You Can't Wait for ROI to Justify Model-Based Design and Analysis for Cyber Physical Systems' Embedded Computing Resources

Fred Schenker
Carnegie Mellon University, Software Engineering Institute
ars@sei.cmu.edu
Jerome Hugues
Carnegie Mellon University, Software Engineering Institute
jhugues@sei.cmu.edu

Abstract

The practical, pragmatic benefits of building early architectural models of the embedded computing resources for Cyber Physical Systems (CPS) have been documented and demonstrated. However, the rate of adoption of this practice by the contractor community has been slow. Empirically, we have observed skepticism with respect to the increased cost of building these models, as being of sufficient value to justify their expense. This paper elaborates the reasons why using traditional methods, such as return on investment (ROI), to justify the increased expense (of building and maintaining these virtual models) is inadequate. Alternate ways to quantify and rationalize the benefits are discussed, but ultimately the decision to adopt may require a leap of faith.

We begin by describing the problem space and advancements in the design and implementation of the embedded computing resources for CPS. We discuss the proposed process change we seek: using model-based methods to reduce integration and test risk. We discuss the potential effects of that change on CPS, as well as our thoughts on ROI and the issues that can arise when using ROI. Finally, we recommend how organizations can move forward with a model-based approach in the absence of solid ROI data.

Introduction

Technological advancements have been suggested, supported, funded, and incorporated into our lives since the earliest times, and we continue to identify opportunities to push our technological envelope. Frequently, technological innovations are identified to produce incremental improvements, less frequently the improvement forces a change to the "way we do business." This paper focuses on a particular technological improvement, one that will have a significant impact on the "performance" of a project (i.e., cost and schedule) to build, test, and sustain a cyber-physical system (CPS).

The improvement we are advocating is that projects build virtual architectural models early in the system development lifecycle of the CPS they are developing. These projects must explicitly define the embedded computing resources within these models and use the models as early as possible to validate the systems' requirements. In particular, the models should be used to identify embedded computing system constraints, so the project can manage the constraint as a risk. To contrast this with our observations of current CPS development projects, we find that the constraints are not identified as risks, requiring mitigation, until much later in the lifecycle, typically when the components are being integrated in a lab. The late validation of the constraint usually causes big problems, e.g., significant cost overruns, schedule delays, and/or compromises in capability.

In the development environment of the future, early architectural models of the CPS, coupled with model-based analysis methods are applied iteratively and recursively from requirements analysis, through product design, and virtual integration and test. As design decisions are made, the architectural model fidelity is increased, enabling more accurate estimates of computing resource performance. Eventually the practical application of the model is replaced by physical hardware and software in a laboratory environment, i.e., a Systems Integration Laboratory (SIL), but the architectural models are kept up to date as issues are found and resolved. After the CPS is completed, the models are maintained, and are used to assess the impact of potential changes, possibly as part of a system upgrade.

By the time you reach the end of the paper, you should have a good idea of the practical, pragmatic benefits of building early architectural models of the embedded computing resources for CPS. You should also understand the challenges that arise when trying to use ROI to justify the increased expense of building and maintaining these virtual models. The paper elaborates, the reasons for this assertion.

The Problem Space

It is apparent that CPS are getting more and more complex. The software that is incorporated within the system gets to be a bigger part of the overall technical solution and the number of physical parameters that are monitored and controlled by the system contribute to this complexity. This increased complexity results in potentially hard-to-predict behavior, as it is very difficult to understand the suitability of a proposed system solution without a deep understanding of how its computing resources are going to be used (as part of the overall technical solution). We might think a solution is adequate, we start the implementation, and then we find issues. Typically, these issues are surfaced at the end of product development, as the components are being integrated, and the system is being tested. As time goes on (i.e., after deployment) the complexity only gets worse, making it more and more difficult to predict the impact of a change made as part of incremental updates or modernization efforts.

At the same time, these CPS development organizations have been slow to adopt a key potential process improvement, to develop a virtual architectural model with associated analysis tools that would help them deal with this very problem. This observation is evident from the cost and schedule issues that arise late in the development process (e.g., integration and test) for virtually every Department of Defense (DoD) cyber-physical system built in the last 30 years. We continue to claim that we "do the best we can," but in the end our efforts fall short, and system after system fails to meet its expectations. We are not doing the "best we can." We are doing the same thing we have been doing for the last 30 years…and we can do better.

One extremely common reason for not adopting newer methods (initially) is that there is a perceived need to prove that the new way is better than the old way. We have all heard people say things such as, "Better the devil you know…," or "The grass is always greener…" The decision makers that need to advocate to move forward with a process improvement can always find ways to delay, "Bring me a rock. No. Bring me a different one." The patterns of behavior for individuals and organizations are well-established. Everett Rogers' theory, *Diffusion of Innovations* is a widely accepted model for characterizing this behavior (Rogers, 2003). Terms such as *early adopters* and *laggards* are well known and used by many in industry.

In other domains, model-based design and analysis has been employed by engineers for centuries. For example, improvements in structural analysis were facilitated by work done by da Vinci, Newton, Euler, among many others, leading ultimately to the development of the *finite element method* for predicting stress in structural components in the mid-1950s. The model-based theories were incorporated into software tools, starting in the 1960s and continuing to this day. It is common practice now for mechanical engineers to use finite element models to help improve the quality of their designs, and to provide an element of verification not available prior to the development of the finite element models and analysis methods. The modeling tools are used iteratively as part of the design process, to optimize and to reduce elements of design risk.

Bridges collapsing, or rockets exploding on the launch pad are graphic images of design failure, and the need to prevent future occurrences can rise to the top of a nation's agenda. As such, it is unlikely that a need to demonstrate financial ROI existed back when we were developing the finite element tools to prevent these very public failures. In these cases, elimination of the public disgrace was the benefit, and the community was fine with that.

To contrast this with embedded computing resources for CPS, the design failures are invisible until the physical devices are connected. As mentioned above, this may be due to increased system complexity. Even when the devices are connected, and the system is not working the way it was supposed to, it may take quite a while to investigate and determine the root cause of the problem. Because these problems are surfaced in the lab, the cost to deal with them can be up to 80x more than what it might have been if caught during design. Models representing the CPS and specifically the CPS' embedded computing resources can be used during early lifecycle stages to predict these issues (or constraints) and can also be used to evaluate alternate designs that might mitigate the future problems.

We do not live in a perfect world. Models can be incomplete, assumptions can be made that are incorrect, boundary conditions may not be represented accurately. All these things might affect the quality of the models, analyses, and resulting designs. If the model-based methods produce an incorrect answer, for whatever reason, the development may go off the rails. However, we do not design systems assuming that the designers and engineers are going to make mistakes. We trust our teams and their processes to produce high quality designs. When we improve our design process, it is usually because we have identified new methods that enhance our understanding of the problem space. Who would argue that properly using models and analytical methods to provide higher fidelity design verification could possibly be more expensive than not doing so? The work to create, verify/validate, and apply the models will cost more than not doing so, but when problems and computing resource constraints are found early in the development process, we will avoid more expensive rework of the system, and possibly provide enhanced capability for new implementations. Better design will cost more.

Cyber-Physical Systems

Cyber-Physical Systems (CPS) are pervasive in DoD systems, their capability is extended by their embedded systems. Embedded systems are made of software and hardware sub-systems, and integrated into a larger system, e.g., in charge of monitoring and controlling a physical process such as the trajectory of a vehicle. They are often associated with real-time or safety non-functional requirements: providing a function that must be completed under time constraints, (e.g., respect of deadline, periodicity, etc.) while ensuring safety invariants, e.g., avoiding unsafe situations that would create an unbearable risk to the system or its environment. CPS adds extra complexity to the system because of the greater degrees of coupling between computations and physical processes and were first recognized by the NSF as part of an emerging field of research in late 2006 (NSF, 2010).

Because of the interleaving between physics and computer sciences concerns, there is hardly a single state-of-practice for engineering CPS: understanding the system concept of operations and high-level requirements is key to narrow down the engineering body of knowledge. For instance, controlling a swarm of UAVs will rely on control theory and flight dynamics in addition to wireless communication stacks and distributed algorithms whereas the definition of a robot operating along with human operators will rely on mechatronics, inverse kinematics along with stringent design methods for real-time safety-critical systems.

Hence, CPS engineering is deeply connected with both Systems Engineering approaches for capturing concept of operations and high-level requirements, methods for architecting systems and specific analysis methods. Industry standards have been developed to facilitate the development of CPS such as simulation techniques to validate a system or Digital Twin to monitor a system as it is being deployed (Bickford, Van Bossuyt, Beery, & Pollman, 2020).

Although these approaches support the engineering of CPS, they do not address the diversity of analysis methods required. As a response, Model-Based Design and Analysis has been suggested as a discipline of its own to support the broad need to address performance, safety, security, or behavioral analyses of a system.

Model-Based Design and Analysis

Model-Based Design

Model-based design, or model-based systems engineering (MBSE) is a key aspect of the DoD's press for digital engineering. Models have been used for centuries to provide an environment to predict the performance of products under development. We do not need to look back very far to see evidence of how models have helped improve our lives.

The evolution of the rail, used for railroads, has seen significant improvement from model-based design and analysis. Early rails were developed for horse-drawn wagons, and they were built using wooden rails, eventually replaced by cast iron. When the steam engine was introduced to power the transport, the increased weight of the locomotive caused significant breakage; the rails were too weak and brittle. This led to huge costs for maintaining the rails. Without having models to represent the material properties of different metals, we may never have gotten to the current standard for steel. Engineers were able to (1) increase carbon content, to improve tensile strength and reduce ductility, (2) increase manganese content to reduce abrasion, and (3) reduce phosphorus and sulfur impurities to improve the brittleness of the rail. All these improvements led to the creation of standards for the material content of the rail, specified by the American Railway Engineering Association (Walsh, 1909).

Without the material science, leading to innovations in the actual rail material, many of the advances of the industrial revolution would have been attenuated. This is because one of the key enablers of the industrial revolution was the means to efficiently transport these products. Necessity was the mother of this development, and it involved many different innovations. However, individuals and companies sponsored the innovations. There was a competitive nature that spurred on the innovators. The innovators did not know for sure, when new rail material was laid for the first time, whether this composition would work better than the previous, but as time went on, the innovators were able to get more and more precise with the chemical composition of the steel, the processes used to produce it, and the methods by which the rails are joined. Now we have standards, and predictable results.

An important observation to be made is that even though the models (i.e., the science) predicted the performance, the rail companies had to verify and validate the rail's performance before committing to large scale production. In practice, this not only provides confidence in the models, but it also provides the environment to understand why the models do not work as expected. This feedback loop improves the model as it incorporates the lessons learned from the evaluation. This is true for all model-based development, and we do not learn about these things if we do not build the environment to develop the models and test them.

Looking back at the evolution of the rail, what became of the companies that did not invest in the efforts to build better rails? No doubt that there were early adopters that benefitted greatly from the material science advancements and built these practices into their development process. There were others that did not fare so well.

In our current context, models are widely used for all sorts of different applications. In some engineering domains, they are the trusted and authoritative sources of truth for the design. The introduction of the model-based methods in such fields as mechanics, thermodynamics, electromagnetic spectrum, electrical engineering, logistics, maintenance, process optimization, and manufacturing have had a transformative effect on the “way we do business.” They are used to document critical design decisions, illustrating graphically the choices that were available to the designers, the design selection criteria, and the rationale for selecting the winning design. In some of these domains, the digital engineering environment is used to transfer model-based designs into analysis environments, and use the results to verify performance characteristics, or conduct model-based what-if analyses. A good example of this is mechanical design, using a model-based 3D computer-aided design (CAD) tool, and analyzing stresses and other aspects using finite element analysis (FEA).

In DoD CPS design, the move towards MBSE is progressing in the right direction. However, there is room for improvement. Many organizations are not working natively in the MBSE tools. They do their work outside the MBSE environment, then “document” the resulting design in the MBSE environment. To unlock the true potential of MBSE, we need to build the system models and the associated analysis environment, as has been done in the other domains, and use the digital environment organically to test design ideas and build quality in.

Model-Based Analysis

As stated in the section above, model-based analysis is how we leverage the investment in model-based design. Without analysis methods, the properties of new rail material compositions would not be evaluated until the rail was laid. Imagine the world where we didn’t do analysis as an integral part of the design process.

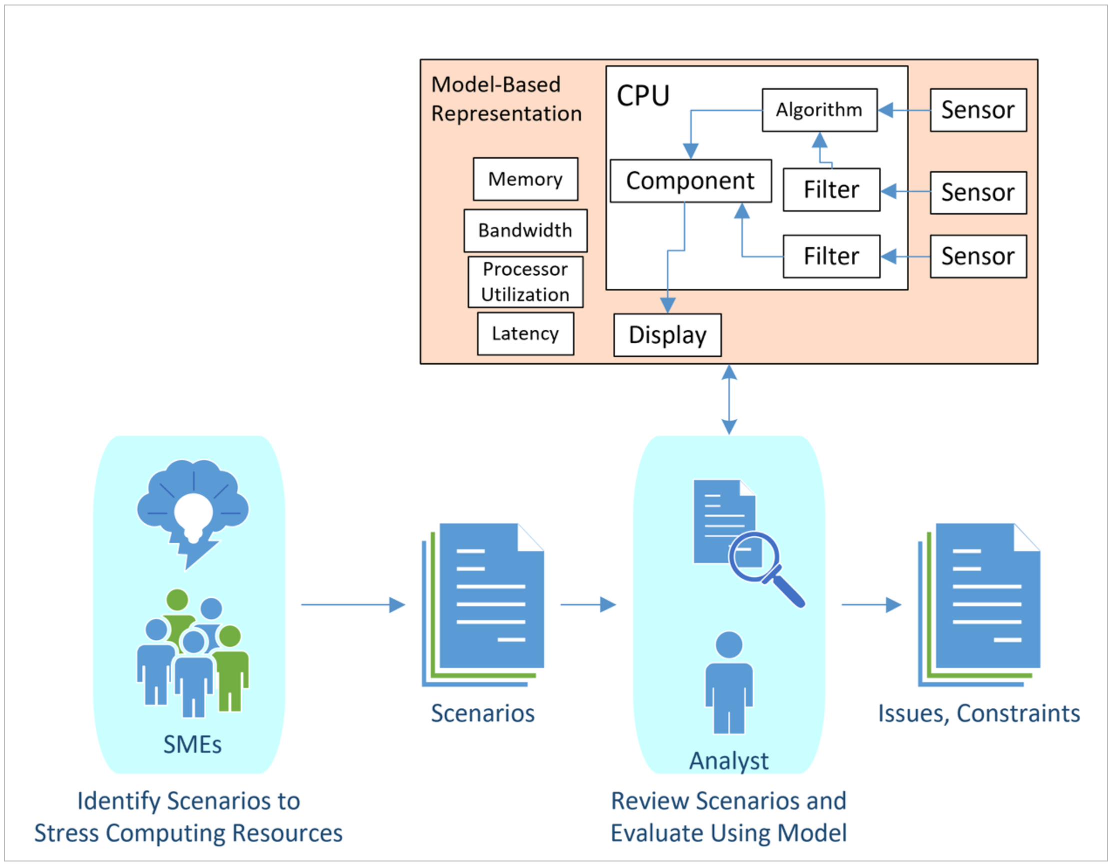


Figure 1: Notional Model-Based Analysis Process

As shown in Figure 1, in a mature model-based design environment, subject matter experts (SMEs) are used to identify design stressors to uncover elements of weakness in the design, and then to build an environment to evaluate designs as they evolve. A simple example is the use of a wind tunnel used to assess air drag in the design of a performance car. Using the analysis environment, the time needed to create a final design can be significantly reduced. With more experience and validation of the analysis methods and tools, the designers and engineers learn to rely on them to provide the early performance prediction needed for design verification.

Once constraints have been identified, they are managed by the design team. Having an environment available to evaluate scenarios that stress the constraints is an essential element to predict product performance. Many times, it is possible to identify unintended consequences of design decisions by using the analysis tools and facilities. These issues, if identified later in the design process (because of the lack of analysis capability or poor implementation of the design constraint) typically have a much more significant impact on the project. The economic case for investing in processes based on early defect detection was argued by Feiler (Feiler, Goodenough, Gurfinkel, Weinstock, & Wrage, 2013) and Hansson (Hansson, Helton, & Feiler, 2018). In short, they present evidence that in the domain of embedded safety-critical systems, 35% of the errors are introduced in requirements and 35% of the errors are injected in architectural design. Nonetheless, 80% of all errors are not discovered until system integration or later. Figure 2 is a graphic depiction of the impact of the late discovery.

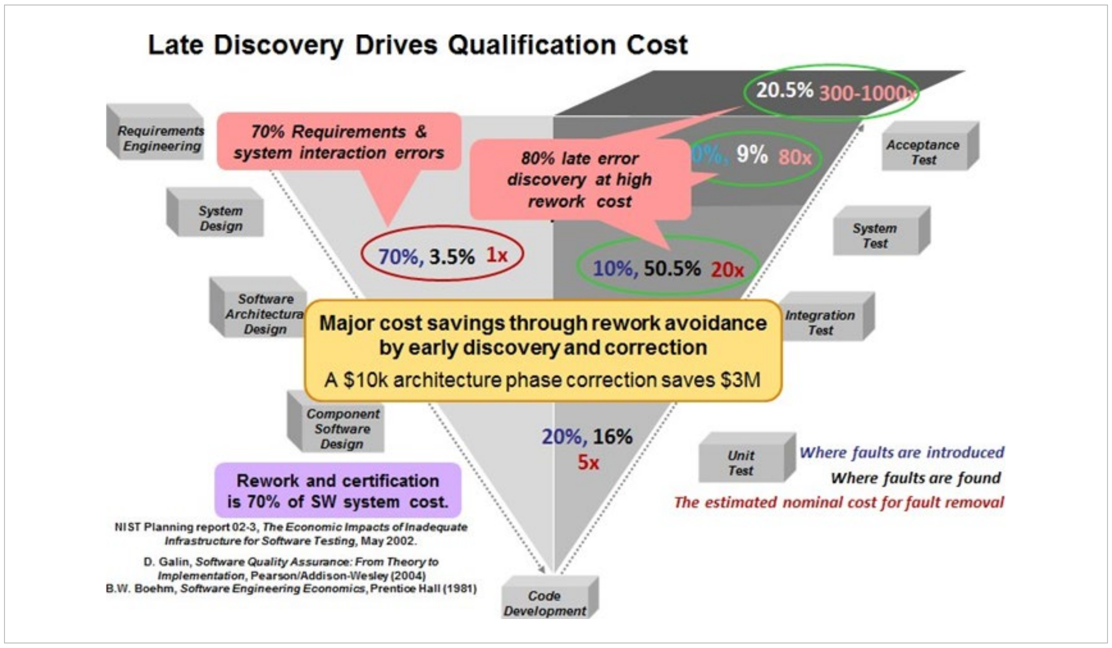

Late Discovery Drives Qualification Cost
Requirements Engineering
System Design
Software Architectural Design
Component Software Design
Code Development
Unit Test
Integration Test
System Test
Acceptance Test
70% Requirements & system interaction errors
80% late error discovery at high rework cost
20.5% 300-1000x
0%, 9% 80x
70%, 3.5% 1x
10%, 50.5% 20x
20%, 16% 5x
Major cost savings through rework avoidance by early discovery and correction
A $10k architecture phase correction saves $3M
Rework and certification is 70% of SW system cost.
Where faults are introduced
Where faults are found
The estimated nominal cost for fault removal
NIST Planning report 02-3, The Economic Impacts of Inadequate Infrastructure for Software Testing, May 2002.
D. Galin, Software Quality Assurance: From Theory to Implementation, Pearson/Addison-Wesley (2004)
B.W. Boehm, Software Engineering Economics, Prentice Hall (1981)

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

Normally, there will be opportunities to tradeoff critical constraint allocations to meet overall system requirements. There may even be options to increase the capacity of the constrained resource if the impact of the constrained resource will be too severe. The point is that the analysis capability provides the design team with early insight into the product performance allowing the design team to improve their management of the technical risk.

For DoD CPS, the physical aspect of the equipment constrains the overall analysis. In addition, the DoD acquisition timeline adds additional turmoil. To contrast, non-DoD CPS (e.g., automotive manufacturers) generally come out with new models every year, so last year's analytical tooling needs only minor modification to work for this year's model. It is also generally the case that last year's models functioned properly, constraints are known and planned for. The DoD acquisition timeline, and the systems engineering process (methodical and rigorous, but also following a waterfall approach) means that by the time requirements have been allocated to components, it may be too late to make some of the changes that would be needed as part of the management of a technical constraint such as an emerging need. It is important for the development team to have an analysis capability throughout the systems engineering processes to help with all those critical systems engineering decisions.

Analysis can be performed to support multiple domains (e.g., performance, safety, security). The different analyses could be performed using similar (or the same) models. In some cases, the analysis performed in support of one domain may conflict with a different domain. Managing the design and development using analytical tools should provide higher levels of design assurance and fewer issues during integration and test. In this context, the Software Engineering Institute (SEI) has been the technical lead of the Architecture Analysis and Design Language (AADL) SAE-AS-5506 standardization effort. AADL provides the foundations for the precise analysis of safety-critical CPS (SEI, 2023). A pilot study by the Aerospace industry consortium Aerospace Vehicle Systems Institute at Texas A&M under the System Architecture Virtual Integration (SAVI) in 2008 chose AADL as primary candidate to address the Embedded Software System affordability problem (Feiler, Hansson, de Niz, & Wrage, 2009). DARPA HACMS (2012-17) used AADL as part of their MBSE toolkit to use formal methods to build embedded computing systems that are resilient against cyber-attack because they have been proven not to have typical security vulnerabilities.

DoD Digital Engineering Strategies

In recent years, Congress mandated that the DoD adopt a Modular Open Systems Approach (MOSA) to systems development, directing procurement officials to pursue modularity in CPSs to reduce costs across families of systems.[1] The Air Force published an agile software strategy to reduce software integration costs across platforms (Roper, 2020). The Army spearheaded development of the Architecture Centric Virtual Integration Process (ACVIP) to find cyber-physical integration errors early through virtual integration (Boydston, Feiler, Vestal, & Lewis, 2019).

Additionally, the Department of Defense (DoD) has heavily invested in DE. The DoD provided a framework for DE in the 2018 *Digital Engineering Strategy*, which 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 (DE) Strategy (DES) calls for practitioners to, "Establish accountability to measure, foster, demonstrate, and improve tangible results across programs and enterprise" (DoD DES, 2018). Recent efforts such as the Joint Multi-Role (JMR) Comprehensive

1 See NDAA 2021 section 804.B.iii.

Architecture Strategy (CAS) provide a framework for measurement by formalizing relationships between Key Business Drivers (KBDs), Key Architecture Drivers (KADs), and Quality Attributes (QAs) for a cyber-physical system (CCDC, 2018). Schenker et al called for practical measures to support achieving these DE benefits (Schenker, Smith, & Nichols, 2022). Without methods to measure DE effectiveness, particularly model-based analysis, it will be difficult for new programs (e.g., the Army's Future Vertical Lift efforts) to gauge whether they are on track to reap the benefits of DE.

Return on Investment

The essential concept to understand regarding return on investment (ROI) is that when an organization commits to invest in something, there ought to be a financial justification for the investment (i.e., the investment should improve some aspect of the product). Sometimes the investment is long term (e.g., research, and it is not clear how the research will be applied). More frequently the investment is tactical, with a specific target in mind. There are generally many opportunities for investment, and one of the most important criteria in ranking the opportunities is the perceived value of the benefit. Built into the ROI calculus, there must be a discrete thing that the organization is trying to improve (e.g., time to market, cost to produce, quality). The improvement may not result in a direct financial benefit, but the investing organization will recognize that the improvement is desirable for the business. For example, reducing the time to bring a new product to market may enable greater market share. The key points are:

1. the benefit may not be easily dollarized because the financial return is indirect.
2. imputing a return introduces *subjectivity* into the ROI calculation.
3. the benefits that are not *directly* financial, e.g., lead time, market share, efficiency, are not made explicit.

Investments do not always pan out, or, more likely, do not meet the original goals of the investment proposal. There are predictable reasons for this, A change in a process may require training and the organization may experience a "learning curve" (i.e., a dip in productivity that eventually is overcome as the staff learns the new methods). A change in technology may be short lived, as there may be portions of the underlying technology that continue to evolve, requiring further investments and possibly eroding the potential benefits.

Some organizations adopt a different approach when it comes to investments in process or people, in which the investments are made incrementally, or continuously. These types of situations typically require instrumentation to measure the impact of the change. The culture of continuous process improvement is therefore more data-driven and requires a fair amount of consistency in the type of task being performed. In these situations, the actual ROI for each increment may not be measured, but rather the entire program may be evaluated over time. The organizations that adopt this approach also need to establish the culture that supports this type of methodology. Individuals need to be willing to change and adapt as the methods evolve.

In the context of DoD CPSs, we have observed regularly that systems fail or are constrained unexpectedly when entering integration and test. One goal that ought to be non-controversial is that an ROI goal for developers of DoD CPSs is to mitigate the impact of this inevitable occurrence. This could be achieved in several ways:

- Earlier identification of the constraints would allow for mitigation strategies to be planned and executed.

- Early identification and correction of defects/issues would improve the overall quality of the system, reducing the likelihood of significant amounts of defects to be found at integration and test.

What is the magnitude of this benefit? Prior research has cited that cost overruns, schedule delays, and technical compromises have a significant negative impact on these CPS programs. Even with the extra investment that we make to finish the programs, we often find that we wind up with *good enough* instead of *what we wanted*. Then, because the requirements have been *paid for*, we must accept that all the requirement implementations that exist are what we wanted, no matter how poorly they are implemented. When future changes are proposed to make the requirement what we want, the objection is that what you got was good enough, and we don't want the taxpayer to pay twice for the same capability.

The Opportunity for Cyber Physical Systems

As cited in the prior paragraph, our experience is that our inability to discover issues and constraints until we perform integration and test, coupled with the correction of the issues found during these activities is almost certain to cause program delays and cost overruns. We could just accept this "meta-physical certainty" and adjust our budgets and schedules to account for this. However, we don't. Time and time again, we claim that (1) we know what the issues are, (2) we have the best people on the job, and (3) we have learned from prior experience. We go in with rose-colored glasses. Then, it happens again, and we find ourselves in the middle of another acquisition nightmare. Note that this occurs both with new acquisitions, and with upgrades of existing systems. It could easily be argued that the magnitude of the impact on the legacy system upgrade is more significant than new system acquisitions… we all agree that it costs much more than it should to upgrade our legacy systems.

It is important to note types of issues we find during integration and test (we do not claim to be an authoritative source for all systems). We find that:

- There are basic incompatibilities between the components that comprise the system, usually connected through the infrastructure of the system. These are most frequently caused by incomplete, or inadequate interface descriptions. It is not that we have not reviewed the interfaces for completeness, it is much more the case that a critical element needed to achieve a process cycle time requirement cannot be achieved because the timing of the element was not (or was incorrectly) specified.
- Something unexpected happens when we connect the components together. These types of issues are difficult to predict. Systems are becoming more and more complicated with data being needed in many different areas of the systems.
- There are computing resource constraints that limit the system capability, especially when the system is under load. These will typically present themselves as latencies (i.e., operations will take longer to perform than normal). Occasionally memory or data bus issues will also constrain system performance or cause unexpected system behavior.

Concurrently, our experience is that most DoD contractors (that we have observed) are not taking advantage of model-based methods to address the root causes of these late-breaking issues. Specifically, we are not finding models of the computing resources being used to assess the adequacy of the planned computational, memory, and bandwidth loading. We do see that the analyses are being performed, but either they are not model-based, or the models are not kept up to date as the system evolves. The use of an architectural model to provide early assessments of computing resource performance issues ought to be at the top of the list of DoD CPS process improvements. Model-based methods are well established in many other engineering domains for similar purpose, but not in this one.

There are many possible reasons for their reluctance to embrace new technology. The most common objection we have experienced is that the modeling and analysis effort is somehow redundant and not necessarily as effective as more traditional methods. The detractors seek conclusive data that demonstrates the ROI. This is very difficult to provide, currently.

In the development environment of the future, we envision that integration and test engineers build a virtual environment to assess the state of development from Day 1, refining and elaborating the model(s) (as the designs are matured), but always able to answer fundamental questions about the system performance, safety, security, modularity, or any other relevant quality attribute. Perhaps the initial models are primitive and incomplete; however, the virtual environment will still be able to provide an early verification & validation (V&V) check on the systems engineering processes: requirements analysis, functional design, and allocation. The systems engineers would either use the environment themselves, or they would reach out to the integration and test engineers to conduct what-if analyses. The results of the analyses would get documented in the system design.

The "Problem" With ROI For Model-Based Analysis

Changing the way we do business is difficult. Changes are disruptive and require commitment. Commitment is needed from both the management and the technical staff. Prior research has shown the types of barriers that exist for situations such as this, and strategies have been developed to manage change (e.g., *The Lippitt-Knoster Model for Managing Complex Change* (Lippitt & Knoster, 1987)). The management must commit to this change wholeheartedly. They need to accept the responsibility for the change, create the environment that would make the change successful, and not back down in the face of opposition. The technical staff need to commit to the new way of producing work products. They need to be flexible with their personal process, and participate actively as a process performer, suggesting changes as appropriate to make the process better.

When deciding whether to employ model-based methods as part of the organization's culture, there are several ways to rationalize the change. All of these (described below) will have potential plusses and minuses. At the end of the day, management will need to decide the path forward. It has become customary to assess change using some data, such as ROI, to reinforce the decision to move forward. We don't think that it is possible to use ROI to justify the change to incorporate model-based methods and will explain our rationale in this section. *What cannot be ignored, through all the discussion that follows, is that model-based methods have been successfully employed in virtually every domain where they have been introduced.*

Creating an ROI Experiment

Wouldn't it be nice if there was a documented study that showed how to use the model-based methods to improve our process? It's not so easy. The following discussion (summarized below) makes several points:

- The DoD acquisition lifecycle is so long. By the time we get to integration and test we can't remember what we found during requirements analysis or other early reviews.
- Teams of developers will not have the same skill sets. Trying to set up an experiment to compare "apples with apples" will be challenging.

- We need to acknowledge that the organization would still be learning how to apply the new technology, while it was conducting the study.
- Determining what to measure may vary by organization. Different organizations will characterize benefits in different ways.

A realistic way to evaluate a proposed process against an established process would be to pilot the new one, iterate to establish a reasonably repeatable process, and then do some sort of side-by-side comparison of the two approaches. This is straightforward for normal process improvement, where the cycle time might be measured in weeks or months. In the context of DoD CPS, the time between early lifecycle work and integration can be five years. Five years, for the duration of an experiment, is a long amount of time. There would be so many opportunities to create legitimate anomalies over such a time interval (that could wind up invalidating the results) that it seems like the experiment would quickly be dismissed, either as a success or as a failure, without the data to support the decision. Even if the scope of the experiment were sufficiently small to enable a quick result, how likely would it be for the process evaluators to feel confident about scaling the result for a large-scale system? One of the real problems with CPS is that software is everywhere within the system, which leads to more and more complexity within the solution space. One of the most important benefits of the model-based approach is that the model will maintain relationships between the software components and predict behavior, that otherwise would be very difficult to predict. How is this measured? The duration of the experiment is an issue.

In a side-by-side experiment, it is important to try to limit the variable to the process itself. We would like to try not to introduce variation, but when humans are involved, and the processes are not performed routinely, it is easy to see that the human element would be an easy way for the experiment's results to be ruled invalid. It's not just the new processes that are not repeatable. In many DoD contractor settings, processes are performed at specific times during the project lifecycle. For example, early in lifecycle, there is a need to review and validate requirements, eventually leading to the systems engineering Systems Requirements Review (SRR). The activity to review and validate the requirements is intensely done for a short amount of time (as compared to the overall project). If we were introducing a new process, to use model-based methods to review, analyze, and validate requirements, a variable might be how the contractor manages their staff to keep them proficient in the process. Humans that perform the same process more frequently become more proficient and more repeatable and the resulting process performance is easier to baseline. According to a 1993 report published by Ericson et al., experts in a particular domain have more knowledge and experience than novices, which allows them to perform tasks more quickly and accurately (Ericsson, Krampe, & Tesch-Römer, 1993). This is because experts have more knowledge and experience that allows them to anticipate and recognize patterns that are critical for performing the task.

This leads to a discussion of a team's proficiency with a new process. It is expected that the introduction of a new type of artifact (e.g., the MBSE model and the resulting analysis capability) would take some time to stabilize the process that takes advantage of these tools. We typically would call this a *learning curve*. How long would it take for a team to become proficient with the new process? It's one thing for management to decree that "you shall use MBSE and model-based analysis." It's quite different to have figured out how to use these tools within the context of the "way we do business." We would also expect that an organization that is learning how to apply a new process would continue to tweak things over a series of pilots. Management ought to expect this and encourage incorporation of lessons learned towards the determination of how best to incorporate the new tools, although they should also expect that eventually the process will stabilize.

The means for calculating the ROI benefit will vary from organization to organization, possibly from project to project. Is the on-time delivery of capability to the warfighter more valuable than avoiding a $500 million cost overrun and two-year delay in schedule? Our experience is that the model-based methods will support either goal, but the development organization must decide what their benefit is going to be. Market share, for example, contributes to top line, increased revenue. ROI is a more complicated bottom line calculation. Other benefits might include flexibility and agility or aid of reuse on other products.

We need to develop objective criteria each time we try to apply the model-based methods. We should prepare to accept that the criteria may change from group to group, although there should be some commonality. For example, the main benefit to the project from adopting model-based methods ought to be that there is significantly less rework required during integration and test. How this is measured, either by effort or schedule or number of issues found (or some combination), ought to be expected. What else they might measure, possibly as leading indicators, is left to the process improvement teams.

How Can You Count Defects that Aren't There?

The primary benefit we have been describing in this paper is that using models and analysis methods earlier in lifecycle will lead to fewer issues later in lifecycle. A conundrum for the calculation of this benefit is that, if the model-based methods were so effective at identifying constraints and issues, many of the expected defects would not be present during integration and test. We wouldn't know, except by making assumptions, what the effect of the model-based methods were. Although we have evidence of the magnitude of the problem (from observation of prior project performance), it could be claimed that "we'll do better this time."

An interesting term from psychology that is useful for thinking about this is "counterfactual," or contrary to the facts. In our context, this might refer to the number of defects, issues, and constraints we find at system integration and system test. The counterfactual thinking would be to ask, *"If only we had been using a model-based approach from the beginning…?"* Daniel Kahneman and Amos Tversky pioneered the study of counterfactual thought, showing that people tend to think "if only" more often about exceptional events than about normal events (Kahneman & Tversky, 1982). It may be the case that the acceptance of the types of issues (that we find every time we build a CPS) is actually normal, and that it is hard for us to ask the if-only question because it is not exceptional.

Using counterfactual thinking, we would envision the world where we had employed model-based methods as an integral part of the design process and use the outcomes to justify the investment in the model-based methods. This approach would be facilitated by some of the work suggested in the following paragraph.

Post-Mortem Analysis

A different way to arrive at a model-based analysis justification for an organization is by using prior project data to illustrate the things could have been avoided if only we had applied model-based methods. By itself, this method would only identify the opportunities. The organization would then need to figure out how to incorporate model-based analysis into their process in such a way as to have discovered the issue earlier in lifecycle. This is a useful method for organizations to identify process improvement opportunities.

Regarding post-mortem analysis, it should be relatively straight-forward for an organization to develop a process:

1. Identify a set of projects to review.

2. Examine the defect database, pareto the defects by amount of time to correct the issue.
3. For each of the defects in the top 80%, determine how a model-based method could have been employed to prevent the issue from occurring, along with an assessment of how practical it would have been to have done this.
4. Summarize the effort that would have been saved, realistically, by using the model-based methods, and use this as the potential benefit for the investment.

Using this approach, a root cause analysis would assess where the issue could have been identified, had model-based methods been used. Note that this practice typically already exists for many organizations, where a defect found late in lifecycle is characterized as an *escape*, and that this type of data is used to improve the quality of design reviews. The model-based methods enhance the ability to critically review the system and component designs as they evolve, and an escaped issue could be considered a failure of the model-based review.

This process could be applied at the end of a project, or it could be done iteratively and recursively as the work progresses. Schenker et al suggested that the feedback on a set of processes, coupled with mature root cause analysis practices would rapidly evolve the adoption of model-based methods by an organization (Schenker, Smith, & Nichols, 2022).

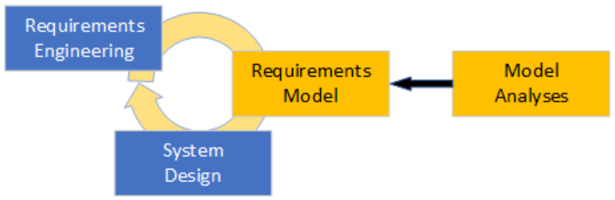


Figure 3: Feedback Loop Incorporating Model-Based Methods

Figure 3 illustrates a way that a model could be used to iterate different technical solutions as requirements are elaborated, and a system design is emerging. Model analyses could be applied to the model as it evolves to predict system characteristics, such as performance, safety, and security.

In the context of this diagram, there would be similar use of model-based methods at the next level of the design, i.e., system/software architecture. A mature development practice would perform root cause analysis of any issues found downstream, with the goal of understanding whether this issue could have been found in the prior step, i.e., requirements analysis. The goal is to use the models to identify as many of the issues and constraints as are practical to identify, accepting that some of the issues will not be practical to identify.

A critical review at the end of a project, addressing all the issues found during the project would be of significant value to an organization trying to implement such a continuous process improvement practice.

Acceptance by Analogy

Yet another way to rationalize the decision to move forward is by examining the experiences of similar applications of model-based methods in other domains. This ought to be a fast review. As stated above, our experience is that over and over again, the introduction of the model-based methods in such fields as mechanics, thermodynamics, electromagnetic spectrum, electrical engineering, logistics, maintenance, process optimization, and manufacturing have had a transformative effect on the "way we do business."

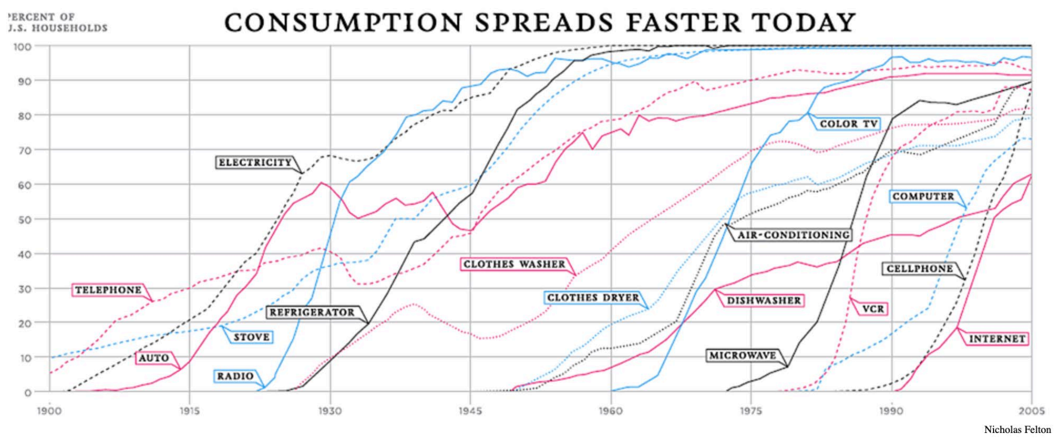
This approach accepts that the underlying benefit from applying a model-based approach will occur analogously in embedded computing resources for CPS as it does in the other domains.

Recommendations and Possible Path Forward

Throughout this paper, we have been asserting that there are real problems with using traditional methods, such as ROI, to justify an investment in better engineering practices, such as model-based design and analysis. The goal of this paper is to point out these problems, while providing some alternative means of justification. Our opinion is that once the organization decides to move forward with model-based methods, the process definition experts will figure out how best to apply the model-based tools within the context of their organization's product development lifecycle. Once a critical mass of practitioners has adopted the methods, then the methods will evolve naturally towards the goal of higher product quality, and a lower number of issues discovered late in lifecycle.

We recognize that using model-based methods for embedded computing resources is still in its early days. There has not been enough practical experience with the tooling and the methods, so it is very difficult to make claims about ROI. We'd be cautious of anyone that did make such claims. Comparing the maturity of this model-based technology to something like CAD and FEA is simply not fair.

Investigating other instances of technology adoption, it is not clear what occurred to make the new technology ubiquitous. The chart below, created by Nicholas Felton of the *New York Times*, shows the technology adoption trends for U.S. households for a variety of different technologies.



graphic reprinted with permission from Nicholas Felton

Figure 4: Adoption Curve for Various Technological Innovations

If we were to examine the curve for the microwave oven, for example, we see that there is a significant change in the adoption of microwaves that occurred in the mid-1980s. It is interesting that the first full year of sales of microwave popcorn took place in 1983. The 1987 *New York*

Times article, “Microwave Key to Popcorn War,” describes the rapid growth of microwave popcorn sales over the next few years, increasing in revenue from $53 million in 1983 to $250 million in 1986 (New York Times, 1987). That had to be accompanied by a rapid growth in microwave oven sales, which is supported by this chart. Microwaves had been around since the late 1960s, with a very slow rate of consumer adoption. After the introduction of microwave popcorn, the number of households jumps dramatically from less than 10% in the early 1980s to about 80% of U.S. households by 1990. It is hard to find a household now that does not have a microwave oven.

It's important to note that while the sales of microwaves were struggling through the late 1960s and 1970s, the appliance manufacturers did not back off their commitments to provide this new technology to the American household. Investment in new technology, advertising, manufacturing capability continued to occur despite the poor sales. Then, something happened, and there was widespread adoption of this technology.

We believe that a similar trigger will occur as the model-based methods become more widely used. One day we believe that we will wake up and find that companies that do not perform this type of work are the exception. In effect we will have leveled up our CPS development capability, leading to more predictable budgets and schedules, and more buying power for the DoD. We don’t know what the trigger will be, but we do expect that there will be a trigger. We think that CPS developers should accelerate their investment in this technology or prepare to be left behind.

Recommendations for Acquirers

When establishing a new discipline within an acquisition organization, it’s necessary to focus not just on the specific practices that should be established, but also on the care and feeding for the practices. Model-based analysis for embedded computing systems is not different. Acquisition programs reside within agencies or PEO structures, and there needs to be support for the practice both at the program level and at the higher echelon level.

1. Continue to set expectations with contractors that model-based design and analysis will be required for current and future acquisitions. Use this as a driver to spur investment in model-based methods. This might have several different elements to it:
    a. Language in SEPs, SOWs, and other acquisition documents
    b. RFIs asking contractors to describe their practices for integrating model-based methods into their development practices.
    c. Award fees and other contract incentives for successful application of model-based methods.
2. Train staff on how to use the tooling to be able to effectively review, verify, and validate contractor model-based deliverables.
3. Build an enterprise-level competency for model-based methods to establish consistency across programs, and collect lessons learned for future process enhancement.
4. Build the supporting infrastructure (digital engineering environment) to provide the capability to collect and analyze contractor deliverables.

Recommendations for Contractors

The challenge for contractors will initially be cultural, because using model-based methods as we have described will have a significant change to the way the contractor does business. Winning over the hearts and minds of all the practitioners, from managers to engineers, will be extremely challenging. In particular, the effort required to build a predictive architectural virtual integration model early in lifecycle will be viewed by management as an unnecessary expense, because the model-based methods have not been demonstrated with ROI, and the shifting left of the effort means that there will be less effort available when the real hardware and software show up in the SIL. We must get past this.

Once the culture has been established, and the team has accepted that the model-based methods will improve our likelihood of success, they will need to determine how to apply the methods to improve the existing development process. Then, they will need to establish the root cause analysis practice when defect escapes are found downstream, to try to improve the model-based processes.

1. Establish the culture to enable the model-based methods to thrive and add value.
2. Establish how the model-based methods are to be implemented. This will naturally involve some form of digital engineering, in identifying the tools, and configuring the tools into some kind of toolchain.
3. Train staff on how to use the tools to perform the new practices.
4. Develop a strategy for model management when working with heterogenous teams of contractors. Don't assume that it's *my way or the highway*. Elements such as where the authoritative source of design information resides needs to be established and communicated with all stakeholders.
5. Take a critical look at the defect resolution process. Examine the criteria for when root cause analyses are performed. Use the results of the root cause analyses to spur innovation with the model-based development methods.
6. Establish a project post-mortem process.
7. Establish a plan for how to account for the added costs and measuring the value received from applying model-based methods to the existing process.

Conclusion

In this paper, the authors make the case that ROI is not a useful way to assess the viability of adopting model-based systems engineering practices, especially for architecting and evaluating the embedded computing resources of CPS. Instead, we propose alternative ways, such as post-mortem analysis, analogy, or just a leap of faith to justify the increased usage of MBSE techniques to support these CPS projects. These findings are informing some of the current engagements performed by the SEI. The SEI is supporting multiple DoD projects in their adoption towards MBSE and will transfer some of these recommendations into practice as part of our transition work.

Bibliography

Bickford, J., Van Bossuyt, D. L., Beery, P., & Pollman, A. (2020, November). Operationalizing Digital Twins through Model-based Systems Engineering Methods. *Systems Engineering, 23*(6), 724–50. doi:https://doi.org/10.1002/sys.21559

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 March 28, 2023, 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, Redstone Arsenal. Retrieved March 28, 2023, from https://apps.dtic.mil/sti/pdfs/AD1103295.pdf

DoD DES. (2018). *Digital Engineering Strategy (DES).* Directorate of Defense Research and Engineering for Advanced Capabilities (DDR&D(AC)). Retrieved March 28, 2023, from https://ac.cto.mil/digital_engineering/

Ericsson, A. K., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review, 100*(3), 363-406. doi:https://doi.org/10.1037/0033-295X.100.3.363

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.* White Paper, Software Engineering Institute. Retrieved March 28, 2023, from https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=47791

Feiler, P., Hansson, J., de Niz, D., & Wrage, L. (2009). *System Architecture Virtual Integration: An Industrial Case Study.* Software Engineering Institute.

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

Kahneman, D., & Tversky, A. (1982). The Simulation Heuristic. In D. Kahneman, T. Slovic, & A. Tversky, *Judgment Under Uncertainty* (pp. 201-208). Cambridge University Press. doi:https://doi.org/10.1017/CBO9780511809477

Lippitt, M., & Knoster, T. (1987). *The Lippitt-Knoster Model for Managing Complex Change.*

New York Times. (1987, June 22). Microwave Key to Popcorn War. *New York Times*. Retrieved March 23, 2023, from https://www.nytimes.com/1987/06/22/business/microwave-key-to-popcorn-war.html

NSF. (2010). *Cyber-Physical Systems (CPS).* National Science Foundation (NSF). Retrieved March 28, 2023, from https://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf11516

Rogers, E. (2003). *Diffusion of Innovations* (5th ed.). Free Press Publishers. Retrieved March 23, 2023, from https://www.simonandschuster.com/books/Diffusion-of-Innovations-5th-Edition/Everett-M-Rogers/9780743258234

Roper, W. (2020). *There Is No Spoon: The New Digital Acquisition Reality.* U.S. Air Force. Retrieved March 28, 2023, from https://software.af.mil/wp-content/uploads/2020/10/There-Is-No-Spoon-Digital-Acquisition-7-Oct-2020-digital-version.pdf

Schenker, A., Smith, T., & Nichols, W. (2022). *Digital Engineering Effectiveness.* White Paper, Software Engineering Institute & Adventium Labs. Retrieved March 28, 2023, from https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=884463

SEI. (2023). *Architecture Analysis and Design Language (AADL)*. Retrieved March 23, 2023, from Software Engineering Institute: https://www.sei.cmu.edu/our-work/projects/display.cfm?customel_datapageid_4050=191439,191439

Walsh, W. J. (1909). *The History of Steel Railway Rails: Thesis.* Forgotten Books.

Biographies

Fred Schenker
Carnegie Mellon University, Software Engineering Institute
ars@sei.cmu.edu
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.

Jerome Hugues
Carnegie Mellon University, Software Engineering Institute
jhugues@sei.cmu.edu
Jérôme Hugues is a Senior Researcher at the Carnegie Mellon University/Software Engineering Institute in the Assuring Cyber-Physical Systems team. He holds a Habilitation à Diriger les Recherches (HDR, 2017), a PhD (2005) and an engineering degree (2002) from Telecom ParisTech. His research interests focus on the design of software-based real-time and embedded systems and tools to support it. More specifically, he concentrates on software architecture to support the design of complex software-based real-time and embedded systems, and programming languages and artifacts to support them. He is a member of the SAE AS-2C committee working on the AADL since 2005. Prior to joining the CMU/SEI, he was a professor at the Department of Engineering of Complex Systems of the Institute for Space and Aeronautics Engineering (ISAE), in charge of teaching curriculum on systems engineering, safety-critical systems, and real-time systems.