Virtually Indistinguishable

Michael W. Grieves

Virtually Indistinguishable:
Systems Engineering and PLM*

Michael W. Grieves [1,2]

[1]NASA Marshall Space Flight Center (MSFC), Huntsville, AL USA
mgrieves@ameritech.net

[2]Florida Institute of Technology, Melbourne, FL USA

Abstract. This paper proposes that Systems Engineering and Product Lifecycle Management are closely related. In fact, the contention is that one of the threads that has led to the formation of Product Lifecycle Management is Systems Engineering. Product Lifecycle Management extends Systems Engineering and uses many of its methodologies and processes. However, this connection between these two areas has not gotten the proper attention from either industry or academia. Cross-pollinating Product Lifecycle Management and Systems Engineering would benefit both disciplines.. The paper is based on the author's observation and work at large organizations with significant Systems Engineering disciplines, such as NASA, the United States' Department of Defense, Boeing Corporation, Lockheed Martin, and others. The discrepancy between Systems Engineering claims of being involved in the entire product lifecycle versus the reality of Systems Engineering ending its involvement with the product at requirement verification informs the perspective of this paper.

Keywords: Product Lifecycle Management, PLM, Systems Engineering, Model-based Engineering, MBSE, Product Development

1 Introduction

Systems Engineering (SE) and Product Lifecycle Management (PLM), as will be argued in this paper, have a great deal in common. What serves to distinguish PLM from SE is the utilization of virtual products and virtual environments. However, System Engineering, as a concept, substantially overlaps (or in keeping to the thematic thread of my recent book is "virtually indistinguishable" from) PLM in a great many ways. This paper is a preliminary effort to connect PLM with SE and to explore what they have in common and where they differ. Both PLM and SE should substantially benefit from this connection.

* This paper is derived from a chapter of the same name in Grieves, M. (2011). *Virtually Perfect : Driving innovative and lean products through Product Lifecycle Management*. Cocoa Beach, FL: Space Coast Press.

Systems Engineering, as a discipline, has been in existence for at least forty years, with some dating its genesis back to World War II. Others date it to the advent of the Space Program. However, Systems Engineering has a much stronger presence in industry than in academia. The most likely explanation is that Systems Engineering, by its multi-discipline nature is not well suited to the narrower, single-discipline orientation of academia. There are recent, although hardly widespread, appearances of Systems Engineering departments.

The literature has tended to be comprehensive explanations of Systems Engineering, starting with Wymore [1] and continuing with recent offerings [2-3]. The International Council on Systems Engineering (INCOSE) has supported a Systems Engineering journal. Industry publications have tended to be in the form of handbooks, with NASA's Handbook of Systems Engineering [4], as a prominent example.

While not originating until literally the 21st century, Product Lifecycle Management has an even less body of work. One of the first papers defining PLM as a new paradigm was my 2005 paper [5]. The two seminal books defining and explaining PLM are generally considered to be Stark [6] and Grieves [7]. The advances in PLM have been primarily driven by PLM software providers and their industry users, with the literature confined primarily to conference papers. Again, the lack of take-up in the academic community can be attributed to PLM's multi-domain and multi-phase orientation.

With the paucity of literature on both these disciplines, it is unsurprising that there are few papers linking Systems Engineering with PLM, but none proposing that PLM and Systems Engineering are closely related.

2 PLM and Systems Engineering

From a review of the literature, Product Lifecycle Management is literally a 21st century concept. However, PLM did not burst on the scene fully formed. Instead, it had its technological predecessors. I had proposed that the threads that made up Product Lifecycle Management were: Computer-Aided Design (CAD), Engineering Data Management (EDM), Product Data Management (PDM), and Computer Integrated Manufacturing (CIM) [7, pgs. 45-54] In retrospect, I realize that this perspective was one of an information systems perspective and that that there was a major conceptual thread that I had ignored. That conceptual thread is Systems Engineering (SE).

Having minored in Systems Engineering as an undergraduate, I have long felt that there was commonality between Systems Engineering and Product Lifecycle Management. My work with NASA, where the scale and complexity of systems do not come much bigger, allowed me to think through the connections between SE and PLM [8]. I suspect that Systems Engineering and PLM will get closer over time as PLM begins to manage product information that was previously "managed" as documents. A major example of this is the requirements of a system whose management consisted of engineers keeping in mind a series of "shall" statements contained in a systems requirements document.

At the risk of overreaching, I will, in this paper, make a fairly good argument that PLM extends Systems Engineering and that PLM needs to incorporate Systems Engineering as a major methodological component. PLM adds a dimension of virtuality to Systems Engineering in order to trade off information for wasted physical resources.

Systems theory is mathematically based [9]. Systems Engineering is a rich and robust concept, with mathematical underpinnings. It has been crucial in organizing major engineering projects. The description of it here will be by necessity cursory, keying in on elements that will be relevant to the discussion of Product Lifecycle Management. Rather than getting buried in the details of Systems Engineering, the purpose here is to focus on the major aspects that will highlight the linkages to Product Lifecycle Management.

2.1 Defining Systems

The idea of systems is that they are not monolithic things, but consist of components that are connected together to produce results that the individual components could not produce by themselves. This means that there is more than one component within the system rectangle that operates on the inputs and that are linked together. One such representation is shown in Figure 1.

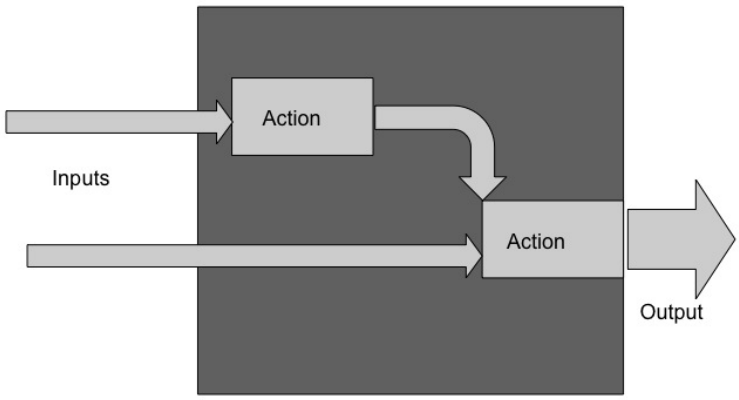

Fig. 1. Systems Definition Model

There are numerous definitions of systems, both tangible and intangible. I would like to propose the following definition of a system for the purposes of this discussion on Systems Engineering and PLM:

A system is two or more components that combine together to produce one or more results from one or more inputs.

Products such as airplanes, power plants, and medical scanners fulfill this definition. However, these are systems within a system. Airplanes are in themselves fantastic systems, but they are also part of a larger system that consists of airports, air traffic control, refueling trucks, and other systems that make up our air transport system.

PLM originally dealt with only the isolated product. PLM was concerned with the makeup and changes in the product itself. The mental view of PLM was of a series of the isolated product suspended in virtual space, reflecting the changes as the product progressed throughout its life. However, as PLM rapidly evolved, it began concerning itself with simulating the product in its environment. PLM now can be said to involve the interaction of the product and the environment that the product operates in.

This makes PLM not only about the isolated product. It makes PLM also about the environment it finds itself in. PLM concerns itself with how inputs affect the product and what effects the product has on its environment. In other words, PLM is about product systems.

2.2 Defining Product Lifecycle Management and Systems Engineering

My definition of Product Lifecycle Management is:

Product Lifecycle Management (PLM) is an integrated, information-driven approach comprised of people, processes/practices, and technology to all aspects of a product's life and its environment, from its design through manufacture, deployment and maintenance—culminating in the product's removal from service and final disposal.

This definition, while slightly changed over time, goes back to 2003 and first appeared in an early executive article on PLM [10]. This definition of PLM and derivations of it, both attributed and unattributed, are widely used. The definition emphasizes a product-focused perspective throughout the product lifecycle rather than a functional area perspective, e.g., engineering, manufacturing, etc.

The NASA Systems Engineering Handbook [4] defines Systems Engineering as "a methodical, disciplined approach for the design, realization, technical management, operations, and retirement of the system." This definition has a great deal in common with the definition of Product Lifecycle Management. These two definitions move even closer when I update my definition of Product Lifecycle Management, as I did in my recent book [11, pg. 28] that PLM is involved not only with the product but also with its environment. A product and its environment can certainly classify as a system.

The NASA handbook goes on to describe a system as a collection of different elements that deliver functionality that could not be gotten from the individual elements by themselves. Much like Product Lifecycle Management, Systems Engineering also views that the system is not simply the physical product or artifact, but that it is also composed of the surrounding environmental aspects including people, facilities, and processes that produce the functionality that the system intends to deliver.

Systems Engineering looks at the parts of a system in a holistic, integrative way as opposed to looking at each of the parts individually. Systems Engineering primarily concerns itself with two major elements. First, Systems Engineering concerns itself with the system that needs to be created to provide a desired functionality. Second, Systems Engineering concerns itself with the development lifecycle phases that enable the creation of this system. While Systems Engineering professes to be concerned with the entire lifecycle of the system, the reality is that it primarily focuses on the development phase of the system.

Systems Engineering starts with a view that a system needs to produce certain functionality. As shown in Figure 1, the view of a system is that of having inputs (arrows from the left), actions (rectangles), and an output (arrow on the right). The output is the functionality of the system, and it is defined as the requirement that the system must meet. The task of Systems Engineering is to determine what inputs are available and what types of devices it can create that will use those inputs in order to produce the actions that will create the required output.

For example, if we want to move an airframe through the air, we have earth's atmosphere as an input. Therefore, we can drive this airplane with either a propeller that creates the action of pulling the airplane through the air or a jet engine that creates the action of pushing the airplane through the air. However, if we want to travel in outer space, we no longer have air as an input, so therefore we have to create a different kind of mechanism, a rocket, in order to create the action that we need.

The premise behind Systems Engineering is that a system can be decomposed into its components and that these components have to interact interdependently in order to produce the requirements that we desire. Systems Engineering decompose the overall system into smaller and smaller units until it gets to individual items called parts that then must come together in order to produce the desired requirements results.

Systems Engineering therefore is about breaking down systems such as a pacemaker, automobile, or space shuttle into more and more discrete levels, and at each of these levels duplicating the system of inputs, actions, and outputs. In addition to simply cascading these system diagrams as the system components get more granular, Systems Engineering has created a number of different representations to reflect this drilling down of system behavior.

The twofold problem that Systems Engineering wrestles with is that in order to obtain specific individual requirements the system has to be decomposed into finer and finer components. At the same time these components have to be coordinated with each

other in order to produce the overall system behavior that the Systems Engineering people desire.

2.3 Requirements to Verification and Validation

Systems Engineering begins with a list of requirements that the system must meet. These requirements are generally expressed as a series of "shall" statements, as in "the system shall exhibit this particular behavior." Matching up to these "shall" statements are validations and verifications [12] that sufficiently demonstrate that the designed system meets these requirements.

This issue always is that the only way to be completely comfortable that the requirements of the system are really met is to observe how the system performs throughout its entire life. That, obviously, is not feasible, so preparing validations and verifications that will test for the behaviors that are the outcome of our requirements is our proxy for observing the system over its life.

In order to adequately define and perform these validation and verification tests, the requirements must be defined with enough specificity to judge whether or not the system met its requirements. Requirements that an automobile be safe or an airplane be fuel efficient is inadequate. There must be measurable specificity to requirements.

There are a number of common issues that arise. The major issue is that the verifications and validations are either not appropriate or not exhaustive enough to demonstrate that the actual system will meet its requirements in use. That has been a flaw in classical Systems Engineering. While Systems Engineering brought a much more systematic approach to design and engineering, the document-based nature of SE meant that verification and validation was often confined to the usual suspects and siloed within sub-systems. As discussed below, PLM can make this richer and much more robust through simulation.

In addition, the lessons learned from the performance of the system in actual use were not fed back into the requirements and/or validations and verifications. As a result, system engineers made preventable errors over and over again, wrongly believing that their requirements translated into actual performance.

As a final note, there are constraints on requirements that must be taken into consideration. It is rare that system engineers have a completely blank sheet of paper on which they can start to set out their requirements. System engineers are generally constrained in the design phase by the technology, subsystem, or platform they select or are required to use. The manufacturing systems that they have available may constrain what they can actually build. The support network may constrain how the product can be sustained. This is not necessarily bad in that there is a history that the requirements can be met with the corresponding verification and validation tests.

3 Models, System Engineering, and PLM

The term model has at least two meanings that are in play with Systems Engineering and PLM. A model is an abstract representation that helps us visualize and understand something that we cannot directly observe, such as a process for doing product design or something we want to distill down to key aspects, such as a product's behavior. These are two types of logical, abstract models that we will look at: things that we do to realize products, process models, and things that products do, behavior models. A model can also mean a mathematical representation of a physical object, such as a CAD model. This type of model is a spatial model. Systems Engineering uses models in the first way, while PLM uses the word in both meanings.

Given the era that Systems Engineering was developed in, it did not require a reliance on computing technology in order to enable it. Thus, Systems Engineering models are of the atom-based, two-dimensional, static kind. More informally, Systems Engineering models consist of documents, blueprints, and flow charts. When Systems Engineering requires three-dimensional, dynamic models, it relies on building physical prototypes.

3.1 Process Models

Figure 2 shows the different representations, which are referred to as a Waterfall representation, Vee representation, and a Spiral representation. These representations are all reflective about the idea that the system is deconstructed into ever more discreet components until it reaches the most basic component which is then designed. These basic components are then assembled up the hierarchy until they become the entire system.

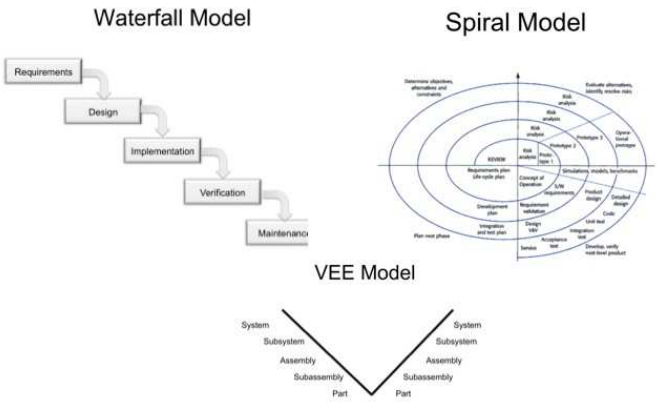

Fig. 2. Systems Definition Model

There are both similarities and differences between the Systems Engineering model and the Product Lifecycle Management models. On the similarity side, the Waterfall model of Systems Engineering does deal with the entire product lifecycle, even though the Vee model and the Spiral model do not. While the PLM model, as shown in Figure 3, does deal with the product development phase, the Vee model and the Spiral model deal more extensively with product development, as illustrated by the activities their models describe.

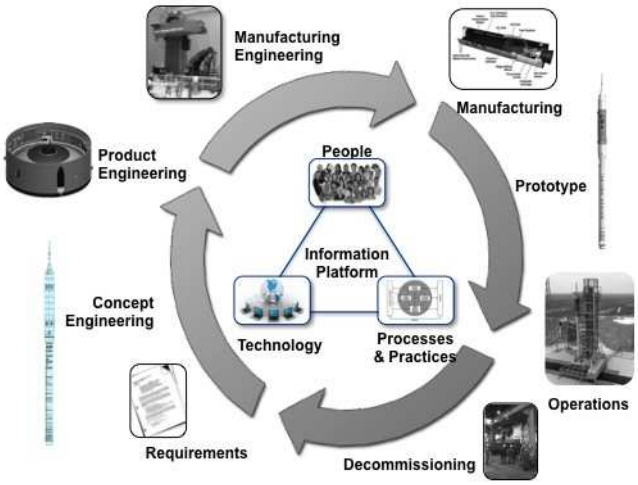

Fig. 3. PLM Model – Source NASA MSFC

The primary differences in models are that the Systems Engineering models are sequential models. Even though this is not what usually happens in Systems Engineering, the conceptual premise of the models is that things happen in sequence during Systems Engineering. In the Spiral model the sequential nature of the model still exists, but with an iterative nature to it.

The PLM model is an integrated model. The functional areas are arrayed around an inner core of information about the product that the functional areas either populate or consume. There is also a natural sequence to the create, build, support, and dispose nature of the product lifecycle. However, also implicit in the circular arrangement is the idea that functions can occur concurrently with each other as the information becomes available. Manufacturing, for example, can begin to design its build processes as the design of the product takes shape.

Systems Engineering, in spite of what might be its actual messy practices, has a logical, sequential flow about it models. Product Lifecycle Management, on the other hand, has a concurrency in terms of activities that take place surrounding the product.

3.2 Behavior and Spatial Models

I have elected to cover the behavioral and spatial models together. The reason for this is that the function and form of the product are closely interrelated. As the old adage goes, "form follows function."

Figure 4 divides spatial form information into two categories two-dimensional and three-dimensional. It also divides behavioral information into two aspects, static and dynamic. If we lay these out in a matrix, we can see where both Systems Engineering and Product Lifecycle Management fall in dealing with both form and functional product information.

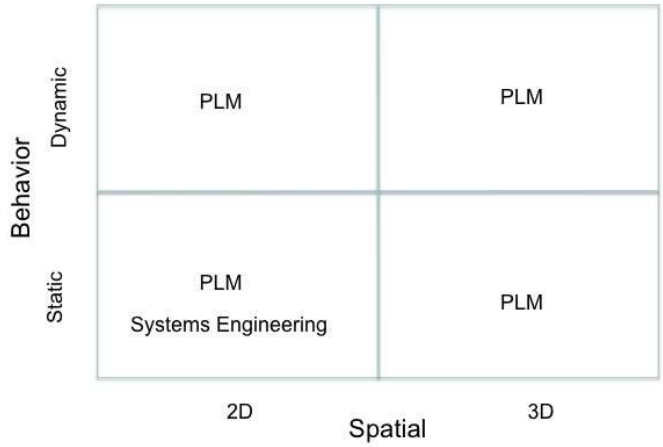

Fig. 4. Systems Information Behavior and Representation

If we look at the quadrant for two-dimensional product information and static behavioral analysis, we see that both Systems Engineering and PLM deal with this information. This information is primarily document based. Two-dimensional spatial information is realized as drawings. Static behavioral information can be thought of as documents that deal with the results of analysis. The behavioral aspects of the product in static mode would be set out in requirements of formulas, analyses, and test results as captured in documents.

The line between static and dynamic behavior can be a little fuzzy. The analysis of a change in material deformation versus pressure applied over time would be dynamic in the sense that it would show changes over time. However, the Systems Engineering

has typically dealt with this analysis as a series of computed points. PLM extends this with a continuous simulation that could stop and examine any point in time or any interval. Systems Engineering's analysis would be classified as static, while PLM's analysis is dynamic. Systems Engineering can be thought of dealing with a series of snapshots, while PLM deals with both snapshots and film.

Some in Systems Engineering will argue that they are now dealing with dynamic simulations. I would agree that this is the case since the advent of computers. However, Systems Engineering has a methodology that does not depend on it being enabled by technology. Because of that, I categorize Systems Engineering as being concerned with static analysis and PLM as being concerned with both static and dynamic analyses.

When it comes to dealing with three-dimensional spatial information, Product Lifecycle Management concerns itself with spatial models, while Systems Engineering does not. Again this may be a matter of definitional issues, in that System Engineering techniques and methodology may be applied to evaluating three-dimensional models. However, if we think of Product Lifecycle Management as extending Systems Engineering, then the Systems Engineering methodology becomes Product Lifecycle Management methodology.

The final quadrant, three-dimensional spatial models and dynamic behavioral models, are clearly the area of Product Lifecycle Management. As has been discussed extensively in my recent work, the use of simulation and virtual models in dynamic settings sets Product Lifecycle Management apart from its predecessors. It is in this quadrant that virtual products operate in independent virtual space.

Not only can the spatial elements of the product replace the use of expensive physical prototypes for visual purposes, virtual products can be put through extensive testing and simulation in order to determine what characteristics that the physical product will have. Again Product Lifecycle Management is not something entirely new. It takes the methodology of System Engineering in order to extend static, document-based information and analysis into three-dimensional dynamic virtual simulation of products.

3.3 Model-Based Systems Engineering

This focus on the system as an abstract, logical model from a behavioral perspective developed into a sub-branch of Systems Engineering called Model-based Systems Engineering (MBSE) [13]. MBSE has as its premise that a system that meets its defined requirements can be mathematically modeled. This is done by breaking down the system into its logical components and defining the behavior of these components and their relationship to other components.

Modeling languages were then created to do just that. One such modeling languages, called SysML [14], was developed in order to provide a standard for the system modeling that Systems Engineering concerns itself with. Modeling languages for MBSE

are not simply languages like computer languages, but they also incorporate diagrams to show the flow of the system.

The system modeling language is basically a much more sophisticated version of the system models shown in Figure 1. The intent of this modeling language is to model inputs, actions, and outputs of the discrete elements of the system and link these discrete elements together in order to have an accurate, albeit abstract, view of the system.

While subsequently, computer tools have been developed in order to automate this systems method, it still remains an abstract version of the system. While this is the useful representation of the system, there generally is a major leap between this abstract version of the system and the three-dimensional parts and components that need to be created in order to produce the system behaviors. An assumption with Systems Engineering was that physical prototypes would need to be built in order to provide the rich detail that would be needed in order to actually test out the system assumptions.

3.4 PLM Models

Product Lifecycle Management, on the other hand, has much richer system representations. Given the technology that enables it, PLM not only models abstract systems behavior and characteristics, but it can also model the actual geometric representation that will provide these characteristics and behaviors. The Systems Engineering model might show a box that represents the wing control surface. This control surface has as its inputs its current angle and the pilot's command to change that angle. The output is a number that represents the new angle of the surface.

Product Lifecycle Management's simulation shows a representation of the actual wing and control surface. It will simulate in a visual fashion the actual movement of the surface as a pilot issues his commands. In addition, the simulation can incorporate external variations such as horizontal and vertical airflows in order to show how the wing surface will actually react in its natural environment. While Systems Engineering models are relatively static, Product Lifecycle Management models are dynamic.

Systems Engineering does not deal with spatial models, except as two-dimensional documents, i.e., drawings. Product Lifecycle Management does deal with spatial models. It is these two different uses of the term "models" that is often confusing. Systems Engineering in MBSE uses "models" as an abstract, logical representation. Product Lifecycle Management uses it to represent spatial models that incorporate the behavioral characteristics.

Systems Engineering produces numerical or logical representations as its reflection of systems performance. Product Lifecycle Management produces those same numerical or logical representations, but then transforms them into spatial representation of systems performance. However, because computing enables it, PLM produces its representation of systems performance in a much richer, more granular fashion. Using

the example above, Systems Engineering might give the angle of the control surface as a specific number. A PLM simulation can show the entire length of the control surface and show that some areas of the surface might have a different angle than other areas because of material deformation, weight distribution, or other factors.

Figure 4 illustrates how Systems Engineering and Product Lifecycle Management would map out in a quadrant with spatial models and behavioral models. As illustrated here, Systems Engineering provides some complexity of behavioral models and little or no complexity of spatial models. Product Lifecycle Management, on the other hand, provides fairly high levels of both visual and behavioral models. This is not to say that Product Lifecycle Management replaces Systems Engineering, but is to say that Systems Engineering is the methodology that creates the complex models that Product Lifecycle Management maintains and uses.

4 Product Information Feedback Loops

While technically systems can operate perfectly well by taking required inputs, performing routines and operations on them, and delivering an output, most systems operate with what is called a feedback loop. The feedback loop is the idea that the output is looked at with respect to a desired goal, and then the inputs and/or routines are modified in order to change the output to close the gap between what the output is producing and what is desired. Sometimes the feedback loop is internal to the system, such as the governor on a motor. At other times, the feedback mechanism will be external to the system, like a driver who looks at the speed that the automobile is traveling at and adjusts his pressure on the gas pedal or brake. In either case the output is compared against the desired output and adjustments are made to either the inputs or the routines that the system is running.

While this idea of feedback is usually implicit in systems that Systems Engineering deals with, it is also an aspect of the Systems Engineering process itself. A set of requirements is set out for the system. An evaluation or test is defined to see if the system being designed meets those requirements. If it does, then the design is approved. If it doesn't, either the system design is modified or the requirements are modified.

The issue with Systems Engineering is that the assumption is made that once the system passes these evaluation or test, then the system will actually perform in its usage so as to meet those requirements. Often times, Systems Engineering feels that they have done their job if the system passes the test, irrespective of whether the product can be actually manufactured, or if the product performs to those requirements in actual usage.

Figure 5 illustrates the difference between the Systems Engineering feedback loop and the PLM feedback loop. On left hand side, both Systems Engineering and PLM are in agreement that products are designed that meet requirements. Those designed products undergo verification and validation processes to insure that the designs indeed meet the desired requirements.

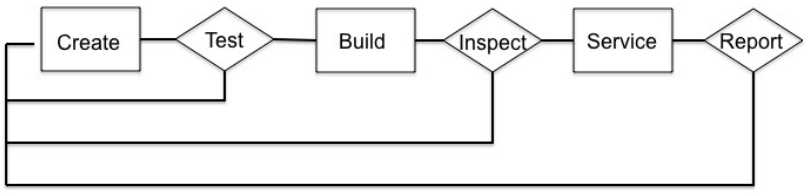

Fig. 5. PLM Feedback Loops

However, PLM goes on to do the same sort of feedback for the manufacturing and usage phases. In the manufacturing phase, the product is checked to see whether or not its physical form can meet the requirements set out for it. If it cannot, then that information is sent back to engineering in order to either adjust the requirements and/or develop a new methodology or manufacturing process in order to produce a product that meets those requirements.

In the support phase, the issue of meeting requirements goes from evaluations or tests that are a proxy for those requirements in the development phase to looking at the performance of the product and actual use in determining whether or not the product indeed met its design requirements. Again, if the product does not meet its design requirements in actual usage or if actual usage surfaces additional requirements, that information is then sent back to engineering. At that point in time, it may be too late to do much with the current product, but this should be fed back into a base of knowledge so that engineering can understand the gap between the requirements they thought could be fulfilled and what actually occurred. They then can adjust the requirements for future versions of the product.

5 Comparing Systems Engineering and PLM

There are currently some differences between Systems Engineering and PLM. Those differences are:

- Systems Engineering is product realization focused while PLM is product lifecycle focused;
- Systems Engineering is functionally based versus Product Lifecycle Management which is lifecycle based;
- Systems Engineering concerns itself primarily with physical products where PLM concern itself with both physical and virtual products;
- Systems Engineering is document based, while PLM is digital based;
- Systems Engineering is a much deeper discipline versus PLM, which is much broader.

5.1 Product development focused versus product lifecycle focused

Systems Engineering has primarily focused on the development phase of defining and creating the system or product. It breaks down the product development phase into a number of divisions, such as: Concept Studies, Concept and Technology Development, Preliminary Design and Prototyping, Testing, and Final Design. It then often lumps manufacturing, support, and disposal into an aggregate category.

The implicit assumption is that the system engineering work is pretty much done once the system is designed and meets the testing requirements. The rest of the lifecycle is simply an exercise in execution. System Engineering believes is has done the hard part of factoring the build, support, and dispose phases into its system view. The messy part of actually making those phases work is someone else's responsibility.

Systems Engineering's view of the product lifecycle is much like the famous cartoon in the New Yorker Magazine [15] that showed a detailed view of Manhattan with the farther away from Manhattan the view was, the smaller and more insignificant the world appeared to New Yorkers. With System's Engineering, the create phase is the dominant view, with the other lifecycle phases being small and insignificant.

Product Lifecycle Management takes a more balanced view of the importance of the lifecycle phases, with primacy given to the usage phase. The ability of the system design to pass the required tests is not its definition of a product's success. The ability of the product to perform to the user's expectations over the time the product is in service determines whether a product is successful or not.

This is not to invalidate the System Engineering perspective. On the contrary, the likelihood of the product to perform well in service is dependent on whether or not rigorous system engineering methodology is employed. However, the feedback loops between manufacturing and engineering and support and engineering need to occur for system engineering to have an accurate understanding of how well their system performed.

Systems Engineering is critical to getting the product lifecycle off to a strong start. However, under PLM, it needs to have more connection to the build, support, and dispose phases. It needs to collaborate in the development process more with those build and sustain phase experts in order to create a system that can actually be built and supported. Systems Engineers need feedback from those phases in order to validate that the system design actually performed to expectations.

5.2 Functional based versus lifecycle based

Systems Engineering, as even its name implies, is more about the engineering function than it is about the rest of the lifecycle. While system engineering does profess to concern itself with the rest of the product lifecycle, the build, sustain, and dispose phases, the reality is that Systems Engineering is more about getting a product that meets the defined requirements than it is about following that product through its lifecycle or to see if that it indeed fulfilled its functions in actual use.

In that respect Systems Engineering appears to be a bit siloed. The systems engineer finishes with a product that he or she deems has met the defined requirements, and then throws it over the wall to manufacturing. While Systems Engineering professes to concern itself with the support and sustainment phase, the reality is that the support and sustainment people are generally on their own when it comes to maintaining the product. If there is a major problem, the Systems Engineering staff gets dragged back in, but as a general rule they are done with the product when they hand it over to manufacturing.

Product Lifecycle Management on the other hand takes a broader view of the product. In fact, the product is not deemed to be successful unless it performs in the sustainment phase to the user's expectations. In addition, information from both the manufacturing and the sustainment phase are fed back into engineering to close the loop between what was theoretically possible with the product and what it actually did.

5.3 Physical product versus physical and virtual product

Systems Engineering generally concerns itself with the functioning of the product. It creates a set of requirements, and, if the product meets those requirements, the product is deemed to be successful. Systems Engineering does not concern itself with operations manuals, marketing collateral, repair manuals, or repair diagnostics.

Product Lifecycle Management does concern itself with these things and views them to be as important to the product as the actual functioning of the product itself. If a product can perform its requirements flawlessly, but the user does not know how to go about obtaining that functionality, then the product is not a successful or useful one.

5.4 Document-based versus Digital-based

At first glance, it might appear that the elements of PLM are people, process/practice, and technology, while Systems Engineering has as its elements only people and processes/practices. However, paper, pencil, binders, and blueprints are also technology.

The difference is that Systems Engineering is document based, while PLM is digital based. However, the difference is not simply the difference between information being represented in atoms or bits. Moving to digital technology represents a quantum jump in not only the representation of information, but in its handling.

Document-based technology is static, while digital technology is dynamic. If we wish to have a different view of a paper-based drawing, then we have to make a new drawing. If we wish a different view of a digital representation, then all we have to do is request it and the computer will rearrange the information.

Document-based technology requires that people follow the steps laid out in a process and methodology. If they choose not to follow those steps, only the active oversight of other people enforces those steps. Digital technology can enforce following process and methodology steps, both directly by requesting that people comply before allowing them to proceed or indirectly by inspecting the digital information to see if it complies with the process requirements. While people can and often do try to circumvent digital enforcement, it adds a layer of difficulty that simply ignoring documents does not have. In addition, digital oversight does not have to be active. The digital technology can simply produce alerts when there is a lack of compliance, allowing humans to take the appropriate action.

The dynamic aspect of PLM also means that product behavior can be simulated and not simply analyzed as with document-based, static information. Systems Engineering requires that products go to prototypes much earlier and at increased costs.

5.5 Deeper versus broader

Systems Engineering is more mature and has had the time to develop into a deeper concept. Systems Engineering has well developed procedures and methodologies concerning how we should go about creating products. Systems Engineering methodologies have evolved over time and have been built upon as issues have arisen and have been tackled.

Product Lifecycle Management is much broader as has been pointed out above. PLM is not only about making the product but is also about sustaining that product. PLM is not only about providing a detailed prescription in how to go about making the product. It is about what information about that product that needs to be captured, organized, and maintained.

6 Suggestions for future direction

I would like to close out this paper by making some suggestions for future direction. In addition to the obvious suggestion that there needs to be more collaboration between the people working on Systems Engineering and Product Lifecycle Management, and that the intersection between the two disciplines needs further fleshing out and investigation, I would like to make three suggestions as to the actions needed in order to meld Product Lifecycle Management with Systems Engineering. These prescriptive suggestions are:

- Organizations should deal with Systems Engineering and PLM together rather than separately,
- Systems Engineering should be engaged across entire lifecycle,
- PLM should more rigorously adopt Systems Engineering methodology.

6.1 Integrate dealings with Systems Engineering and Product Lifecycle Management

Even though Product Lifecycle Management generally starts as a design and engineering initiative, organizations deal with Product Lifecycle Management and Systems Engineering as two distinct disciplines. Systems Engineering standards and PLM processes are dealt with as two distinct things, when they should be considered from an integrated perspective. It is not uncommon to find organizations have two distinct policies and procedures for PLM and Systems Engineering, even as they overlap.

Systems Engineering spends a great deal of time and effort in developing "bluebooks". These bluebooks are the standards for addressing engineering practices and processes. Far too often, these bluebooks are developed at great expense, only to sit on the shelf of the engineer’s cubicle. These bluebooks need to be made actionable, so that they are not simply a standard procedure that it is static, but are incorporated in the PLM design and development tools. This is but one example of combining Systems Engineering and Product Lifecycle Management.

6.2 Engage Systems Engineering across the lifecycle

As noted above, Systems Engineering often simply fades out in the phases beyond the product development phase. Part of that issue was that systems engineers have very little visibility of the product once it has left their siloed area in the engineering department. With Product Lifecycle Management, systems engineers can remain engaged with the product throughout the product lifecycle. As noted above, the feedback loops from manufacturing, support, and even disposal need to be enabled in order to provide Systems Engineering with the necessary visibility of further product lifecycles.

Systems engineers need to follow the product throughout its life in order to ascertain whether the work that they've done on the front end really did produce a system that

performed as required and expected. Simply testing the requirements in the development phase is no longer acceptable. If the systems engineer is to be truly involved with the entire system, then they will have to stop limiting their engagement to the design and development phase.

6.3 Adopt Systems Engineering methodology within PLM

In a number of organizations, the information systems group provides the PLM processes and procedures. While that may be acceptable from an information systems standpoint, it often does not address the problems that systems engineers actually have. Information systems needs to be the enabler of this capability, but the Systems Engineering methodology should drive it.

As I noted above, making the engineering bluebook standards actionable is an example of incorporating Systems Engineering methodology within Product Lifecycle Management. PLM has done an excellent job in capturing and organizing product information. It has also begun to develop the workflow issues. It will provide real value when it can enable the Systems Engineering methodology, which up until now has been involved with more static information such as documents and reports.

There are going to be many other opportunities for Product Lifecycle Management and Systems Engineering to integrate and provide additional value to its users. The above suggestions are but an initial list of some obvious areas where this should occur. This area has been under-investigated by both Product Lifecycle Management and Systems Engineering researchers. I would hope that would change in the future.

References

1. Wymore, A. W. (1967). *A mathematical theory of systems engineering: the elements*. New York,: Wiley.

2. Blanchard, B. S. (2004). *System engineering management (3rd ed.)*. Hoboken, NJ: John Wiley.
3. Kossiakoff, A., Sweet, W. N., Seymour, S., & Biemer, S. (2010). *Systems engineering principles and practice (2nd ed.)*. Hoboken, N.J.: Wiley-Interscience.
4. NASA Systems Engineering Handbook. (2007). NASA.
5. Grieves, M. (2005). Product Lifecycle Management: the new paradigm for enterprises. *Int. J. Product Development, 2*(Nos. 1/2), 71-84.
6. Stark, J. (2005). *Product lifecycle management : 21st century paradigm for product realisation*. London: Springer.
7. Grieves, M. (2006). *Product Lifecycle Management: Driving the Next Generation of Lean Thinking*. New York: McGraw-Hill.
8. Caruso, P., Dumbacher, D., & Grieves, M. (2010). Product Lifecycle Management and the Quest for Sustainable Space Explorations, *AIAA SPACE 2010 Conference & Exposition* Anaheim, CA.
9. Bertalanffy, L. v. (1950). An Outline of General System Theory. *The British Journal for the Philosophy of Science, 1*(2), 134-165.

10. Stackpole, B. (2003, May 15). There's a New App in Town. *CIO Magazine*, 92-101.
11. Grieves, M. (2011). *Virtually perfect: Driving innovative and lean products through product lifecycle management*. Cocoa Beach, FL: Space Coast Press.
12. Grady, J. O. (1998). *System validation and verification*. Boca Raton: CRC Press.
13. Estefan, J. A. (2008). Survey of Model-Based Systems Engineering (MBSE) Methodologies: INCOSE.
14. Friedenthal, S., Moore, A., & Steiner, R. (2008). *A practical guide to SysML : Systems Model Language*. Burlington, Mass.: Elsevier/Morgan Kaufmann.