Carnegie Mellon University
Software Engineering Institute

Adventium Labs

# Digital Engineering Effectiveness

*Alfred Schenker*[1] *ars@sei.cmu.edu,*
*Tyler Smith*[2] *tyler.smith@adventiumlabs.com*
*William Nichols*[1] *wrn@sei.cmu.edu*
1 Carnegie Mellon University Software Engineering Institute
2 Adventium Labs



This material is based upon work funded and supported by the U.S. Army Combat Capabilities Development Command Aviation & Missile Center 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.

This material is based upon work supported by the U.S. Army Combat Capabilities Development Command Aviation & Missile Center under contract no. W911W6-17-D-0003/W911W621F703A with Adventium Enterprises, LLC d.b.a. Adventium Labs.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of U.S. Army Combat Capabilities Development Command Aviation & Missile Center.

# 1 Abstract

The 2018 release of the DoD's Digital Engineering (DE) strategy and the success of applying DE methods in the mechanical and electrical engineering domains motivate application of DE methods in other product development workflows, such as systems and/or software engineering. The expected benefits of this are improved communication and traceability with reduced rework and risk. Organizations have demonstrated advantages of DE methods many times over by using model-based design and analysis methods, such as Finite Element Analysis (FEA) or SPICE (Simulation Program with Integrated Circuit Emphasis), to conduct detailed evaluations earlier in the process (i.e., shifting left). However, other domains such as embedded computing resources for cyber physical systems (CPS) have not yet effectively demonstrated how to incorporate relevant DE methods into their development workflows. Although there is broad support for SysML and there has been significant advancement in specific tools, e.g., MathWorks®, ANSYS®, and Dassault tool offerings, and standards like Modelica and AADL, the DE benefits to CPS engineering have not been broadly realized. In this paper, we will explore why CPS developers have been slow to embrace DE, how DE methods should be tailored to achieve their stakeholders' goals, and how to measure the effectiveness of DE-enabled workflows.

# 2 Introduction

We, as an engineering community, are designing, assembling, and deploying the most ambitious and complex systems ever made. The details of these systems stretch beyond the ability of one, ten, or even one hundred individuals to comprehend; therefore, we must engineer these complex systems with teams of thousands. Success in pursuits such as these requires systems management—and a key tenet of systems management is measuring progress.

## 2.1 Problem Statement

The emergence of *digital engineering* (DE) has the potential to improve project outcomes (e.g., reduction of acquisition risk for cost and schedule) for cyber-physical systems (CPS) by enabling defect detection to "shift left". *Shifting left* is enabled by developing new methods (e.g., model-based analysis) that discover significant defects earlier in the product lifecycle. Allowing defects to escape and not be discovered until final integration and test, means that much more effort will be spent fixing them. Coupling the late discovery of defects with high expectations and pressure from management is the perfect environment for making careless mistakes. It will also likely result in other inefficiencies, reducing the project team's effectiveness, and potentially burning out the workforce. The benefits of DE have been clearly demonstrated in other domains (e.g., nuclear power system design). However, recent studies highlight the challenges of both implementing DE and measuring the DE process for CPSs. Shifting defect detection to the left suggests that requirements and design issues can be found earlier (before test of the code or embedded system). However, there is no guarantee that applying DE methods early in the development lifecycle for CPSs and software will result in the improved likelihood of attaining stakeholders' goals. In this paper, we review research on DE for CPSs and recommend how to measure DE methods.

## 2.2 Key Takeaways

Effective application of digital engineering is challenging. If we approach digital engineering as a box to check, we will fail. Instead, engineering leaders should:

- Establish individual and organizational goals that are qualitative, rather than quantitative.
- Establish measurement objectives early and foster a culture that encourages rigorous process.
- Incorporate tight feedback loops between the digital engineering artifacts and the evolving system design. Use the models to reduce system development risks.
- Clearly differentiate what you track from what you measure.
- Do not attempt to measure digital engineering application to cyber-physical systems development *purely* by counting defects. Instead, employ proven systems engineering design and testing practices, and use digital engineering to accelerate and improve them, rather than replace them.

# 3 Background

CPSs—also called embedded computing systems—are "engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components" (National Science Foundation, 2021). In the last three decades the category of CPS has grown to include virtually all automobiles and aircraft.

## 3.1 Research Context

Although the capabilities of modern systems are unparalleled, the costs—particularly the costs of software—are rising at an unsustainable rate. The amount of software used in modern aircraft is growing exponentially (as shown in Figure 1), and the cost of software is steeply rising as well. This situation leads to the question, "How can we avoid exponential growth in cost if software size is growing exponentially?"

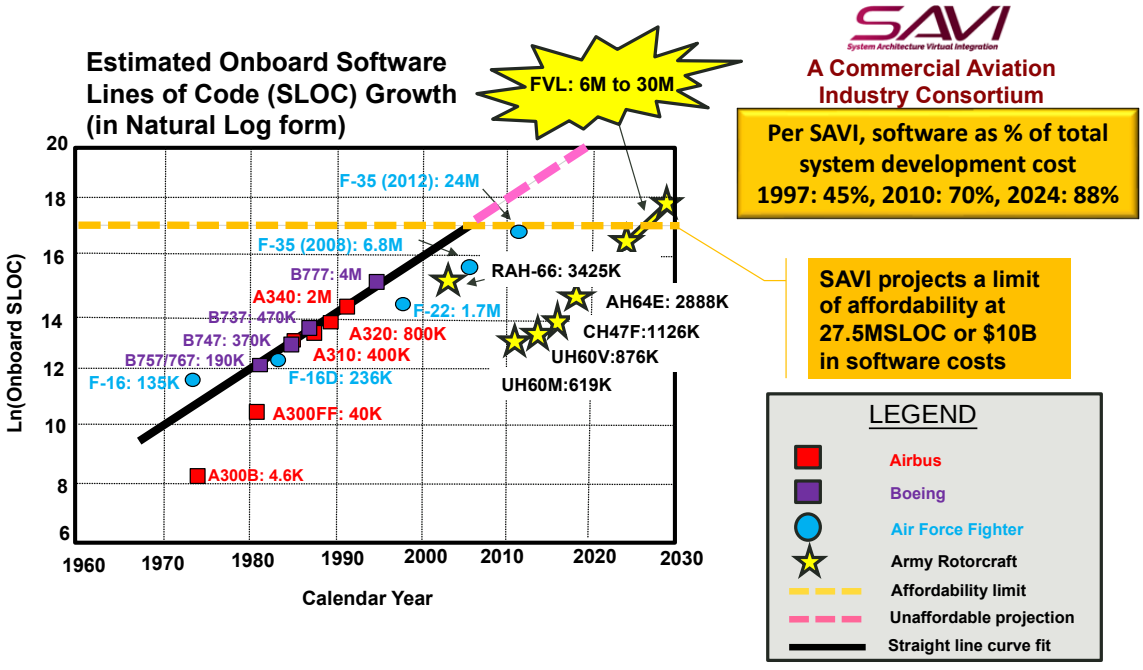


*Figure 1 The Growth of Software Lines of Code in Aviation, Graphic from (Lewis, 2019)*

At the beginning of the space race in the late 1940s, the Army, Air Force, and their contractors grappled with the challenges of the unprecedented complexity of large-scale rocketry. Early efforts (e.g., the Jet Propulsion Laboratory [JPL] *Corporal* rocket, which was developed and tested in the late 1940s and 1950s) were frequently stymied by integration errors and component failures. The ad hoc approach to building and testing components that had worked for simpler systems was no longer sufficient (Johnson, 2002, p. 86). Just like today, engineers and engineering managers were faced with exponential increases in complexity and insufficient tools to manage that complexity.

JPL addressed this problem by adding systems engineering rigor (i.e., design reviews, change control, configuration management), which contributed to the success of NASA's Mariner program in the early 1960s and 1970s (Johnson, 2002, p. 108).

Today's challenges in embedded computing system complexity echo the early challenges of the space race. Once again, we face a major leap in system complexity and we must adapt or face major cost overruns and missed deadlines, just as experienced by the Joint Strike Fighter program (GAO, 2021) experienced. The Mariner program demonstrated that systems engineering rigor could help programs contend with complexity, and the emerging practice of DE has the potential to meet our complexity needs now.

## 3.2 History of Digital Engineering

The Department of Defense (DoD) defines Digital Engineering (DE) as "an integrated digital approach that uses authoritative sources of system data and models as a continuum across disciplines to support lifecycle activities from concept through disposal" (DoD DES, 2018). DE has been with us—in some form or another—since the 1960s. The Apollo program relied heavily on simulations for both astronauts and mission control personnel to reduce risk to operations (Kranz, 2000). Modern manufacturing is driven by computer-aided design (CAD) tools that allow high-fidelity design and evaluation before bending metal. Modern aircraft like the Boeing 787 are extensively modeled and evaluated using digital tools to evaluate everything from flight dynamics to manufacturing. There is no doubt that DE works for *some* domains of engineering. For others, particularly CPSs, the answer is less clear.

There was a revolution in automated manufacturing when suppliers of factory automation equipment began to adopt 3-D (CAD) tools. The change occurred gradually, starting with Sales and Marketing; imagine the impact on a proposal when yours is the only one showing off a 3-D model of a new assembly line. The change then slowly became integrated into the machine design process. Eventually, it was possible to conduct design reviews using modeling tools, and animations could be constructed to show the sequence of operations. These changes enabled many different stakeholders (e.g., machine builders, machine operators, maintenance mechanics, safety engineers, etc.) to "see" the virtual machine months before it was built. Through the design review process, many stakeholders could modify the design to suit their unique needs, in ways that would have been done much more difficult using legacy methods.

Applying Finite Element Analysis (FEA) to the mechanical design process illustrates how a legacy method can be used to verify and validate a design, even as the design process undergoes a metamorphosis. FEA was first commercially developed in the late 1960s with help from NASA as an open-source product called NASTRAN. As the mechanical design process has evolved—from paper to 2-D CAD to 3-D CAD—analysis tools have evolved as well. In many cases these analysis tools can be integrated into the design process, providing the opportunity for a tight feedback loop to experiment with applying mechanical loads to the surfaces in order to observe the virtual effects, with the design updated as a result of the analysis.

The motivation to incorporate model-based analysis methods into the design process was influenced by the consequences of not applying them, e.g., late discovery of design flaws or architectural issues such as the Ariane 5 Launch failure (Dowson, 1997) or the Therac-25 radiation therapy machine failure (Leveson & Turner, 1993). The lead time associated with redesigning after late discovery could result in a serious impact on the program schedule; often these changes involved tooling with months of lead-time. Clearly, the means to perform virtual prototyping was necessary as a risk mitigation mechanism.

When defects are discovered late, the first impulse is to try to fix it with software. There is a common perception that software has no lead-time. Although software can be written quickly, depending on the complexity of the CPS, the implementation of the redesign might have unintended consequences. Ultimately a fix that was first thought to be resolvable through only software might require changes to other parts of the system. For example, a change to a data element format (as measured by a CPS) might require changes to the device driver, but also may impact the GUI and/or the system diagnostics. So, a "simple" fix might take weeks to implement and, in response to schedule pressure, might be implemented sub-optimally.

Nuclear plant design provides another recent example of successful DE use. Modern engineering tools simulate the physics of reactor designs. Simulation tools separately design and analyze structural mechanics, neutron transport, coolant flow, and heat transfer. The simulations are used to test and iterate on design decisions in a feedback loop, without the need to construct physical prototypes. Using overall plant simulation, nuclear plant designers can test operating conditions and lifetime performance. Construction has also benefited from 3-D CAD that enables the fabrication and assembly of complex systems in limited and confined spaces. In all cases, models can be used as the medium for evaluating changes or for external review, enabling or accelerating feedback loops in the design process.

Keys to the success of existing DE approaches include not only physically faithful simulations, but also interfaces that enable virtual and incremental design with an Authoritative Source of Truth (ASoT). In nuclear plant design, physical modeling propagates the hydraulic channels and material properties to neutron transport and fluid flow. Physical structure and neutron flux propagate to heat analysis. Structure, fluid flow, and flux propagate into plant analysis. Each model reuses the design from other models but makes local simplifications or includes model-specific additions. For example, heat transfer analysis can lead to changes in the nuclear fuel structure. Each characteristic has an ASoT, and each change can be analyzed as part of a design process feedback loop. Operational conditions can be tested, leading to redesign that begins with fuel properties and distributions. The tools available have a profound influence on an effective design workflow.

### 3.2.1 The Promise of Digital Engineering for CPSs

The promise of DE for CPSs, as with many other domains of systems engineering, is to "shift left." As shown in Figure 2, defects are much less expensive to fix when they are found on the left side of the traditional engineering "V" model. This benefit has been demonstrated conclusively for physical systems.

DE of CPSs should enable similar feedback loops to those observed in domains like nuclear power system design or FEA. An initial design includes components with interfaces, properties, and constraints. A virtual integration (analogous to the simulated behavior of a nuclear power plant) could either verify that the components are compatible or identify necessary design changes. A virtual analysis would simulate the execution of the virtual system to verify that (1) the constraints were satisfied and (2) inputs produced the expected outputs. Failure in this virtual test would require recycling of the design. It is important to note that this virtual integration and analysis does not replace more traditional

methods of modeling and simulation, it provides the means to perform the tests—and find the issues—earlier.

The economic case for DE based on early defect detection was argued by Feiler (Feiler, Goodenough, Gurfinkel, Weinstock, & Wrage, 2013) and Hansson (Hansson, Helton, & Feiler, 2018). In short, these researchers present evidence that, in the domain of embedded safety-critical systems, 35% of errors are introduced in requirements, and 35% of errors are injected in architectural design. Nonetheless, 80% of all errors are not *discovered* until system integration or later.

According to conventional wisdom, the cost of correcting an issue in later phases can be 300 to 1000 times the cost of correcting it in phase (Dabney, 2003). As the systems grow, the escalating cost of late discovery overwhelms development costs. (See Figure 2).

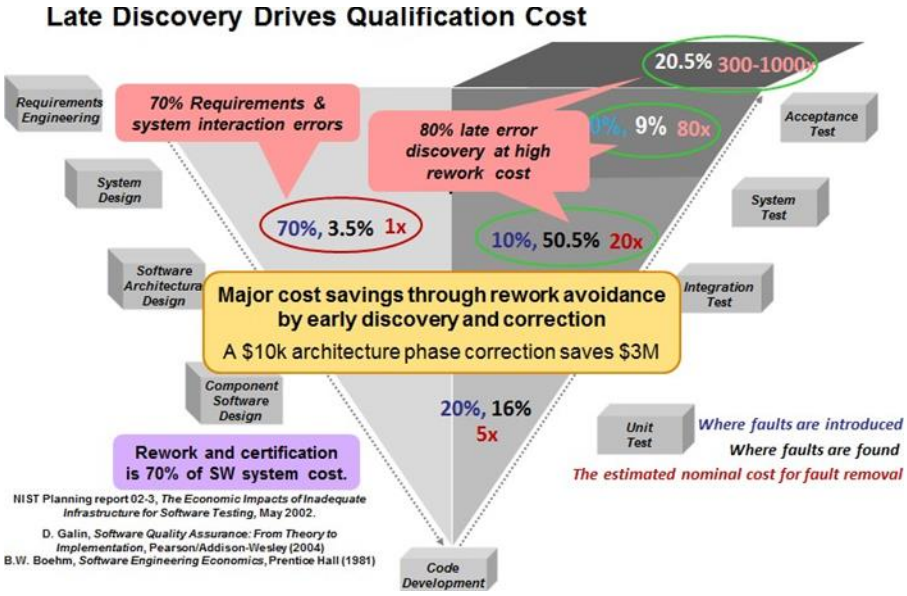


*Figure 2: Gap Between Defect Origin and Discovery (Feiler, Goodenough, Gurfinkel, Weinstock, & Wrage, 2013)*

## 3.3 Recent Department of Defense Digital Engineering Initiatives

### 3.3.1 DoD Initiatives

In recent years, Congress mandated the DoD to adopt a Modular Open Systems Approach (MOSA) for systems development, which directed procurement officials to pursue modularity in CPSs to reduce costs across families of systems.[1] The Air Force is pursuing an Agile software strategy to reduce software integration costs across platforms (Roper, 2020, p. 15). The Army spearheaded development of Architecture Centric Virtual Integration Process (ACVIP) to find cyber-physical integration errors

[1] See NDAA 2021 section 804.B.iii

early through virtual integration (Boydston, Feiler, Vestal, & Lewis, 2019). In FY2020 Congress also mandated that the DoD "establish a digital engineering (DE) capability to be used (A) for the development and deployment of digital engineering models for use in the defense acquisition process; and (B) to provide testing infrastructure and software to support automated approaches for testing, evaluation, and deployment throughout the defense acquisition process."[2]

The DoD is planning significant investment in DE, including providing a framework in the 2018 *Digital Engineering Strategy* (DES) (DoD DES, 2018). The DES relates five expected benefits from DE:

1. informed decision making/greater insight through increased transparency
2. enhanced communication
3. increased understanding for greater flexibility/adaptability in design
4. increased confidence that the capability will perform as expected
5. increased efficiency in engineering and acquisition practices

The *Digital Engineering Strategy* calls for practitioners to, "[e]stablish accountability to measure, foster, demonstrate, and improve tangible results across programs and [the] enterprise" (DoD DES, 2018). Recent efforts such as the Comprehensive Architecture Strategy (CAS) provide a framework for measurement by formalizing the relationships between key business drivers, key architecture drivers, and quality attributes for a CPS (CCDC, 2018). Now, a new way is needed to measure those quality attributes, as called for in the DES. Without methods to measure DE effectiveness, it will be difficult for new programs like the Army's Future Vertical Left (FVL) to gauge whether they are on track to reap the benefits of DE. Effective measures should provide systems managers with the information they need to determine how well they are meeting each of these five benefits.

### 3.3.2 U.S. Army JMR MSAD

The SEI and Adventium Labs participated in the Army's Joint-Multi-Role Mission System Architecture Demonstration (JMR MSAD) Science and Technology (S&T) program, which ran from 2013-20. This program exercised and evaluated DE tools and standards for CPSs, particularly tools oriented toward ACVIP. The program included three demonstrations, all conducted with significant industry participation and contributions:

1. The Joint Common Architecture (JCA) Demonstration (2014-2015) demonstrated the use of the Future Airborne Capability Environment (FACE™) Technical Standard and Joint Common Architecture (JCA) Functional Reference Architecture (Wigginton, 2016).
2. The Architecture Implementation Process Demonstrations (AIPD) (2017-2018) evaluated approaches to model-sharing among organizations.
3. The Capstone demonstration (2018-2020) evaluated a larger portion of the system development lifecycle, including the use of DE tools to adapt a CPS design to new requirements (Jacobs, et al., 2021).

The JMR MSAD program highlighted major new capabilities in DE but also brought the problem of measurement to the fore. The government asked performers to evaluate whether new DE tools and approaches were effective; however, the answer was unclear. Most performers tracked the effort (person-hours) required to perform various engineering tasks. However, these measurements did not yield clear results, for several reasons:

[2] See NDAA 2020 section 231

- Performers had to learn to use DE tools and had little time to separate learning from using. This made it difficult to compare the effectiveness of the new process against baselines.
- Performers were using multiple new technologies simultaneously, such as the Future Avionics Capability Environment (FACE™) and Architecture Analysis and Design Language (AADL). This confounded the challenge of learning these technologies and made measuring their effectiveness difficult as the "learning" period was not well defined.
- Performers integrated a mix of new and existing cyber-physical components but did not clearly differentiate the engineering costs and benefits of applying DE between new and existing components. We expect greater gains from applying DE on a *new* component during its design phase than from applying DE to an existing component after its design is complete.
- The schedule of the Capstone exercise likely motivated some performers to delay application of DE technologies until late in the design process, when the effectiveness of DE was likely diminished.

For example, Raytheon, one of the performers in the JMR MSAD Capstone exercise, presented the chart shown in Figure 3 at the program wrap up meeting. Raytheon measured labor hours spent conducting both DE and traditional engineering and software to complete development and integration tasks directed by the Government. Figure 3 refers to specifically to the ACVIP approach to DE.

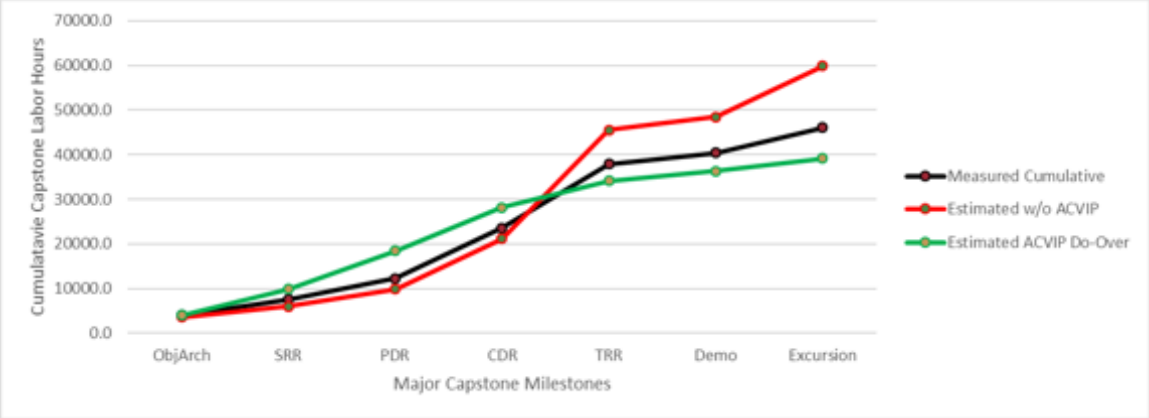


*Figure 3 Raytheon, a JMR MSAD Capstone Performer, Provided this Chart as part of their Lessons Learned Briefings (Raytheon and General Electric Aviation, 2020)*

The structure of the Capstone exercise meant that Raytheon had no definitive baseline against which to measure DE effectiveness, which presumably motivated their use of estimates for the purpose of comparison. These estimates show the expected "shift left," in which there is an up-front cost to DE (around the time of PDR, the red line is lowest) but a larger savings later (at TRR and Excursion the red line is highest). However, the Capstone excursion involved many new DE technologies and (as noted in Figure 3) performers like Raytheon predicted a different effort distribution in a hypothetical repeat of the exercise (green line). Although encouraging for the benefits of DE, these results indicate uncertainty in the digital engineering measurements employed in the JMR MSAD Capstone exercise.

The Capstone exercise performers used a combination of new and existing components as input to the integration exercises. This meant that, in some cases, performers conducted DE after a component design had already been completed, limiting the opportunities for DE to influence the component development process.

The DEVCOM Capstone Final Report noted the need for further work in defining the concept of a "defect" in the context of DE. (Jacobs, et al., 2021) As discussed later in section 4.3.2, defect tracking

can be an effective measurement approach to evaluate digital engineering, but only with a clear definition of defect. For example, some Capstone Mission System Integrators (MSIs) found that a supplied component did not behave as expected. The component was not necessarily *broken* but its behavior model (used to facilitate DE activities for the component) was not consistent with its actual behavior. (Jacobs, et al., 2021) Was this a defect in the implemented component? A defect in the model? The answer is not obvious. The definition of defect used for DE effectiveness measurement needs to be sufficiently precise to avoid ambiguity in such a situation.

#### 3.3.3 Continued Army Investment

Ongoing DoD efforts, such as the Army's Future Vertical Lift (FVL) acquisition programs continue to leverage DE via model-based Government Furnished Information (GFI) for performers and including ACVIP in requirements. The methods we propose in this paper will help government stakeholders assess the effectiveness of these techniques as implemented by their suppliers.

## 3.4 Digital Engineering Effectiveness Challenges

#### 3.4.1 Measurement Has Side-Effects

Measurement comes with a few challenges. The problems begin when determining the subject of the measurement. Before measuring something, we must know the goal of measuring it and the decisions the measurement will inform. Is the goal to become more efficient, produce a higher quality product, or something else? What are the implications of that goal? By "higher quality," do we want a product that is longer lasting, has fewer defects in use, or something else altogether? Further, are we using the measurement to compare, select, predict, or evaluate? Simply determining what to measure and why is only the beginning.

We cannot always measure something directly; instead, we often measure something related to the object of the measurement. For example, customer satisfaction is difficult to measure directly; however, usage trends or customer change requests can be useful indicators of the customer's satisfaction. However, two problems result. The first is accounting for confounding and/or complex relationships, and the second is that using measurements to control the system changes the system.

*Confounding* is a problem encountered when trying to identify cause-and-effect relationships because there are multiple causes and/or multiple effects. Some causes can be hidden, and sometimes causal factors interact. For example, a sidewalk could be wet from rain or from a lawn sprinkler. Just knowing the sidewalk is wet doesn't tell you why it's wet.

A related problem is *complexity*—when causes can have multiple outcomes. For example, ice cream sales may predict shark attacks because both have the common cause of summer heat. It's best to be cautious and understand whether the object of the measurement is a *cause* or a *result*.

The second problem, often called Goodhart's Law, can be paraphrased as "When a measure becomes a target, it ceases to be a good measure." More precisely, Goodhart stated, "Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes" (Goodhart, 1984). We interpret this statement as an observation that using a measurement from the system to exercise control changes the system's behavior. If the relationship is not trivial, the conditions under which the initial behavior was observed will no longer apply.

Therefore, we must understand the causal relationships of the objects of measure with the subjects of measure, including the environment in which the relationships were observed. This could be problematic for DE, where we aim to make substantial changes in workflows.

Another problem is that the measurements we choose might not translate to different goals. Measurement and analysis rely on assumptions about how a system behaves and the environment where the system operates. Assumptions valid for one analysis may not be suitable for another. Story points, for example, are often used as a proxy for relative effort, and sometimes as a proxy for relative size. This isn't such a big problem because cost and effort are related; however, that correlation can ruin an analysis that combines an assumption of size with an assumption of effort. Productivity is the ratio of size and effort. Defect density is the ratio of defects and size. Combining the derived metrics based on different assumptions can therefore lead to unpredictable results.

Finally, measurement plans can be inflexible. Selecting a measurement plan involves tradeoffs between the value of the information and the cost of acquiring measurements. Changes to the workflow can disrupt this balance. Changes in the information needs can require measurements that had been dropped from the original plan. Changing the measurement plan has costs and can introduces delays, therefore, expect resistance to changing an existing measurement plan. However, reusing the existing measure may not be appropriate.

These are just the conceptual challenges. The practical problems remain: instrumentation, accuracy, cost of measurement, and validation. We are not suggesting that measurement is impractical. Instead, we are saying that we must be cautious about simply reusing measurements devised for other purposes and applying them in the new context of DE for CPSs.

#### 3.4.2 Culture Change is Hard

As we change processes to incorporate the lessons learned from DE experiences, we need to recognize that implementing DE is likely to fundamentally change many aspects of the business model. For example, when we describe an activity as "enabling a shift left", we must recognize that we won't be able to achieve shifting left unless the necessary resources are available earlier in the life cycle. So, to build the capability to shift left, we need the same types of engineers (e.g., test engineers, integration engineers, acceptance testers) to be involved much earlier in the process.

This approach seems to make sense, until you view it from a PM's perspective. The PM fears that there will be extra effort needed at the end of the contract to resolve issues that arise when trying to perform final integration and testing. The PM might assume that early lifecycle spending to shift left will jeopardize the resources needed at the end of the lifecycle. Since the PM might not believe that shifting left will result in fewer issues in final integration and test, they might resist committing these critical resources early.

Perhaps the PM is right, and maybe that is the point of this paper. There is no guarantee that applying DE effort early in lifecycle will result in more efficient integration and testing. For example, we can spend lots of time building models and conducting analyses without producing meaningful results.

However, if DE is done well—the modeling and analysis activity focused on design tradeoffs that we know (either from experience or prior analysis) must be done correctly—the organization should be able to uncover and resolve many of the issues that would not normally be discovered until much later (i.e., during integration and test).

Therefore, implementing DE will result in higher quality and more predictable product development. The organization must be committed to the principles embodied in DE and continue to refine their process as they learn how best to apply their effort. Clearly there will be a stratification of competency in contractors as they evolve their processes to incorporate DE artifacts and practices.

# 4 What Information Is Relevant to Digital Engineering?

If you don't establish the abstract concepts you want to measure, it will be difficult to interpret your data. You must establish the questions you want to answer before moving on to concrete measurements.

What information do you need to know to run a successful DE effort? For this paper, we focused on goals one, three, and four from the *Digital Engineering Strategy*:

- Goal 1: Informed decision making/greater insight through increased transparency
- Goal 3: Increased confidence that the capability will perform as expected
- Goal 4: Increased efficiency in engineering and acquisition practices

## 4.1 Information Related to Decision Making/Insight

Consider a parent who is annoyed by their child leaving the front door open and allowing flies to get into the house. The parent, wanting to eliminate the flies in the house, devises a plan to reward the child for killing the flies. However, the opposite happens: The child let more flies into the house because they were being paid a bounty for killing them. When the parent changed the incentive to reward the number of consecutive days without letting a fly into the house, the child's behavior changed.

There is a similar conundrum with respect to shift-left processes. By their very nature, we hope to discover much earlier the issues that would "bite us" at the end of the project. It is easy to create measurements to count defects by lifecycle phase, but how do we know that the ones we find are the really bad ones—the showstoppers? We won't know until we get to the end of the project whether all the effort we spent to shift left paid off in smoother integration and qualification. We need ways to identify that our upfront investment in DE was worth it.

## 4.2 Information That Capability Will Perform as Expected

As part of the AIPD exercise in 2017 and 2018, participants created, exchanged, and virtually integrated models with one another. Through this activity, we learned that agreement on a modeling language is necessary but not sufficient to enable efficient communication of DE concerns among stakeholders. Additional information specifying modeling patterns and paradigms is required to facilitate effective DE collaboration between organizations. (This discovery motivated the creation of the AADL Annex for the FACE 3.0 Technical Standard and a variety of modeling templates used in later Army efforts (SAE International, 2019).

## 4.3 Information Related to Increased Efficiency in Engineering and Acquisition Practices

### 4.3.1 Where does rework originate, where is it performed?

In practice, counts of defects depend upon how you categorize them and when you choose to count them. Data suggests the relative number of defects injected during construction is high (Vallespir & Nichols, 2012), however, most of these defects are usually detected and addressed in reviews or unit test without being recorded in the defect tracking system. That is, these defects tend to be found near to their origin and are not the expensive defects that delay projects (Boehm B. , 1981).

Although rigorous unit test is effective for construction defects, it is far less effective at finding requirements or design errors; these continue to be found during integration or system test. This gap in

origin and removal of architecture and design errors is thus an opportunity for DE to speed up the process by identifying errors at or close to their origin.

### 4.3.2 Are we learning from our mistakes?

A leading indicator of an organization's culture is the way that the organization adapts its processes to account for defects or issues that escape in-phase detection. Sometimes, it is unavoidable (i.e., the issue was so complex that the investment in modeling and analysis to find it would have been ineffective). However, many times a retrospective analysis (e.g., a post-mortem review) of the issue points to a potential change to some aspect of the development/test process that will reduce the likelihood that this same problem will recur. Characterizing the types of defects as they are discovered and conducting some form of triage enables the organization to understand where it needs to focus its attention.

Many organizations that seek to apply modeling and analysis methods to their development process already possess the data they need to focus their DE efforts. These organizations have historical records of prior efforts that should help them understand which types of defects are typical, and which are not. Standard techniques, such as Pareto analysis, can be used to organize the data to illustrate where the pain is, and DE experts can recommend methods that address these pain points.

DE methods (e.g., modeling and analysis) are providing new ways to assess product design and architecture earlier in the lifecycle. So, as we perform root cause analysis, we need to ask this question: How could we have found this if we had been using DE methods? The answers to this question should provide the right type of guidance to the organization.

Repenning and Sterman describe the challenges of workflow and culture change in their article "Nobody Ever Gets Credit for Fixing Problems That Never Happened," (Reppening & Sterman, 2001). In this article, they use the example of the rush to embrace Total Quality Management (TQM) that many organizations adopted in the 1980s and early 1990s. They note that by the mid-1990s TQM appeared to have run its course, and these same organizations were abandoning it. They were now rushing to search for the next "silver bullet."

However, their research showed that "[…] companies making a serious commitment to the disciplines and methods associated with TQM outperform[ed] their competitors." They go on to point out that business leaders are facing a cultural paradox. In a world where it is becoming increasingly easier to identify and learn about new technologies that could improve their performance, the challenge is how to implement the technology the right way. They point out, "You can't buy a turnkey six-sigma quality program. It must be built from within."

The same applies to how model-based methods, such as those characterized as DE, need to be viewed. Each organization has its own challenges and nuances, and the application of the DE methods must be engineered to fit the organization's culture.

# 5 How Do You Measure Digital Engineering?

Project management measures are not necessarily the same as measuring the effectiveness of a paradigm shift, in particular the use of DE to "shift left." A major lesson from the JMR MSAD Capstone demonstration was that performers reused measures intended for project management rather than measurement tailored to evaluate the performance difference resulting from a change in methodology.

## 5.1 Measurement of Decision Making/Insight?

Although some processes are straightforward to measure, effective DE is typically a more subjective than objective measure. We can observe what happens, but not why it happens. We could assess the ways in which the DE has been applied. For example:

- How is the hardware modeled (i.e., is it a black box or a white box model)?
- Which computing resources are modeled (e.g., processors, memory, network bandwidth)?
- How are the models used for feedback control, or to verify and validate the system design?
- How do system engineers use the results of the model-based analyses?

These methods are distinctly not quantitative in nature. That is, we can measure some aspects but won't be able to produce a number to put on a scale and claim effectiveness. However, if we evaluate data in the context (with a demonstrated history) that the methods have proven to be effective on prior projects of similar scope, it is possible that a qualitative assessment, such as what is described above, is what we need.

However, measuring the things we are accustomed to measuring (e.g., effort, schedule, quality) might not be effective either. How do you know the fidelity of the modeling effort based only on the hours spent on modeling? We could measure attributes of the models (e.g., Has the model been reviewed?), but how do you know that the quality of the system is where it needs to be at this point in the life cycle? Do you base the measure of quality on the number of defects found?

To make matters more difficult, many organizations do not even count defects until the system goes through a major review (e.g., SRR, PDR). As the requirements are elaborated, (e.g., into use cases or scenarios) and distributed for others to review, no one is tracking the issues that are found as defects—until after the SRR milestone. At that point, the defects are all considered to be "baselined." How is it possible to understand the quality of the evolving system design under these conditions? More importantly, can we use DE tools (e.g., models and analysis methods) to inform the measure of quality?

Fundamentally, measuring changes to the culture or way of doing business is a multifaceted problem the defies simple or narrow measures. Objectively, we can measure activities and outcomes, but we must take care to measure enough different aspects in a sufficient amount that we don't encourage "hitting some target."

Figure 4 represents a way that an organization can use a DE approach to provide feedback and improved insight about the evolving quality of the system during, for example, requirements elaboration.

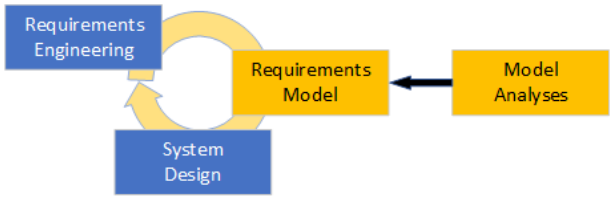


*Figure 4 - Feedback Control Using DE For Design and Requirements*

The use of a requirements model during the requirements engineering lifecycle phase provides an opportunity to verify and validate the planned implementation of a requirement in the context of the system, as it evolves. Note that this application is consistent with the traditional, batch-oriented approach typical of weapon system development and the more iterative approaches, such as Agile or

DevSecOps. The model provides the representation of the system that informs the system design activity as the requirements are elaborated.

## 5.2 Measurement of Capability Performing as Expected

Following the AIPD exercise, Adventium Labs conducted an Army-funded study on multi-organization DE. Adventium Labs found that comparative tracking of communication channels used between organizations (e.g., phone, email, model exchange) is an applicable method for determining whether your DE methods are sufficient for communication. (A decline in email accompanying introduction of DE tools may indicate that the DE tools are providing sufficient information (Smith, Whillock, Edman, Lewis, & Vestal, 2018).)

When creating DE artifacts of CPSs such as system models, the number of approaches to creating the model can be overwhelming. The number of available modeling styles, tools, and patterns can make it challenging for stakeholders to communicate with one another about what information to provide in DE artifacts. To address this issue, Adventium Labs also developed a collection of "report cards" for evaluating DE applications. These report cards provide a rubric for objectively evaluating DE artifacts. For example, the Model-Readiness Report Card for ACVIP consists of 15 measures that provide indicators about how well a project is applying ACVIP (e.g., Are you using a standardized modeling language? Are you adhering to its semantics?). The report cards can be used for evaluation by a customer, third party, or as a self-assessment.

### 5.2.1 How Do You Measure Performance?

Identifying the key information need begins with identifying the performance goals. Overarching goals include reducing cost, reducing overall development time, and increasing the resultant predictability of project performance. Although project management measures might be helpful, they are inadequate because they do not account for the multitude of factors that affect cost and schedule. That is, each project is a unique system.

In addition to the indicators for overall cost and schedule, we need more insight into what happens along the way. What were the activities performed, in what order, and what were the results? Where were evaluations performed, and what decisions were made? What are the sources of cost and rework? It's critical to understand what activities were performed, the ordering of performance, what they produced, and how they contributed to the performance goals of cost and lead time we enabled. Rather than just outputs, a detailed process analysis helps us answer not only what, but how and why.

### 5.2.2 Analysis of Rework–JMR MSAD

Shift left implies that we should shift effort from test into design and design analysis. This includes not only additional time in design (modeling) activities, but it also includes identifying design defects and implementing design changes. The indicators of successfully shifting left include more work in design, significant effort in design evaluation (virtual integration and virtual analysis), identification and classification of design defects in the design evaluation, and the amount of effort fixing those discovered defects. The key measures, therefore, include the following:

- effort spent on each of the specific activities
- counts of defects found by life cycle phase (e.g., design review, virtual integration, virtual analysis, and later physical integration and test)
- a judgement about which phase the defects were injected in
- the effort required to identify and implement the necessary changes
- categorization of the defects to determine the detection effectiveness of each activity by defect type

On the JMR MSAD Capstone demonstration, our initial attempt at measurement included creating work packages for the requirements, design modeling, virtual integration, and virtual analysis to estimate the effort. Unfortunately, this failed for the following reasons.

First, work was not tracked against specific work packages. Work packages were planned for a specific time period (sprint). At the end of the period, we knew which work was completed and the estimated effort (as a portion of the period effort). However, the actual effort update could only apply a sprint factor to all efforts. That is, the relative effort of the completed work packages to each other were not updated. We had no way to measure a change in relative effort for each work package, thus we could not identify a change in relative effort applied to activities (e.g., design, virtual integration, virtual analysis). This problem could be avoided by tracking work packages individually (e.g., Kanban style) rather than as a batch.

Second, the problem was compounded by failing to account for information needs when changing the workflow and work sequencing. The information need was to isolate effort in the different activities. Traditionally, a single story would progress through the workflow, perhaps noting the time of state change as different activities were completed. Instead, the story was decomposed into separate tasks to isolate the activity effort. However, stories were not tracked individually; they were only tracked as a batch within the sprint. Moreover, the related sub-stories were not executed consecutively and usually not in the same sprint. Thus, not only was all information on actual activity effort lost, but the sum of effort for related stories was also lost. The activity mix of stories changed in each sprint while reported effort was the estimate adjusted by a sprint-based factor. However, we did see that the virtual integration and virtual analysis stories were not occurring at the expected rates. This indicated that the workflow was not conducive to iterative feedback loops. This measurement problem might have been mitigated with a more pipelined workflow in which a design element immediately went into virtual integration or virtual analysis.

Third, we did not track defects. We had no way of knowing how much of the time in virtual integration or virtual design was involved in the time needed to execute the tests versus the time needed to fix the defects found. We could have measured the effort through time spent in the activity, or by specifically identifying a new story as rework resulting from a defect discovered in virtual integration or virtual analysis. There are options for setting up the accounting that interact with the intended workflow. For example, are defects fixed as they are found in the analysis, or are they sent for further evaluation and remediation? The measurement plan requires starting with the measurement goal and then assessing how to measure that goal within the context of the workflow. If the workflow changes, the measurement plan must be revisited.

## 5.3 Measuring Improved Efficiency in Engineering and Acquisition Practices

Properly realized and applied, DE should move defect discovery closer to the defect origin, resulting in fewer issues found in physical integration and test. ACVIP envisions accomplishing this by modeling interfaces and constraints (Boydston, Feiler, Vestal, & Lewis, 2019). Virtual activities can implement verification activities that identify requirements and design issues before building code or integrating code into the CPS. It is also possible that more rigorously designed products will have fewer defects. In either case we expect fewer design or architectural issues to be discovered during physical integration and test. Also, the additional steps early in the process should shift effort to earlier activities while reducing effort in integration or test.

Before physical construction, the modeled components should be virtually integrated to find architectural defects. In other words, integration issues, if present, should be found during virtual rather than

physical integration. A virtual analysis simulates the integration of the components operating as a system to uncover requirements defects. These activities require effort, but they also produce artifacts (issue reports and product changes); fortunately, effort and changes are both measurable and analyzable. Another change in workflow is incremental development with tighter feedback control using virtual integration and analysis.

The virtual integration and analysis should occur whenever components are created or modified, well before software integration, build, or system test. These defects should, therefore, be found more frequently. The reduced time (and scope of other changes) between a defect's injection and discovery in virtual integration or analysis should reduce the cost of identifying the source of the defect and the cost to fix it.

Using ACVIP as part of a DE effort should result in the following:

- incremental model development
- virtual integration and analysis
- more defects discovered in virtual integration and analysis
- fewer defects discovered in physical integration and analysis
- more effort expended prior to integration and test
- less effort expended post integration and test
- a shift in organizational skill sets to support digital tools
- increased exchange of digital artifacts in place of or supporting traditional development artifacts

| Lifecycle Phase | Injection % | Discovery % | Discovery % |
|---|---|---|---|
| Requirements | 70 | 3.5 | 80 |
| System Design | | | |
| Software Architecture | | | |
| Component Software Design | | | |
| Code Development | 20 | 16 | 15 |
| Unit Test | 10 | 50.5 | 3 |
| Integration Test | | | |
| Acceptance Test | | 9 | 2 |
| Post-Acceptance Test | | 20 | 0 |

*Table 1: The New Status Quo for the Development of CPSs*

Table 1 represents our view of the new status quo for the development of CPSs. While we don't think it is realistic to eliminate the discovery of defects in integration and acceptance test, we do think that by effectively using DE methods, an order of magnitude reduction in "out-of-phase" defect discovery is possible. Over time, the defect-based effectiveness measurement of DE should show a trend toward the desired defect detection rate shown in Table 1.

At least two model-based engineering mechanisms directly contribute to efficiency: (1) earlier discovery of design issues and (2) shorter feedback loops for learning and correction. A less direct mechanism includes benefits from precise, consistent, and exchangeable design artifacts. The result should be fewer defects, faster fixes of defects, and less total rework.

**Error! Reference source not found.** illustrates our view of how to represent the incorporation of effective DE practice in the context of the traditional system engineering V Model. The outside boxes are consistent with lifecycle processes represented in the traditional V Model. The black arrows repre-

sent the verification and validation activities that relate the testing to the design. The inside boxes represent DE artifacts in the form of (1) models (or representations) of the CPS or CPS elements and (2) analyses that can be performed on the models.

The analyses are typically developed to answer specific questions about the system's properties (e.g., latency [data], processor utilization, memory utilization). The DE environment provides the system development organization with an early view into the expected behavior of the physical system. With the exception of the Requirements model, the DE artifact, though labeled with different words, represent the same virtual model incorporating higher levels of fidelity. For example, the Software Architecture model might represent the components as black boxes, with just their interfaces modeled; but in the Component model, these black boxes would be replaced with white boxes, and the overall system model would be updated to reflect the higher level of fidelity in that part of the virtual model.

When this approach is coupled with an analysis capability that can be applied recursively (even automatically, if deployed in a CI/CD pipeline), the development organization can identify and react quickly to potential issues, solving the problems in phase. These problems would not normally be found until months (or even years) later, using simulation or hardware-in-the-loop laboratories.

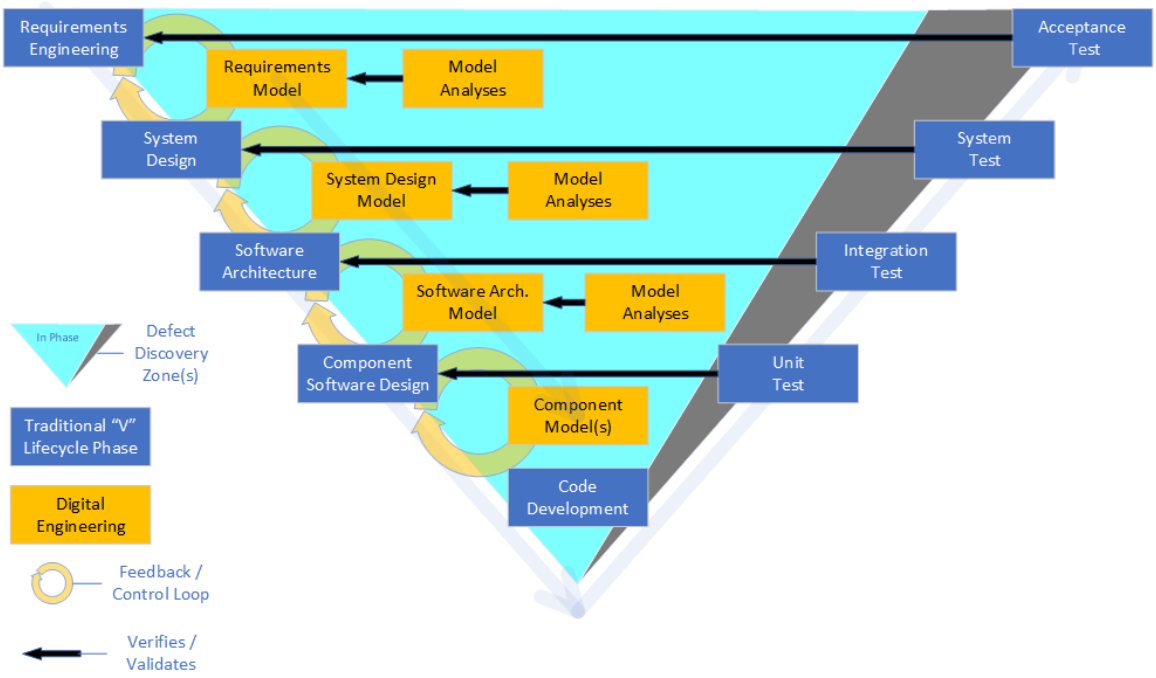


*Figure 5: Incorporating Effective DE Practice in the V Model*

Finding design issues earlier (preferably in-phase, but at worst in-phase + 1) should reduce total costs. As noted in Section 2.2.1, a substantial number of requirements and design defects are discovered in system test. Historical data in research by Boehm and Menzies shows that test or operational defects have cost factors of 10 to 100 times more than defects found earlier (e.g., in design reviews) (Boehm & Basili, 2001) (Menzies, Nichols, Shull, & Layman, 2016). The high cost of late changes is realized in the traditional CPS domain because of the (1) potential for changes to propagate through the system, (2) the difficulty of isolating the root cause of a test failure, (3) the likelihood that personnel who

performed the initial work will be unavailable for diagnosing and fixing the problems, and (4) fading memory resulting from the time span between initial development and fixes.

Tighter feedback loops that identify problems close to their creation mitigate all of these problems. Tighter feedback loops also benefit learning and improvement. Ideally, defect analysis includes examining the process to understand why the mistake was made. This is most effective when the original developers are involved, and the analysis is performed promptly. Those most familiar with the development not only are best suited to analyze the root cause of a problem, but also to adjust future practice to avoid similar mistakes. Reducing the time gap between the design and problem evaluation improves the effectiveness of root cause analysis.

Finally, the exchange of artifacts is an opportunity for misunderstanding. The authoritative source of truth reduces errors from redundancy and inconsistency. Digital artifacts can be used directly in reviews rather than tools such as PowerPoint, which require transcription (avoiding the associated transcription errors) and can allow more flexibility and precision when addressing reviewer questions. DE artifacts improve the effectiveness of communicating the design and identifying issues early.

# 6 Conclusion

If you take one thing away from this paper, it's that DE is a major change for CPS development, and that measuring the effect and effectiveness of that change requires a methodical approach. Don't focus exclusively on quantitative measures. They can be skewed by myriad factors that are out of direct control. Instead, increase the usage of qualitative measures and be mindful that your measures do not become targets.

What we are trying to get across is how to identify the practices that distinguish an organization that is just "checking the DE boxes" from one that is building the DE culture into its development approach. These distinguishing practices include the following:

- Be mindful of your expectations when measuring. Be ready for up-front costs (i.e., a learning curve) when adopting DE methodology.
- Recognize that culture change will likely be necessary. There are proven methods for overcoming cultural resistance that should be considered as part of the planning for DE.
- Identify separate measures that show whether DE is being used (e.g., tighter feedback loops) from whether it is effective (higher product quality and avoiding cost and schedule overruns).
- Learn how to apply DE practices for maximum effect. Some problems will not be appropriate, others may not warrant the additional effort. Although initially the organization should initially err on the side of over-modeling, the feedback control should indicate whether or not the effort is adding value.

What we learned from Capstone was that much of the DE was applied out of phase (i.e., after the design was completed, and used primarily for verification purposes), so it didn't function as part of a control loop. The measures applied in Capstone were also applied outside of the control loop. The result was measurements with limited confidence.

Opportunities for future research should include evaluating the structured application of DE practices as part of a feedback control loop to inform lifecycle design activities. The instrumentation for these evaluations is especially important since the resulting data will likely conclusively show the value of the early application of DE practices.

# 7 References


Boehm, B. (1981). *Software Engineering Economics.* Englewood Cliffs, NJ: Prentice Hall.

Boehm, B., & Basili, V. R. (2001). Software Defect Reduction Top 10 List. *Computer, 34*(1), 135–137. doi:10.1109/2.962984

Boydston, A., Feiler, P., Vestal, S., & Lewis, B. (2019). Architecture Centric Virtual Integration Process (ACVIP): A Key Component of the DoD Digital Engineering Strategy. *22nd Annual Systems and Mission Engineering Conference.* Retrieved from https://www.adventiumlabs.com/publication/architecture-centric-virtual-integration-process-acvip-key-component-dod-digital

CCDC. (2018). *Comprehensive Architecture Strategy (CAS) Version 4.0.* Redstone Arsenal: U.S. Army Combat Capabilities Development Command (CCDC), (Aviation and Missile Center (AvMC). Retrieved March 28, 2022, from https://apps.dtic.mil/sti/pdfs/AD1103295.pdf

Dabney, J. B. (2003). *Return on Investment of Independent Verification and Validation Study (Preliminary Phase 2B Report).* NASA.

DoD DES. (2018). *Digital Engineering Strategy (DES)*. Retrieved March 29, 2022, from Directorate of Defense Research and Engineering for Advanced Capabilities (DDR&E(AC)): https://ac.cto.mil/digital_engineering/

Dowson, M. (1997). The Ariane 5 Software Failure. *ACM SIGSOFT Software Engineering Notes, 22*(2), 84. doi:251880.251992

Feiler, P. H., Goodenough, J. B., Gurfinkel, A., Weinstock, C. B., & Wrage, L. (2013). *Four Pillars for Improving the Quality of Safety-Critical Software-Reliant Systems.* Pittsburgh: Software Engineering Institute. Retrieved March 28, 2022, from https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=47791

GAO. (2021). *F-35 Joint Strike Fighter: DOD Needs to Update Modernization Schedule and Improve Data on Software Development.* Government Accountability Office (GAO). Retrieved March 28, 2022, from https://www.gao.gov/assets/gao-21-226.pdf

Goodhart, C. A. (1984). Problems of Monetary Management: The UK Experience. In *Monetary Theory and Practice* (pp. 91-121). London: Macmillan Education UK. doi:978-1-349-17295-5_4

Hansson, J., Helton, S., & Feiler, P. H. (2018). *ROI Analysis of the System Architecture Virtual Integration Initiative.* Pittsburgh: Software Engineering Institute. doi:10.1184/R1/12363080.v1

Jacobs, W., Boydston, A., Ba, P., Rhamy, K., Davis, J., Padilla, M., & Roberson, H. (2021). *Joint Multi-Role Mission System Architecture Demonstration – Capstone Demonstration, Government Final Report.* Redstone Arsenal: U.S. Army CCDC Aviation and Missile Center (AvMC).

Johnson, S. B. (2002). *The Secret of Apollo Systems Management in American and European Space Programs.* Baltimore and London: The John Hopkins University Press. Retrieved March 28, 2022, from https://www.press.jhu.edu/books/title/2845/secret-apollo

Kranz, G. (2000). *Failure Is Not an Option.* Simon & Schuster. Retrieved March 28, 2022, from https://www.simonandschuster.com/books/Failure-Is-Not-an-Option/Gene-Kranz/9780743214476

Leveson, N. G., & Turner, C. S. (1993). An Investigation of the Therac-25 Accidents. *Computer. IEEE Computer Society*, 18-41.

Lewis, W. (2019). Tool Expo Keynote Address - AADL Addressing Software Related Integration Issues and Costs. *Tool Expo for Model-Based Embedded Systems Development.* Huntsville, AL.

Menzies, T., Nichols, W., Shull, F., & Layman, L. (2016). Are Delayed Issues Harder to Resolve? Revisiting Cost-to-Fix of Defects Throughout the Lifecycle. *Empirical Software Engineering, 22*(4), 1903-1935. doi:https://doi.org/10.1007/s10664-016-9469

National Science Foundation. (2021, January 7). *Cyber-Physical Systems (CPS)*. Retrieved from https://beta.nsf.gov/funding/opportunities/cyber-physical-systems-cps

NIST. (2002). *The Economic Impacts of Inadequate Infrastructure for Software Testing.* National Institute of Standards and Technology (NIST). Retrieved March 28, 2022, from https://www.nist.gov/document/report02-3pdf

Raytheon and General Electric Aviation. (2020, December 14). MSAD Capstone Program Wrap-Up MSI 3 Assault.

Reppening, N., & Sterman, J. (2001). Nobody Ever Gets Credit For Fixing Problems That Never Happened. *California Management Review, 43*(4).

Roper, W. (2020). *There Is No Spoon: The New Digital Acquisition Reality.* U.S. Air Force. Retrieved March 28, 2022, from https://software.af.mil/wp-content/uploads/2020/10/There-Is-No-Spoon-Digital-Acquisition-7-Oct-2020-digital-version.pdf

SAE International. (2019). *Architecture Analysis & Design Language (AADL) Annex F: AADL Annex for the FACE™ Technical Standard Edition 3.0.* SAE International. doi:https://doi.org/10.4271/AS5506/4

Smith, T., Whillock, R., Edman, R., Lewis, B., & Vestal, S. (2018). *Lessons Learned in Inter-Organization Virtual Integration.* SAE International. doi:https://doi.org/10.4271/2018-01-1944

Vallespir, D., & Nichols, W. (2012). An Analysis of Code Defect Injection and Removal in PSP. *Proceedings of the TSP Symposium 2012*, (pp. 3–20). Pittsburgh.

Wigginton, S. (2016). Joint Common Architecture Demonstration (JCADemo) Final Report. U.S. Army RDECOM. Retrieved March 28, 2022, from https://apps.dtic.mil/sti/pdfs/AD1012511.pdf

# 8 Biographies

Alfred R. Schenker

Mr. Schenker works in the SEI's Software Solutions Division, and has worked there for over 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 over 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.

Tyler Smith

Mr. Smith has over 10 years of experience that covers data modeling, software integration engineering, and user interface design. Tyler is leading the Army-funded Future Vertical Lift support effort, for which Adventium Labs is providing Model Based Engineering expertise, and FRIGATE, a NASA-funded effort to generate failure recovery plans from traditional engineering models. Tyler has led Adventium's data modeling and open API development efforts and spearheaded Adventium's analysis interoperability strategy, which enables rapid integration of AADL-based analysis tools into a wide range of third-party system engineering environments. Additionally, he oversaw the Capstone Pilot Multi-Organization Model Integration exercise that validated AADL-based model integration, ASoT practices, and CVIT.

William Nichols

Dr. Nichols is a senior researcher at the SEI with a PhD in Physics. His experience includes data acquisition, scientific software development, nuclear design and plant simulation, and software project management. While at the SEI he has coached software project management and published research on using software metrics for project and program management.