Product Lifecycle Management: The Salvation of Systems Engineering

Nathan W. Hartman
Computer Graphics Technology
College of Technology
Purdue University
nhartman@purdue.edu

C. Robert Kenley
Industrial Engineering
College of Engineering
Purdue University
kenley@purdue.edu

The Devil is in the details, but so is salvation.
— Admiral Hyman G. Rickover

Abstract. Many of the challenges faced by the systems engineering community articulated in *Systems Engineering Vision 2025* relate to detail-oriented activities that typically occur during the design, integration, assembly, and test of systems. With advances in computational capabilities, virtual system prototypes are available that use highly detailed representations of a system's components and provide a means for performing the virtual integration, assembly, and test of systems and for composing custom designs of systems and systems of systems. This paper describes how linking systems engineering with product lifecycle management provides a means to address these systems engineering challenges. This linkage will allow systems engineers to increase the actual and perceived value that they are delivering to the system's stakeholders and provide systems engineers a pathway to more meaningful and relevant participation throughout the system's lifecycle.

1. Introduction

Systems engineers are masters of the abstract; product designers are masters of the tangible.

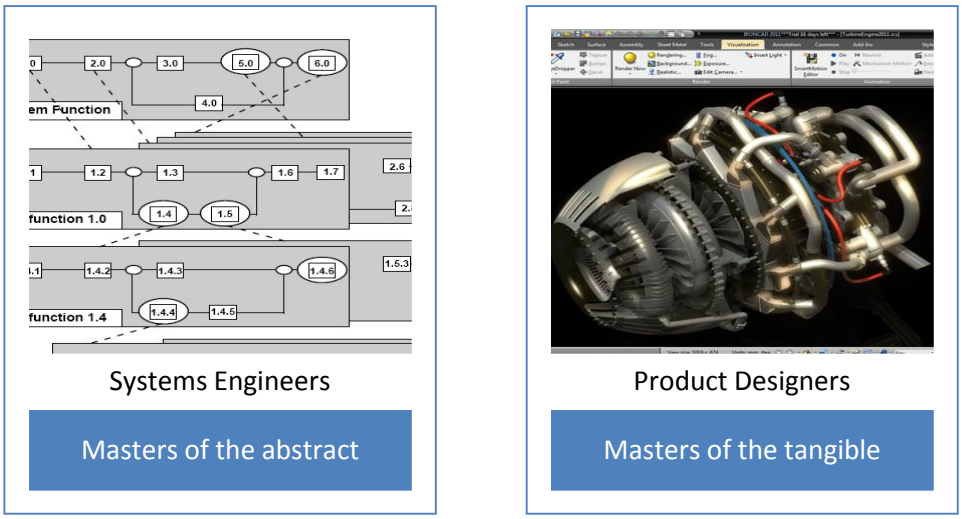

Figure 1. Two Engineering Tribes

The desire of the systems engineer is to ensure the pieces of a system work together to achieve the objectives of the whole of the system. The strength of systems engineers is their ability to articulate the objectives of the system and the operational sequences of the system from a functional point of view. This approach uses many abstract representations that are purposefully distinct from the representation of tangible system that executes these sequences in the real world to achieve the system's objectives. (Wymore, 1993, 7-8) describes several benefits of dealing in abstractions summarized in Table 1. The "avoids" statements are means that are more relevant to the masters of the abstract for achieving their end, which is to provide functionality to the user. The "provides" statements are means that are more relevant to the masters of the tangible for achieving their end, which is to define concrete solutions. The desire of the product designer is to specify an artefact that satisfies the objectives and associated constraints (Visser 2004). The strength of designers is their ability to produce tangible artifacts that can be used to manufacture and assemble products that can be offered for sale.

Table 1. Benefits of Using Abstractions

Avoids	Provides

• Confusing ends (what the user wants) with means (what the engineer designs)
• Stating the problem in terms of a preconceived set of solutions
• Confusing system-wide functional figures of merit such as reliability with:
  • its estimate based on a design
  • its estimate based on observations or measurements of a specific system under specific conditions
	• Insight through considering multiple solutions that may be:
	  • Practical solutions
	  • Idealized solutions
	  • Do-nothing (status quo) solutions
	  • Otherwise bizarre solutions
	• Context for evaluating and comparing solutions

2. The Challenges Faced by the Masters of Abstraction

A World in Motion: Systems Engineering Vision 2025 (INCOSE 2014) presents the challenges of a systems engineering community that seeks to ensure the pieces of systems work together to achieve the objectives of the whole of the system. As examples of current practice that touches on product lifecycle management, the vision describes three specific applications in industry:

1. The virtual rollout of the Boeing 787 Dreamliner that virtually created parts, and integrated and assembled the virtual system using visualization and simulation.
2. Model-based systems engineering at Ford Motor Company that manages design complexity for onboard electrical and software systems by applying multiple modeling technologies including UML, SysML, and Simulink that are integrated using an underlying system for configuration management and product data management (CM/PDM).
3. Allowing customers to compose custom designs of Scania trucks by providing clients the ability to select the cab, engine, chassis, engine, transmission, and accessories. The details of components such as engine cylinders, push rods, and

combustion chambers are specified to increase their interchangeability and to reduce the number of variations of components that must be manufactured and stocked.

In these three examples and others, it was not necessarily the case that systems engineers were the perceived or actual agents who drove these changes in business processes. Instead, changes like these are in response to the availability of technologies such as ever-increasing computational power, immersive technologies for data visualization, 3D printing (also known as additive manufacturing), advances in materials science, and the emergence of the internet of things. In addition, the globally connected IT environment that permits geographically diverse collaboration throughout the supply chain and with multiple stakeholders while designing products and after products are placed into service. These advances appear to be outpacing the advances in methods and tools used by systems engineers that allow them to contribute their wisdom and ability to apply abstractions to these situations to deliver innovative system solutions that keep pace with technology changes and still enable systems engineers to continue their role of ensuring product quality and safety up front.

Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases (INCOSE 2007). The current focus of MBSE is on tools and methods that capture document-based representations of requirements and functionality in descriptive models that can be analyzed for completeness and consistency within the scope of systems engineering and that have the potential to be converted to executable models that generate quantitative information for evaluating system behavior.

The link between the abstractions used in formal systems modeling for specifying, analyzing, designing, and verifying systems and the knowledge representations used for models used in design engineering, manufacturing, operations, maintenance, and repair is quite weak and must be strengthened to provide a model-centric approach that integrates technical, programmatic, and business realities. Model-based approaches that strengthen this link will enable systems engineers to apply their patterns of thinking and navigate among related viewpoints to understand the complex behavior of a product and the entire enterprise system that interacts with the product throughout its life cycle.

Systems Engineering Vision 2025 describes a scenario of model-based systems engineering being the "norm" for systems engineering execution with system models that have both "black box" views based on functional abstractions and "white box" views that incorporate geometric, production and operational views. In this scenario, these models will be linked to 3D printing to produce physical prototypes to increase the ability to investigate tradeoff between form, fit and function.

3. The State of the Art and Practice of the Masters of the Tangible

Product lifecycle management has many contemporary definitions (Grieves 2005; Grieves 2011, Stark 2011, CIMdata 2015), which tend to include the following common characteristics:

1. Product lifecycle management is generally viewed as a strategic method for conducting business within the extended product ecosystem – not just design and production within a company, but it also accounts for supply chain, sustainment, and maintenance among many other phases of the product lifecycle.

2. Product lifecycle management is an environment, a way of working. It is a system of methods, processes, and practices *enabled* by modern information technologies. It is not simply a collection of tools.
3. The human aspects of interpreting information, using tools, and enacting strategic decisions cannot be discounted. Given the highly configurable nature of toolsets to match corporate processes, the information author/consumer metaphor for articulating how product data moves through an organization accounts for the various human roles in designing, producing and sustaining a product.
4. The concept of product lifecycle management covers the entire product lifecycle, from requirements definition to product disposal. While early incarnations of product lifecycle management were very tool-centric and primarily addressed managing CAD files, modern product lifecycle management systems are actually a collection of tools deployed in different ways to move product data through the various groups in an enterprise and to give life to that product data over a span of decades.

"Product Lifecycle Management is the product viewpoint of the whole business. It is everything that improves the development and management of products from an enterprise and lifecycle perspective…. Without products, we have nothing to sell, and therefore have no business" (PLM Interest Group 2010, 486). The current focus of product lifecycle management is on tools and methods that capture document-based representations of physical items in digital product models that can be converted to instructions for manufacturing and assembling products.

In summary, product lifecycle management is an environment in which information technology tools and processes allow a company increased access to product definition data to better develop, manage, and support their products. It is a collection of interconnected technologies that enable companies to make better business decisions throughout the lifecycle of a product by leveraging the intelligence embedded in digital product definitions. Product lifecycle management defines and controls data collection processes, integration, transformation, analysis and visualization processes; from establishing a product's requirements, to the design, manufacturing, maintenance and recycling of the product. Figure 2 illustrates a conceptual depiction of product lifecycle management, with a key element being the business-oriented themes, which require various levels of product and process representations at multiple levels of sophistication to serve the authors and consumers of product information across the lifecycle.

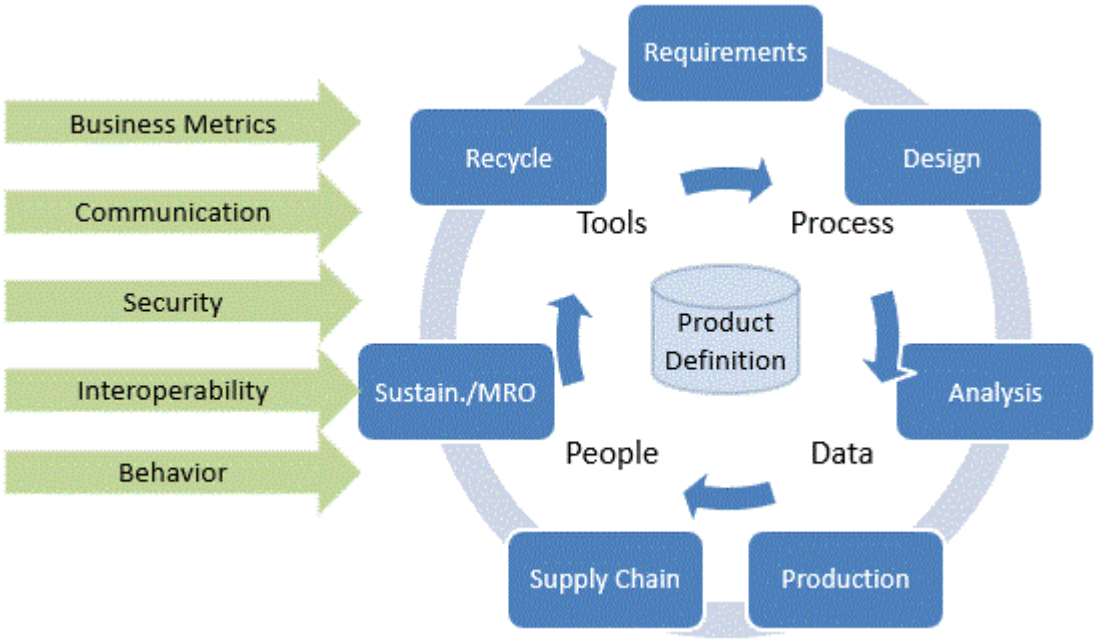

Figure 2: Conceptual model of product lifecycle management

4. Connecting the Masters of Their Domains: A Path to Salvation

Admiral Hyman Rickover, who led the US nuclear navy for over three decades, compared systems engineers and their modes of operating to a mythical character and described the way out of being caught up in a mode that only deals in abstractions.

> In Greek mythology, Antaeus was a giant who was strong as long as he had contact with the earth. When he was lifted from the earth he lost strength. So it is with engineers. They must not become isolated from the real world…The Devil is in the details, but so is salvation. (Theodore 1995, 195)

When designing and producing products, humans use various forms of representations to depict the nature of the product at any given time within the lifecycle (Buciarelli 1994, Hartman 2004, Hartman, 2005, Henderson 1999; Collins 1987, Keller and Keller 1996). Historically, artists, scientists, inventors, and others have used drawings made in many levels of detail to depict their ideas. These drawings would use different styles of projection and artistic rendering to describe the shape, function, and behavior of a product. During the stages where systems engineers define requirements or when conceptual designers select materials, product representations are rather abstract as described previously. However, the realm of the product designer often in concrete details expressed by mission parameters or functional requirements, finished machining processes and tolerances, or manufacturability or supportability concerns. In contemporary design and production environments, designers are creating digital representations to describe their products to both humans and machines at various stages of the lifecycle. While those representations were often done as 2D drawings in the past, modern design environments yield 3D models as a design artifact. Companies are now leveraging the embedded intellectual property, the communicative power, and the investment of time in the creation of the 3D models during the development stages of the lifecycle by using their derivatives to drive downstream process. Therein lies one of the primary differences between product lifecycle management and systems engineering. Systems engineering uses various abstractions of a product and its elemental components to

define a system in terms of its required input, outputs, and functions, and its components; to *predict* future performance of the built product; and to *measure value* to clarify the trade-offs within and between the choices for requirements and components (Buede 2009, 131). Product lifecycle management uses representations that are more detailed as a mechanism for *communication* to support manufacturing, operations, and product sustainment. Figure 2 shows an adaptation from Grieves (2011) of what is becoming known in the product lifecycle management community as a model-based definition (MBD).

Currently, much of the focus around the concept and creation of a MBD is targeted at replacing traditional 2D drawings (Figure 3). Instead of using orthographic projection with associated dimensions, tolerances, and annotations, model-based definitions use CAD models with associated annotations and symbols to define the *shape* of the product. However, even in current practice, little if any attention is given to the *behavior* and *context* elements of a model-based definition. Model-based definitions are often used primarily as a proxy for drawings in defining what a finished object should look like and how to inspect it to see if it has been made correctly. The *shape* definition elements of a model-based definition include items such as geometry, dimensional and geometric constraints, or tolerancing information, but are often sterile, lacking explicit information that can be used further on in the product lifecycle. The *behavioral* elements of an MBD include material models, machining process models, additive manufacturing material and process models, supply chain capability and capacity. The *contextual* elements of a model-based definition include in-situ data regarding operating performance, production factory performance, user/operator conditions. However, these capabilities do not exist in what is often a monolithic representation within modern CAD tools, and therein lies the crux of current discussion – how do we link product lifecycle management and systems engineering? One of the keys to doing this is to be able to include the behavioral and contextual elements of a 'model' and their relationship to the systems depicted in a typical systems engineering 'model'.

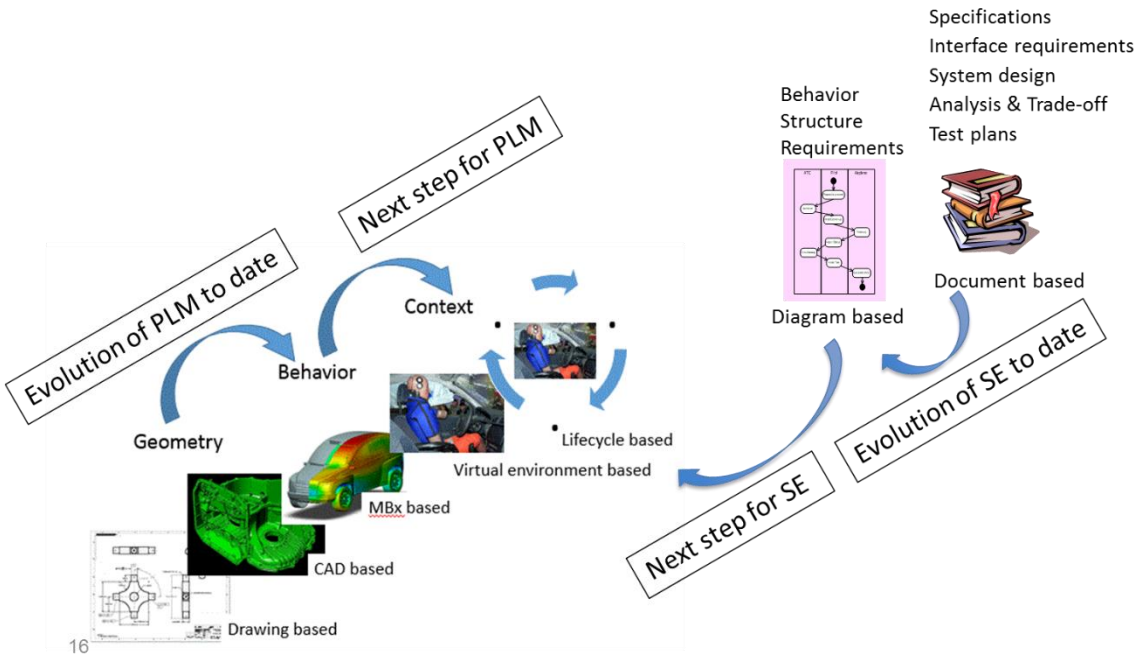

Figure 3: The evolution of the model-based definition in PLM and SE

The evolution of many products from purely mechanical, to electromechanical, to software-driven electromechanical has required a level of preliminary analysis and prediction

not seen in previous decades. SE tools are typically good at the latter items, while product lifecycle management tools are not, and vice versa. Understanding how these elements of a product will interact, and how one manages the development/production/support data from these elements, are vexing issues. As the nature of product design and production have evolved due to the proliferation of digital tools and data, the nature of product definition information has evolved as well. Table 2 provides a brief comparison of systems engineering and product lifecycle management along several contemporary factors which influence the development of complex products.

Table 2. Comparison of SE and PLM on Issues of Product Complexity

	Systems Engineering	**Product Lifecycle Management**
Level of Representation	Abstract	Concrete
Information Type	Models that depict subsystem interaction	Models that capture geometric, dimensional, and behavioral constraints
Process	Validation to stakeholder requirements	Physical production matches virtual definition
Infrastructure	Abstract models and schematics to depict performance	Interconnected design, supply chain, and sustainment using digital product model
Lifecycle Stage	Early	Later
Organizational Role	Systems engineers	Everyone else
Regulatory Compliance	Validation that the system functions correctly	Validation that the components are made correctly

Systems engineers use models to define system functionality, define the options for the physical components of the system (in this instance, hardware and software are treated as physical components, allocate functionality to physical components, model system dynamics and execution, and predict system performance and resource utilization, as shown in Figure 4. The operational concept and functional architecture are abstract entities according to Lewis (1986), who stated, "abstract entities have no spatiotemporal location; they do not enter into causal interaction; they are never indiscernible one from another." The physical architecture and allocated architecture are abstract also, as they are models of the actual physical entities that comprise the system. MBSE information is actionable in the sense that the descriptions of desired behavior, structure, requirements are adequate for designers to take action. Snelders and Schoormans (2004) discuss how this notion of actionability for products of systems engineering is optimistic at best:

> The actionability of product attributes is defined in purely pragmatic terms, with a focus on new product development: ''By 'actionable' we mean that the attributes indicate specific actions the manufacturer must take to build such a product'' (Shocker & Srinivasan, 1974, p. 922). At the same time, Shocker and Srinivasan are optimistic, in that even ''vague attributes (such as 'sportiness') may still prove sufficiently actionable for the framework to provide guidance to manufacturers''

In addition an optimistic interpretation of the actionability of systems engineering products, the descriptive models produced using MBSE are not used by designers and are not even

used by systems engineers who perform system validation tasks using system simulations. The information produced by MBSE models is not executable and is linked with simulation tools in an *ad hoc* manner, if at all.

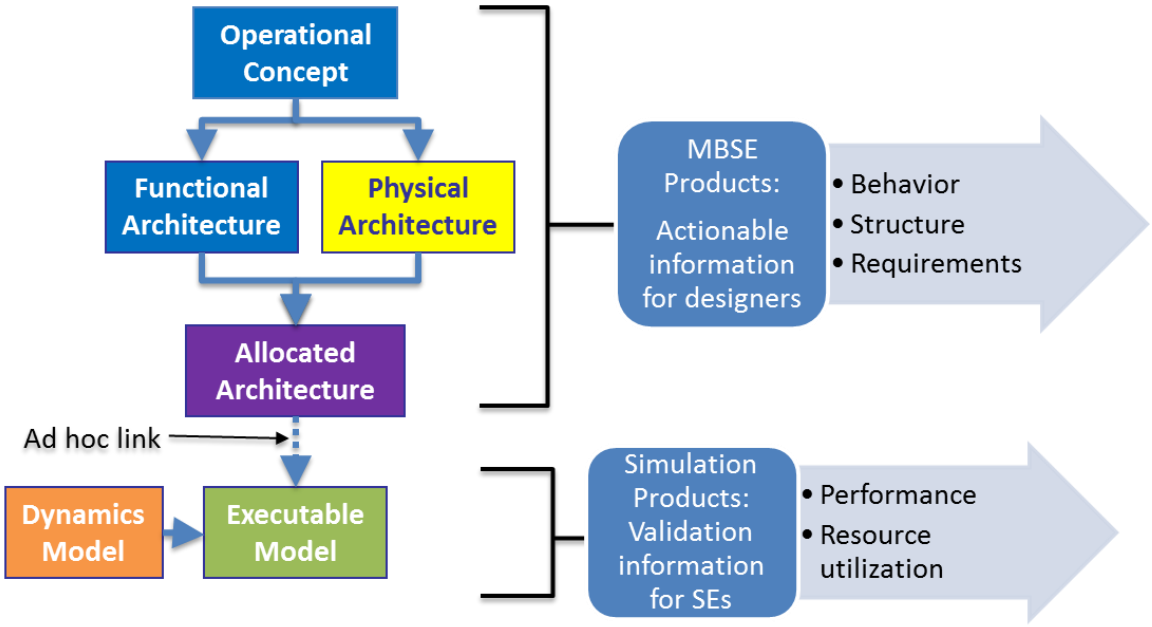

Figure 4: Current State of MBSE and System Simulation (adapted from Levis and Wagenhals. (2000))

Modern product lifecycle management tools struggle in defining, managing, and propagating many items that are quite apparent in the top-down Vee-model representation of the project lifecycle (Figure 5). The following list are common items that are often confused by the differences between different functional groups in an organization and their mandate (design/optimize vs. design/make):

- Requirements analysis and requirements management
- Description of the functions and modeling of the system architecture
- Breaking down the functions for distribution among the individual specialist design disciplines
- Synchronization of component design and development
- Simulation of all parts of the system
- Validation of subsystems and components against requirements

The Vee-Model is used routinely in the aerospace community, as well as other areas where product complexity is high, to show the interaction between the lifecycle phases, and the verification/validation loop between requirements and production.

The abstract world of SE and the tangible world of PLM do not share a common authoring environment for product definition, and they do not share a common language or ontology for defining the elements of those products either. As such, authors and consumers of product definition cannot move information fluidly between the levels of product abstraction going down and coming up the Vee

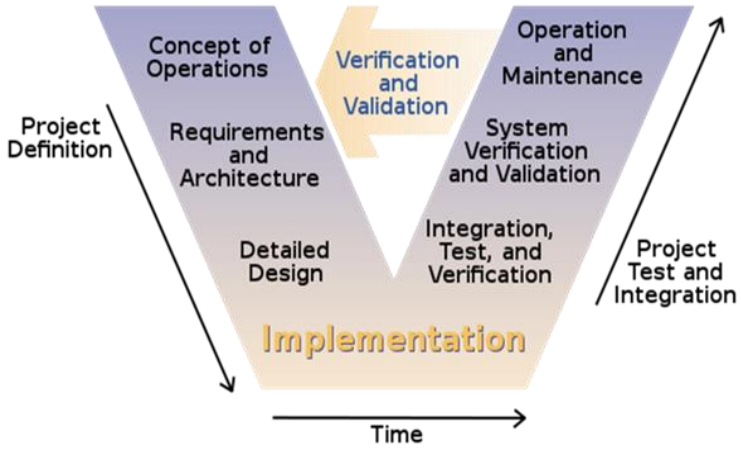

Figure 5: Vee-Model for the Product Lifecycle (from Hartman, Rosche, and Fischer (2012))

As indicated in the three examples of the virtual rollout of the Boeing 787 Dreamliner, model-based systems engineering at Ford Motor Company, and customers composing custom designs of Scania trucks, systems engineers were not the perceived or actual agents for introducing these processes. All three examples are located in the Vee-Model at the bottom and along right-hand side of the Vee-Model. Herein lies a key source of the current condition of systems engineering and its need for salvation.

The systems engineers who are engaged early in the lifecycle drive paperwork down the left-hand side of the Vee and develop initial verification and validation plans; however, they are rarely engaged in a meaningful way in the final planning or the execution of the integration and test activities coming up the right-hand side of the Vee. If we carefully inspect the Vee-Model in Figure 5, we see that during verification, the testers "feed the beast" in the systems engineering department with reports, as indicated by the arrow labeled "Verification and Validation," but that useful initial inputs and feedback from systems engineering that would be indicated by arrows from left to right are missing. It is very likely that testers perceive systems engineers to be mere clerks who check the box that the testers have turned in the required reports that systems engineers specified when coming down the left-hand side of the Vee. In addition, the models and rationale that systems engineers used earlier to define requirements and test cases may not be available to the integration and test team or, if they are available, in a form that is useful to them. Having completed their required reports, the integration and test team moves on to solve the day-to-day problems that they face in assembling a product that works and is acceptable to stakeholders.

Figure 6 represents the version of what should be occurring coming up the right-hand side of the Vee during verification and validation. Ideally, systems engineers would engage with designers, integrators, assemblers, and testers interactively and dynamically to ensure that the product is built right and the right product is built. The product lifecycle management environment should allow for the capture and representation of knowledge that systems engineers develop coming down the left-hand side of the Vee and provide efficient and meaningful approaches to updating this knowledge based on the details that are discovered during verification and validation. The outcome of the effort will support updating the systems engineering models to predict the test results more accurately. Instead of being clerks

who check the box as reports are turned in, systems engineers can engage their intellectual abilities in solving the concrete, yet complex problems that occur in the integration and test phase of activity. This combination of model-based systems engineering and product lifecycle management can ultimately lead to an outcome described by Buede (2009, 359), but rarely implemented in practice: "As confidence in a specific model increases, the model can be used to replace some of the instrumented tests and demonstrations." However, a key enabler to this scenario is a model-based product definition that includes not only shape, but behavior and context elements as well.

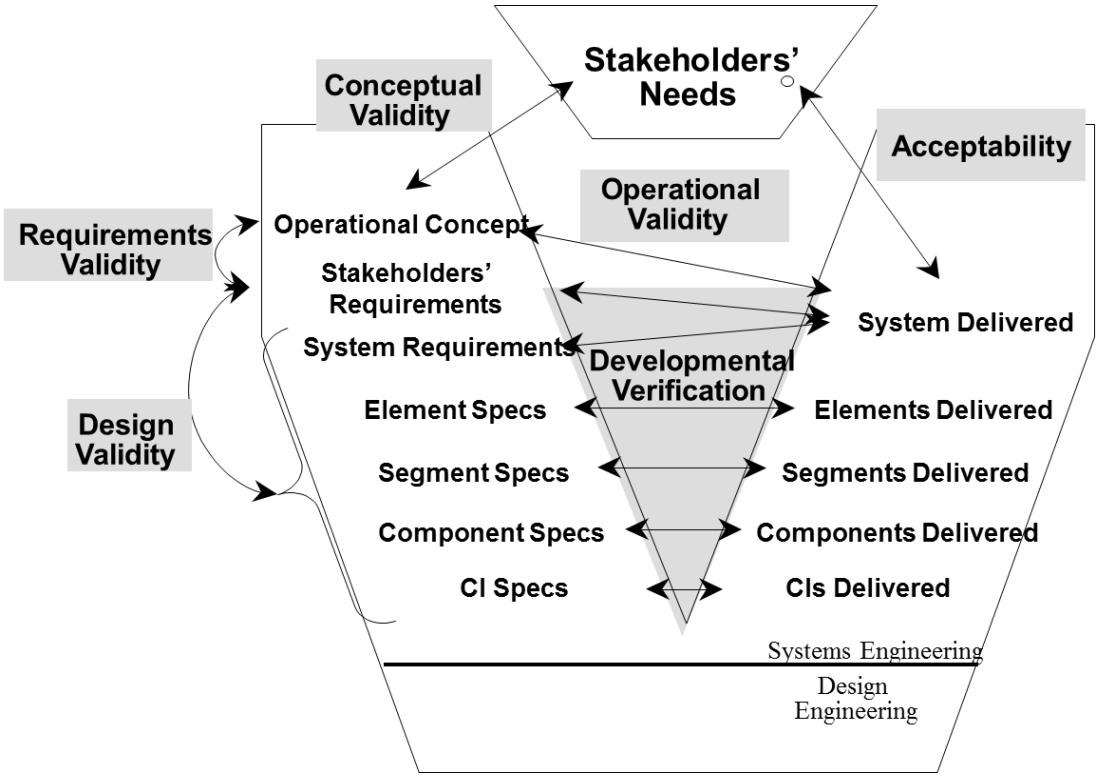

Figure 6: Verification, Validation, and Acceptance [from (Buede 2009)]

Integrating MSBSE tools into the PLM environment can also support designers as well. The benefit of integrating product lifecycle management and systems engineering applies to the top-down, requirements-driven environments that gave rise to the Vee model, which are becoming rare as the operational environment of system is changing more rapidly and requirements cannot remain static for long. It also applies to environments where customers compose made-to-order system designs or where owners of different systems decide to compose a system of systems from existing systems to achieve previously unanticipated objectives. The systems engineering models used coming down the left-hand side of the Vee generally are not developed with the benefit of detailed knowledge of actual design solutions. The product lifecycle management environment is able to provide a medium of communication (i.e., a tangible model-based definition with behavioral and contextual elements contained) that allows the composers of these new creations that use existing design solutions and systems engineers to have meaningful interactions. In addition, big-data analytics can be used on information captured in PLM to generate suggestions for existing design solutions to meet current requirements. From these interactions will come an understanding of the impacts of the compositional choices on performance, interfaces, and the value delivered, which will increase confidence that the composition will function rightly

and perform the right functions. During the integration and test phase coming up the right-hand side of the Vee, access to MBSE models and simulations will assist designers in discovering and validating proposed design changes to mitigate failures or shortfalls that were discovered during testing.

Within a typical product data environment, the definition of the elements of a product often use abstractions that are able to provide the *behavioral* and *contextual* elements of the model-based definition that is emerging as the conduit for communication within the product lifecycle management environment. The communication inputs and outputs of the various actors in the product lifecycle require a very flexible, yet robust digital product representation that can be translated and disseminated with high degrees of fidelity to the original definition. While many commercial software vendors, standards developers, and advocacy bodies lobby for one CAD-derivative format over another, there exists a framework for selecting appropriate derivative data formats (Hartman, Rosche, and Fischer 2012). The framework does account for the notion of translation between various product data authoring and consuming software tools. Yet, it does not address translation at an operational level nor does it propose solutions for addressing the inevitable translation, validation, and curation challenges that arise when trying to move semantically rich product data from one tool to another. The use of ontologies to manage this process seems promising (McKenzie-Veal Hartman, and Springer 2010; Chaparala, Hartman, and Springer 2012); however, they tend to only accommodate the shape definition elements contained in CAD and CAD-derivative data today. If ontologies are to be useful in helping bridge the gap between systems engineering and product lifecycle management, they must also capture behavioral and contextual information with high degrees of fidelity. We must also be able to articulate system-level ontologies, such as “airplane” or “automobile,” that are able to characterize the actions of the entire system. Sub-level ontologies would then be applied, such as “engine” or “controls,” with each successive layer of product (system) complexity broken down into its constituent elements. Finally, the identification and mapping of the larger product ontological schema will begin to drive a convergence of the toolsets used within product lifecycle management and systems engineering. Not only will the tools evolve to capture the shape, behavior, and context of specific components, but they will also begin to capture the interactive verification and validation process across the Vee-Model. More precise models of the physics of materials and machine behaviors, along with better process characterization, will lead to better integration between the functional areas that contribute to the design and verification of complex products. It also will enable systems engineers to have more tangible evaluation of compositional choices and more impactful participation in integration and testing. With many of the costs and factors that impact the production, supply, and sustainment of modern complex systems determined by the decisions made during the design process, this research agenda becomes all the more critical.

REFERENCES

Buciarelli, L.L. 1994. *Designing engineers.* Cambridge, MA: The MIT Press.

Buede, Dennis M. 2009. *The engineering design of systems: models and methods*. Hoboken, US-NJ: Wiley.

Chaparala, R., NW. Hartman, and J.S. Springer. 2012. Examining CAD interoperability through the use of ontologies. *Proceedings of the Computer-aided Design and Applications Conference, Niagara Falls, Canada, June 11-14, 2012.*

CIMdata. 2015. *Product lifecycle management definition.* Retrieved from https://www.cimdata.com/en/resources/about-plm on March 25, 2015.

Collins, H.M. 1987. *Expert systems and the science of knowledge.* In W.E. Bijker, T.P. Hughes, and T.J. Pinch (eds.). The social construction of technological systems: New directions in the sociology and history of technology. Cambridge, MA: The MIT Press, p. 329 - 348.

Grieves, M. 2005. *Product Lifecycle Management: Driving the Next Generation of Lean Thinking*. New York: McGraw-Hill.

Grieves, M. 2011. *Virtually Perfect: Driving Innovative and Lean Products through Product Lifecycle Management.* Cocoa Beach, Florida: Space Coast Press.

Hartman, N.W. 2004. The development of expertise in the use of constraint-based CAD tools: Examining practicing professionals. *The Engineering Design Graphics Journal, 68* (2), 14-25.

Hartman, N.W. 2005. Defining expertise in the use of constraint-based CAD tools by examining practicing professionals. *The Engineering Design Graphics Journal, 69* (1), 6-15.

Hartman, N.W., P. Rosche, and K.L. Fischer. 2012. Examining the role of collaborative formats in product lifecycle workflows. *Proceedings of the 9th International Product Lifecycle Management Conference, Montreal, Canada, July 9-12, 2012.*

Henderson, K. 1999. *On line and on paper: Visual representations, visual culture, and computer graphics in engineering design.* Cambridge, MA: The MIT Press.

INCOSE. 2007. *INCOSE Systems Engineering Vision 2020*. San Diego, US-CA: International Council on Systems Engineering.

INCOSE. 2014. *A World in Motion: Systems Engineering Vision 2025*. San Diego, US-CA: International Council on Systems Engineering.

Keller, C.M. and J.D. Keller. 1996. *Cognition and tools use: The blacksmith at work.* Cambridge, UK: Cambridge University Press.

Levis, Alexander H., and Lee W. Wagenhals. 2000. "C4ISR architectures: I. Developing a process for C4ISR architecture design." *Systems Engineering* no. 3 (4):225-247.

McKenzie-Veal, D., N.W. Hartman, and J.S. Springer. 2010. Implementing ontology-based information sharing in product lifecycle management. *Presented at the Engineering Design Graphics Division of The American Society of Engineering Education 65th Annual Midyear Meeting, Houghton, MI, October 3-6, 2010.*

PLM Interest Group. 2010. Product Lifecycle Management Journal no. 37.

Rockwell, Theodore. 1995. *The Rickover effect: the inside story of how Adm. Hyman Rickover built the nuclear Navy*. New York: Wiley.

Snelders, Dirk, and Jan P. L. Schoormans. 2004. "An exploratory study of the relation between concrete and abstract product attributes." *Journal of Economic Psychology* no. 25 (6):803-820. doi: http://dx.doi.org/10.1016/j.joep.2003.08.004.

Stark, J. 2011. *Product Lifecycle Management: 21st Century Paradigm for Product Realisation*. London: Springer-Verlag.

Visser, Willemien. 2004. *Dynamic aspects of design cognition: Elements for a cognitive model of design*. Paris, FR: INRIA.

Wymore, A Wayne. 1993. *Model-based systems engineering: an introduction to the mathematical theory of discrete systems and to the tricotyledon theory of system design*. Boca Raton, US-FL: CRC press.

Biographies

Nathan Hartman is a Professor in the Department of Computer Graphics and Director of the Purdue University Product Lifecycle Management (PLM) Center of Excellence. He teaches undergraduate courses in 3D modeling and model-based definition, interoperability and data exchange standards, and product data management. He also teaches graduate courses in PLM, research methods, and measurement and evaluation. His research focuses on the use of 3D CAD tools and digital product definition within the product lifecycle, model-based enterprise, long-term archival of CAD data, and 3D data interoperability and exchange. He holds a Bachelor of Science in Technical Graphics and a Master of Science in Technology from Purdue University, and a Doctor of Education in Technology Education from North Carolina State University and has held industry positions with Fairfield Manufacturing Company, Caterpillar, and RAND Worldwide.

C. Robert Kenley is an Associate Professor of Engineering Practice in Purdue's School of Industrial Engineering in West Lafayette, IN. He teaches graduate courses in systems engineering at Purdue and has over thirty years' experience in industry, academia, and government as a practitioner, consultant, and researcher in systems engineering. He has published papers on systems requirements, technology readiness assessment and forecasting, Bayes nets, applied meteorology, the impacts of nuclear power plants on employment, and agent-based simulation and model-based systems engineering for systems of systems. Professor Hartman holds a Bachelor of Science in Management from MIT, a Master of Science in Statistics from Purdue University, and a Doctor of Philosophy in Engineering-Economic Systems from Stanford University.