the Architecture Analysis Description Language (AADL), which is in use by a variety of international industries. Supporting tools have matured and include commercial products that facilitate the use of the Object Management Group's Systems Modeling Language (SysML) standard; the Open Source AADL Tool Environment (OSATE) has been developed and released to the public by CMU, SEI and an array of international participants.

Adoption will inform the target community of the opportunities that are available to them along with the standards and associated tools.

- There are a set of training resources available to the community to aid in understanding the enabling environment elements such as AADL, SysML, VE methods and associated tools.
- VE principles are robust and well-used practice in a variety of industries and government organizations, but need to be tailored for ACVIP.

Adoption of these practices are mutually beneficial. As discussed in the contracting and incentive section above, this could be done by incorporating ACVIP as a contract award criterion. This would require the contractor to challenge their status quo and come up with creative and novel applications of ACVIP. Similar to the adoption of the microwave oven as described above, technologies, tools and techniques will evolve to shape ACVIP over time. The ACVIP content would appear in multiple places within the proposal document, and threaded through incremental and final contract evaluation activities.

We need to build some experiments that have hypotheses, producing data enables the evaluation of alternative implementations (with/without ACVIP) to verify/validate the hypotheses, i.e., demonstrate the value.

Bumps in the Road to Adoption

The synergy needed between software, hardware and systems engineers will need to evolve in ways that have not been emphasized in the past. ACVIP has its greatest advantage when models are throughout product development **and** sustainment. Therefore, we will need to extend the discipline of creating and managing embedded computing systems engineering artifacts.

Putting these kinds of efforts into areas of the lifecycle, where they have not traditionally been, will require a change in culture. Changing culture is hard and takes time through shaping and supporting new behaviors. Incentives can be instrumental in helping to achieve those shifts in behavior. Integrating a mindset of seeking out value improvements as a part of daily work must replace environments where people feel like they are only supposed to follow the documented requirements. Success for this transformation will be built on trailblazers trying new approaches and showing that the high likelihood for a payoff for both customer and supplier transition into added value that all side can benefit from. The trailblazers need to be recognized for making good faith efforts, even if some do not pay off.

Transformations, by their nature, are initially unsettling to a status-quo environment. Committing to a new process can lead to underperforming before those approaches can wring out implementation challenges (often referred to as the learning curve). For any such initiative there are always start-up and scaling issues that need to be worked through (Dubner 2020). Teams must be allowed to work on small improvements that can be scaled and grow such that they will eventually show skeptics positive examples that illustrate how to apply new technologies and process improvements focused on adding value.

Summary and Recommendations

After a time, repeated behavior patterns need to be accepted as a highly likely future event. Cost and schedule overruns for these highly complex and integrated CPS are going to happen unless we change the way we acquire these products. The consequences of schedule slips and unfulfilled performance goals are manifold and widely known. This future cost realization can be turned into a value-benefit bogey such that mutual needs are met: improved business performance for industry, and on-schedule acquisitions that fully deliver desired capability.

In CPS, “software is never done” (DIB SWAP, 2019) for three reasons: (1) capability changes are relatively affordable to introduce, (2) constant vigilance is needed to stay abreast of cyber-security risks throughout the lifecycle of the product, and (3) software development is a constantly changing field where new methods and tools are also in motion. As such, process improvements will need to be managed for any CPS and should be accomplished in a way that is mutually beneficial to both supplier and customer. VE is a proven framework that can be extended into CPS to evaluate and introduce innovative new methods as a win-win partnership between industry and government.

Evaluating VE initiatives in an MBSE and ACVIP model-based environment will reduce uncertainty and risk for incorporating capability and process changes as well as clarify the VE benefits of the innovation. Architecture-Centric Virtual Integration is a robust process that is supported by mature standards and tools on which to perform VE improvements. Embracing

ACVIP also reduces design and integration risk for initial product development and for follow-on fact-of-life cyber-security challenges and periodic capability improvement throughout the lifecycle (SAVI, 2015).

Integrating VE and ACVIP into the workflows of customers and suppliers engaged in developing CPS must be woven into the technical and business dimensions of the combined enterprise. This would include impacts to contract types, statements of work/objectives, development and operations processes, cost estimation, incentive mechanisms, and business processes.

Combining VE and ACVIP in this way is novel. As such, our first recommendation is to build a process performance baseline to be used to establish the rate at which "showstopper" defects escape the established process controls (e.g., peer reviews, unit tests, and hampering project execution). The data would be used as a reference to measure improvements, such as ACVIP implementations, that would be rewarded by VE. VE can only work when there is a solid reference point from which to measure. Even if you do not implement ACVIP, root cause analysis of the defects that escaped (and also analysis of the ones that were not detected) should be an essential part of all process improvement initiatives.

Our second recommendation is to pilot the application of ACVI and VE on selected acquisition programs to work out the details of the most effective metrics to be used, the most impactful analysis to be performed, and the value to be gained by all stakeholders. That pilot data will help develop measures and criteria for defining subsequent successful scaling of the initiative into other programs. Many innovations that have pleasing initial outcomes collapse under the weight of being fully deployed due to issues with scaling for wider use (Dubner, 2020).

Our third recommendation is to perform VE analysis at the enterprise level to determine the processes and products used for cyber-physical products operating in common domains and across related organizations to identify duplication and opportunities to gain greater efficiencies. Comparisons can be made to evaluate where points of leverage add overall portfolio value and determine systemic reuse strategies. VE can also be used to evaluate the positive or negative impacts of using/declining common products, modules, infrastructure, development, security, and operations tools, etc.

Taking on any process reengineering or transformational effort is fraught with risks and challenges, but they are worth doing where the benefits are high enough. We have identified areas to address and methods to employ that are replete with even greater benefits than the costs to bring them about.

Research Issues

- VE concepts can to be applied to software development strategies.
- Products are developed across multiple organizations, but integrating across a wide array of actors makes it very difficult to get everyone on the same page.
- Value of virtual integration is harder to address due to different constituencies.
- Costing the scale for software teams is inconsistent and best-guess approach.
- Better systems engineering up-front will ensure there is less expensive over the long run.
- VE has yet to be connected to CI/CD for weapon systems.

Results

This analysis will identify the aspects of VE that can be applied to the acquisition and lifecycle of CPS employing embedded computing resources. Programs will be able to identify the future cost of change and the ability to ensure investments in modeling and analysis are preserved instead of traded off in the early stages of an acquisition when they do the most good.

Keywords:

Value Engineering, Architecture Centric, Virtual Integration, Model Based Engineering, Model Based Systems Engineering, Software Engineering, Hardware Engineering, Systems Engineering, Functional Decomposition, Design Patterns, Integration, Modularity, Acquisition, Contractual Incentive, System of Systems, Cyber-Physical, Real-Time, Safety-Critical, Cyber-Secure, Maintainable, Automated Testing

Authors' Biographical Information:

Alfred R. Schenker
Mr. Schenker works in the SEI's Software Solutions Division, and has worked there for more than 20 years. He works to improve software acquisition and product development practices throughout the armed services, and other organizations. He has actively worked in software process, architecture, model-based systems engineering, and metrics. Before joining the SEI, Mr. Schenker spent more than 20 years in industry as an active contributor in all phases of product development. Mr. Schenker is also an inventor, and has obtained patents for a pressure switch (used in automotive airbag applications), and for a manufacturing process to seal gas inside a vessel.

Nickolas H. Guertin, PE
Mr. Guertin is a Senior Software Systems Engineer at Carnegie Mellon University's Software Engineering Institute. He received a BS in Engineering from the University of Washington and an MBA from Bryant University and is a registered Professional Engineer. His combined military/civilian career was in submarine operations, ship construction, weapon and sensor development and defense acquisition improvement.

He provides engineering and transformational leadership to programs in the system acquisition and engineering communities. He performs software systems engineering, improving competition, modular open systems as well as prototyping and experimentation leadership in government and academia for software and cyber-physical systems.

References

About SAVE International. (n.d.). Retrieved April 2, 2021, from https://www.value-eng.org/page/About

Basili, V.; Caldiera, G.; Rombach, H. D. (1994). "The Goal Question Metric Approach"

Boydston, A.; Feiler, P.; Vestal, S.; & Lewis, B. (September 2019). *Architecture Centric Virtual Integration Process (ACVIP): A Key Component of the DoD Digital Engineering Strategy*. U.S. Army.

Clements, P., Kazman, R., & Klein, M. (2012). *Evaluating software architectures: Methods and case studies*. Boston, MA: Addison-Wesley.

Defense Acquisitions Annual Assessment: Drive to Deliver Capabilities Faster Increases Importance of Program Knowledge and Consistent Data for Oversight. Retrieved April 22, 2021, from https://www.gao.gov/assets/gao-20-439.pdf

De Graaf, R., Van der Linde, G., De Jong, H., & Vogt, B. (2019). Value engineering as a specialty for SYSTEMS Engineering: Exploring opportunities. *INSIGHT, 22*(1), 41-44. doi:10.1002/inst.12237

Department of Defense Open Systems Architecture and Data Rights Team. *Open Systems Architecture Contract Guidebook for Program Managers* (June 2013). Under Secretary of Defense for Acquisition Technology and Logistics

Designing and Acquisition of Software for Defense Systems. (n.d.). Retrieved April 2, 2021, from https://dsb.cto.mil/reports/2010s/DSB_SWA_Report_FINALdelivered2-21-2018.pdf

Dubner, Stephen J. Produced by Matt Hickey. (2020, March 19*). Policymaking is not a Science (YET)* (Ep. 405). Retrieved April 02, 2021, from https://freakonomics.com/podcast/scalability/

Feiler. (n.d.). (Rep. No. CMU/SEI-2018-TR-002).

Feiler, Goodenough, Gurfinkel, Weinstock, & Wrage. (2013). *Four Pillars for Improving the Quality of Safety-Critical Software-Reliant Systems*. Carnegie Mellon University, Software Engineering Institute. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=47791

Hansson, Helton, & Feiler. (n.d.). *ROI Analysis of the System Architecture Virtual Integration Initiative* (Rep. No. CMU/SEI-2018-TR-002). Carnegie Mellon University, Software Engineering Institute.

Kendall, F. (2013, March/April). *Use of Fixed-Price Incentive Firm (FIF) Contracts in Development and Production*. Retrieved April 02, 2021, from https://www.acqnotes.com/Attachments/Frank%20Kendall%20Use%20of%20Fixed-Price%20Incentive%20Firm%20Contracts%20Mar%202012.pdf

Lions, J. et. al.(1996). *ARIANE 5 Flight 501 Failure.* Report by the Inquiry Board. Retrieved from http://sunnyday.mit.edu/nasa-class/Ariane5-report.html

Rodriquez, S. (2014). *Top 10 Failed Defense Programs of the RMA Era*. War on the Rocks. Retrieved from https://warontherocks.com/2014/12/top-10-failed-defense-programs-of-the-rma-era/

SAVI AFE 61S1 Report (2015). Retrieved April 2, 2021, from https://savi.avsi.aero/wp-content/uploads/sites/2/2015/08/SAVI-AFE61S1-05-002_Summary_Final_Report.pdf

Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage (a.k.a. the Software Acquisition and Practices [SWAP] Study). (n.d.). Retrieved April 2, 2021, from https://media.defense.gov/2019/Apr/30/2002124828/-1/-1/0/SOFTWAREISNEVERDONE_REFACTORINGTHEACQUISITIONCODEFORCOMPETITIVEADVANTAGE_FINAL.SWAP.REPORT.PDF

Value Engineering. (n.d.). In International Council of Systems Engineers (INCOSE) Systems Engineering Handbook.

Weapon System Sustainment: DOD Needs to Better Capture and Report Software Sustainment Costs. Retrieved April 02, 2021, from https://www.gao.gov/products/GAO-19-173

Wrubel and Gross (2015) *Contracting for Agile Software Development in the Department of Defense: An Introduction,* (Rep. No. CMU/SEI-2015-TN-006) Carnegie Mellon University, Software Engineering Institute