

9 LAWS OF

EFFECTIVE SYSTEMS ENGINEERING



by Zane Scott
Vice President of Professional Services
Vitech Corporation
www.vitechcorp.com

9 Laws of Effective Systems Engineering

Copyright © 2012, 2016 Vitech Corporation. All rights reserved.

Vitech Corporation
info@vitechcorp.com
www.vitechcorp.com

CORE™, GENESYS™, and STRATA™ are registered trademarks of Vitech Corporation.
Other product names mentioned herein are used for identification purposes only
and may be trademarks of their respective companies.



Executive Summary

Systems engineering is increasingly important in today's business world. Even in businesses and industries where the term "systems engineering" is unknown, the need to guide the overall design and maintenance of business products and processes is apparent. Business cannot afford to develop products that won't meet their customers' needs or implement processes that will not "plug into" their enterprise frameworks.

For the manager seeking project success, the design team seeking to deliver a solution, and the customer seeking an answer to their needs, systems engineering is critical. It is through the application of sound systems engineering practices that the ultimate solution can be crafted to hit the mark while minimizing or eliminating unintended consequences. It is the systems engineer who maintains the systems perspective on the underlying needs and value proposition throughout the quest for a solution. It is the systems engineer who tracks the interaction of the system with its environment and works to prevent any unplanned, detrimental interactions that might result from system design choices made along the way.

Without this systems perspective, solutions can go seriously awry. Unintended consequences can make the "cure" quite literally worse than the "disease." Design choices can cause the solution to veer away from the customer needs that called for the solution in the first place. Sound systems engineering approaches stand against these possibilities.

Meeting the need for efficient and effective solution design is best accomplished by following nine "laws":

Law #1 - Begin with the End in Mind

Law #2 - It Doesn't Help to Solve the Wrong Problem

Law #3 - Insight is the Goal

Law #4 - The Model is the Main Thing

Law #5 - To Catch (Design) a System, You Have to Think Like One

Law #6 - It's All about Relationships

Law #7 - Even a Set of Views is not a Model

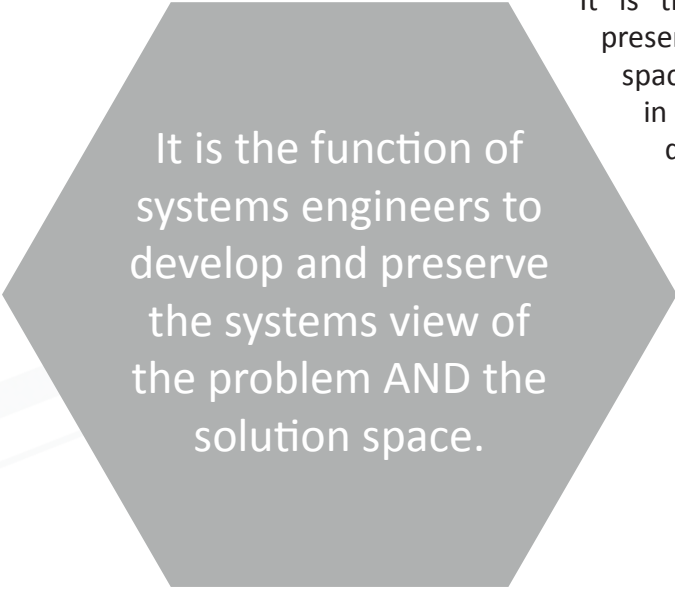
Law #8 - Choose the Representation that Best Suits the Audience

Law #9 - Systems Come in Threes

Unintended
consequences can
make the "cure"
quite literally worse
than the "disease."

The Laws

The job of the systems engineer (with or without that title) is to see that new or improved products and processes hit the intended targets. Meeting customer needs and improving business process quality are critical. Very often systems engineering has been shuttled off to act as the project record keeper to assemble, catalog, and retrieve project documentation. But an effective systems engineering process resides at the very center of successful system solutions.



It is the function of systems engineers to develop and preserve the systems view of the problem AND the solution space.

It is the function of systems engineers to develop and preserve the systems view of the problem *and* the solution space. Systems engineers keep the solution on track and in context. They do far more than serve as guardians of documentation.

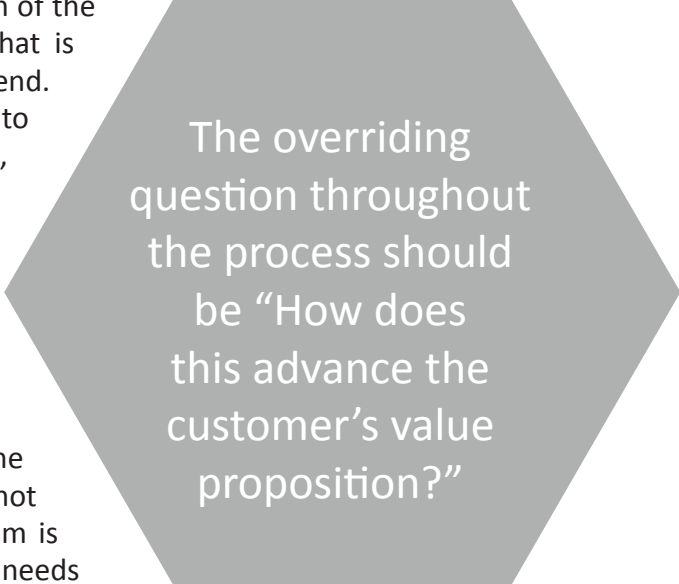
In Vitech's 20-plus years of delivering the right products and services on schedule and under budget, it has become clear that the path to efficient and effective systems engineering is governed by nine laws.

Law #1 - Begin with the End in Mind

It is critical to remember throughout the project that the customer's value proposition is the end to which everything else is the means. It's not about developing specifications. It's not about slavish devotion to a specific process. Diagrams and models are not ends unto themselves. In fact, it's not even about delivering a system; this is simply a way to bring about the desired results. Meeting the customer's needs without the introduction of unintended consequences is what it's all about. The only reason to satisfy the requirements is that those requirements are the expression of the customer's needs. When those are truly satisfied, customers and stakeholders alike have truly benefitted from the solution.

The integrity of the design process is preserved by beginning with the end firmly in mind and keeping the satisfaction of the requirements in sight at every juncture. Everything that is done along the way should be done in service to this end. Processes, specifications, and models can all serve to help reach the destination, but serving the processes, specifications, and models instead is wasted effort and counterproductive. The overriding question throughout the process should be “How does this advance the customer’s value proposition?” Maintaining a tight linkage to that destination keeps the design process on track from beginning to end.

The design process should converge on a solution to the customer’s problem. Without a clear direction, that is not possible. From the beginning stages where the problem is clearly defined to the final design choices, the process needs the discipline of a coherent methodology to guide it through the decisions and choices that must be made.



The overriding question throughout the process should be “How does this advance the customer’s value proposition?”

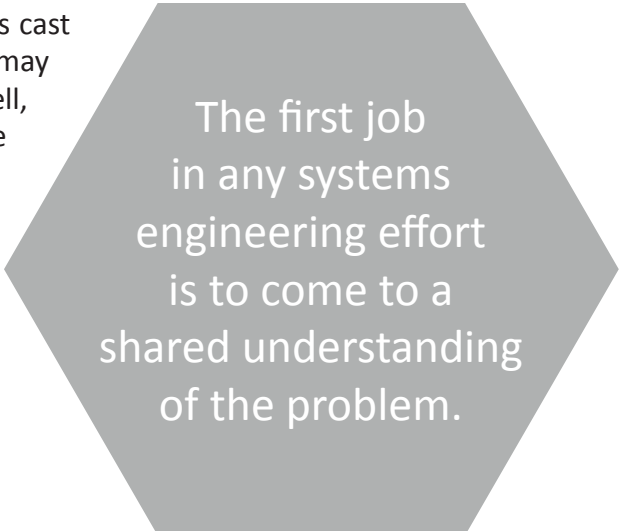
Law #2 - It Doesn't Help to Solve the Wrong Problem

Russell Ackoff, business management professor and systems thinker, said it best: “We fail more often because we solve the wrong problem than because we get the wrong solution to the right problem.” The danger is that the process of seeking a solution must be pointed at the right problem in order to solve it, and any failure to understand what that problem is will cause the process to be off the mark. Yogi Berra was right when he observed that, “If you don’t know where you’re going, you’ll end up someplace else.”

Customers have particular needs driving their quest for system solutions. Sometimes those needs are felt but not well understood by the customer. Typically, the customer describes “the problem” by describing symptoms in the best way they know how — countless requirements statements. These symptoms are pain points caused by the problem but may not provide a clear or complete description of the problem itself. Frequently, these statements are accepted at face value as a true and accurate representation of the real customer needs and desires. Too often, these statements are simply reorganized, decomposed, and faithfully traced, establishing an incomplete or fundamentally incorrect foundation for the challenge at hand.

9 Laws of Effective Systems Engineering

The first job in any systems engineering effort is to come to a shared understanding of the problem — not necessarily the problem as cast by the requirements statements, but the real problem. This may entail helping the customer to mature their understanding as well, but until the problem is identified and understood, it cannot be addressed. The first challenge is to model the problem in order to gain a complete understanding of it. If nothing else, one needs to employ the technique of asking the “five whys” to progressively uncover the thinking behind the requirement statements, which serve as an imperfect proxy for the problem statement.



The first job in any systems engineering effort is to come to a shared understanding of the problem.

Law #3 - Insight is the Goal

Systems engineering seeks to shed light on the problem, and by so doing, illuminate the path to a solution. Along the way, design choices must be made, and again, it is the job of the systems engineer to provide light by which to make these choices. Good information feeding a good process leads to insight, and insight leads to better choices. This is the power of systems engineering.

As in other engineering disciplines, models play a key role. They allow us to unambiguously capture and communicate our understanding. They provide a mechanism to reflect reality in such a way as to focus on the critical dimensions. And they enable us to coherently reason about a problem and its solution in a way that is not possible in the abstract.

When systems engineering is based in a systems model, the ability to provide critical insight is multiplied. A key objective of using model-based systems engineering (MBSE) is to garner greater insight into the systems solution under design.

From the outset, the model allows for an integration of the design across the system lifecycle and across all domains:

- Requirements
- Functional behavior
- Physical architecture
- Validation and verification

This integration allows for the patent traceability of system requirements through the design and back again. This traceability provides assurance that the solution truly meets the requirements in addressing the problem. We can examine the system design expressed in the model and see clearly whether each and every requirement has become the basis of one or more behaviors. Likewise, we can see whether those behaviors, which have come from a functional analysis of the requirements, are all allocated to particular components. This gives us a reasonable assurance that the components of the physical implementation of the systems design will perform the behaviors which are based on the requirements. This assurance means that the physical implementation of the system will meet the purposes for which it was intended.

Insight goes far beyond the fundamental bookkeeping of traceability.

But insight goes far beyond the fundamental bookkeeping of traceability. In a traditional systems problem, it begins with true analysis of the concept of operations and requirements statements to identify gaps, conflicts, over-specification, and risks. It includes analyzing the challenge in the behavioral domain, first to seek the logical solution of *what* the solution must do before determining the physical implementation, or *how* it will do it. It continues to test and evaluate, verify, and validate. Throughout, we look for completeness and consistency, applying our analytical approaches, automated support, and collective experience to elicit the necessary insight in the conceptual phase, so that the ultimate system meets the stated needs efficiently, effectively, and economically.

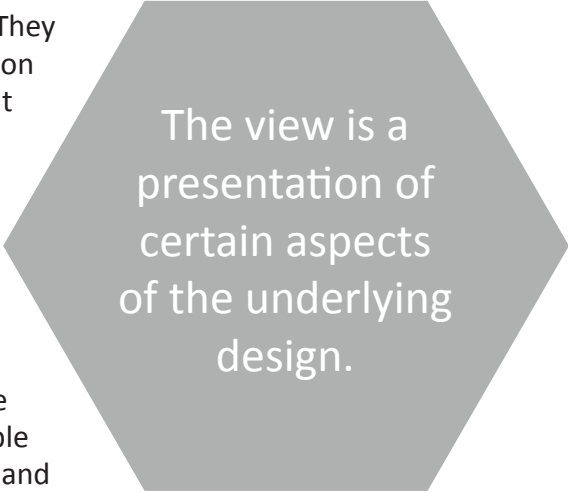
As a result of having a model, we can secure insight into the ultimate success of the system under design. Therefore, a coherent and completely integrated model offers the true design insight, which is our aim.

Law #4 - The Model is the Main Thing

At one level this is obvious — engineering classically uses models to understand, analyze, and ultimately solve the challenges we face. But what seems obvious holds a key truth. Proper use of model-based approaches unlocks the power of systems engineering. The model itself becomes the focus, not the documents describing the model. The old approach was to capture the system specification in a set of documents and then extract them for the purpose of implementing the system solution. In a model-based design, the documentation is generated *from* the model and reflects the model itself.

In systems engineering as it is most often classically practiced, the documents (specifications) are substituted for the model itself. In many contexts we hear of a set of views referred to as a system model. But specifications and views are not a model. The model is the set of entities, relationships, and attributes that contain the complete design of the system solution.

Rather than being the system design container, documents and views are projections of a model from a specific perspective. They satisfy specific — and valuable — viewpoints, but by definition are limited in scope to address a particular need. So whether it is a DoDAF viewpoint, a SysML diagram, or a functional flow block diagram, the view is a presentation of certain aspects of the underlying design. It does not become the model, even in combination with other views. The documents or views flow from the model rather than the model flowing from the documents.



The view is a presentation of certain aspects of the underlying design.

Ultimately, the model is a tool for reasoning through the solution space. For example, the model provides a reasonable context for trade studies. We can use the model to test and compare alternative functional allocations. Where the model used is one which has been constructed with the rigor and discipline of the principles of model-based systems engineering, these studies and comparisons can be made within the context of the system requirements. This allows us to hold the context of the comparisons constant across the alternatives.

The model allows us to integrate the design into a unitary whole. The model can be seen, measured and executed as a whole. This leads to a high level of confidence in the design. A single, integrated model of the systems solution is the heart and soul of effective systems engineering. It is easy to be led astray into thinking that a set of documents and diagrams are the model. The more robust and useful the set, the easier it is. But we must focus on the model and not the representations.

Law #5 - To Catch (Design) a System, You Have to Think Like One

Western education classically teaches us to analyze or deconstruct the subject of our investigations. We seek to understand any whole by understanding the operation of its parts. But this analytic thinking is not enough. We need to engage in synthetic thinking as well. The distinguishing characteristic of a system is that the system is more than the sum of its parts. Therefore, we must synthesize our thinking at the system level. Systems thinking, for true understanding, involves synthetic as well as analytic thinking.

One of the dangers of using analytic thinking alone is that it can lead us away from the view of the system as a whole. The loss of this vision can remove the context for the elements of the system. This is often referred to as component engineering. Components are developed in isolation from one another and then cobbled together to form a system. This means that the synergistic results which are the point of the system design are either lost or badly compromised.



Figure 1

Component engineering destroys synergy by isolating parts of the system.

9 Laws of Effective Systems Engineering

As the advertising slogan “Pay me now or pay me later” reminds us, there are critical functions in life that cannot be avoided. Systems engineering from a systems perspective is one of those. Skipping the systems thinking up front does not mean saving that effort — it means saving that effort for later, often at integration and test when it is far more problematic. Ultimately, retrofitting and rework are an extremely costly way to do the integration work of systems engineering, to say nothing of the costs involved in accepting degraded performance because it is too late or too expensive to make the changes necessary for integration.

Throughout the design, we use synthetic thinking to maintain the relationships among all the aspects of the model.

When we engage in systems thinking, we move from layer to layer, considering all aspects of the system at a particular level of granularity. Systems thinking begins at a high level where there is relatively little granularity or detail. Here we consider all aspects of the design and their relationships to each other. We use analytic thinking within the layer to flesh out the level of detail required at that particular level. Throughout the design, we use synthetic thinking to maintain the relationships among all aspects of the model.

Using this layered approach, the system model is complete within each layer in relationships between the various aspects, and is analyzed to the proper level of detail. No aspect of the system design is pushed beyond the detail level of any other aspect. Every design decision is made within the context of the entire system, and there is no need to revisit the design in order to restore that context.

The system design is coherent throughout the design process in that all aspects of the design are held in context with each other. This means that the ultimate detailed design will assure the fulfillment of the purposes of the system and that the system, when implemented, will meet its originating requirements.

Law #6 - It's All about Relationships

When we talk about systems, it's all about interactions and interrelationships. We're focused not upon the performance and characteristics of the independent pieces but the performance and characteristics of the whole. It's about interfaces and links as much as it is about individual components and parts.

Likewise, systems engineering is all about the relationships. All of the entities that make up a system are webbed together into the model through relationships that define their place in the system. These relationships (together with modeling constructs to capture key additional aspects) are what transform a collection of entities into a traceable, executable model.

The relationships are themselves organized into a meta-model or schema. Done well, this meta-model not only forms a framework for capturing the system representation, it provides a framework for thinking about the problem and its potential solutions. As the model takes shape, the entities are related to each other in a model of how the engineers see the problem being solved. Because those relationships exist and draw together the entities into a cohesive model, the solution under consideration can be examined, compared to other potential solutions, and tested against the requirements.

At its simplest level, the basic or foundational schema can be simply expressed as “requirements are the basis of behavior, and behavior is allocated to components.” Likewise, “components perform behavior, and behavior is based on requirements.” This bidirectional set of relationships is the foundation of any model.

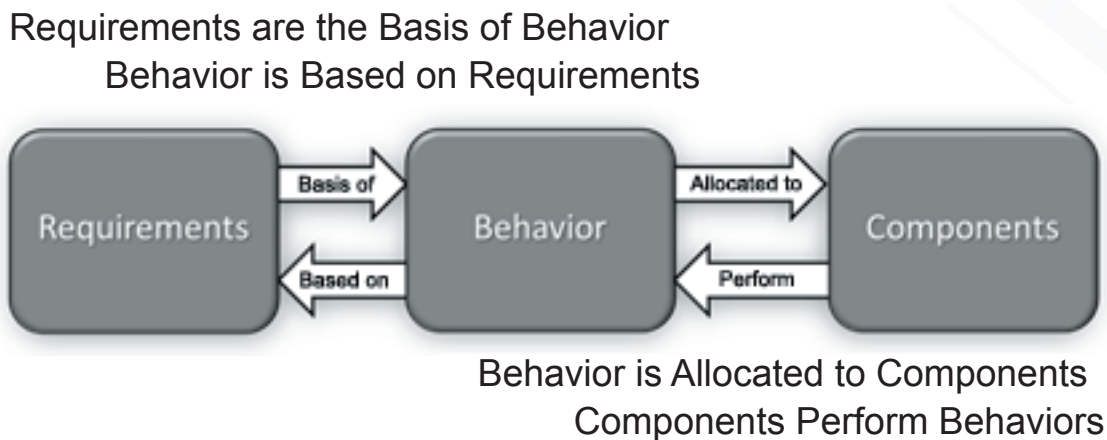


Figure 2

A basic schema relating requirements, behavior, and architecture

This very basic schema is expanded to deal with the complexity of real world problems and solutions. But even in expanded form there is always the basic simplicity of A relating to B and B relating back to A. Every relationship has two ways of expression that manage the two sides of the relationship. In actually constructing a model, there are many more expressions of these relationships which allow us to construct models for particular solutions. For example, a behavior may be “decomposed by” another behavior. It may be “triggered by” an item. These expressions allow for the construction of a much more nuanced and richer model of the system solution.

9 Laws of Effective Systems Engineering

For example, a requirement is typically the basis of a function (behavior). But it can also be refined by (or refine) another requirement; it can be verified by a verification requirement; it can be the basis of a use case — there are many possibilities. These relationships serve the thinking process and tie the requirement (or any other entity) to the model in a variety of ways. This is the basis of the richness of relationships and the multiplicity of linkages they form.

The relationships form pathways through which we can trace the system requirements into implementation. Likewise, because the relationships are always bidirectional, we can use them to trace the physical implementation back to the system requirements. And reflecting back to insight, the relationships and the underlying meta-model provide the framework for reasoning. The fundamental meta-model should be as simple as possible to support the required level of analysis and reasoning, no simpler. Moving beyond the foundational concepts above, a more complete schema connecting the operational and system domains is shown below.

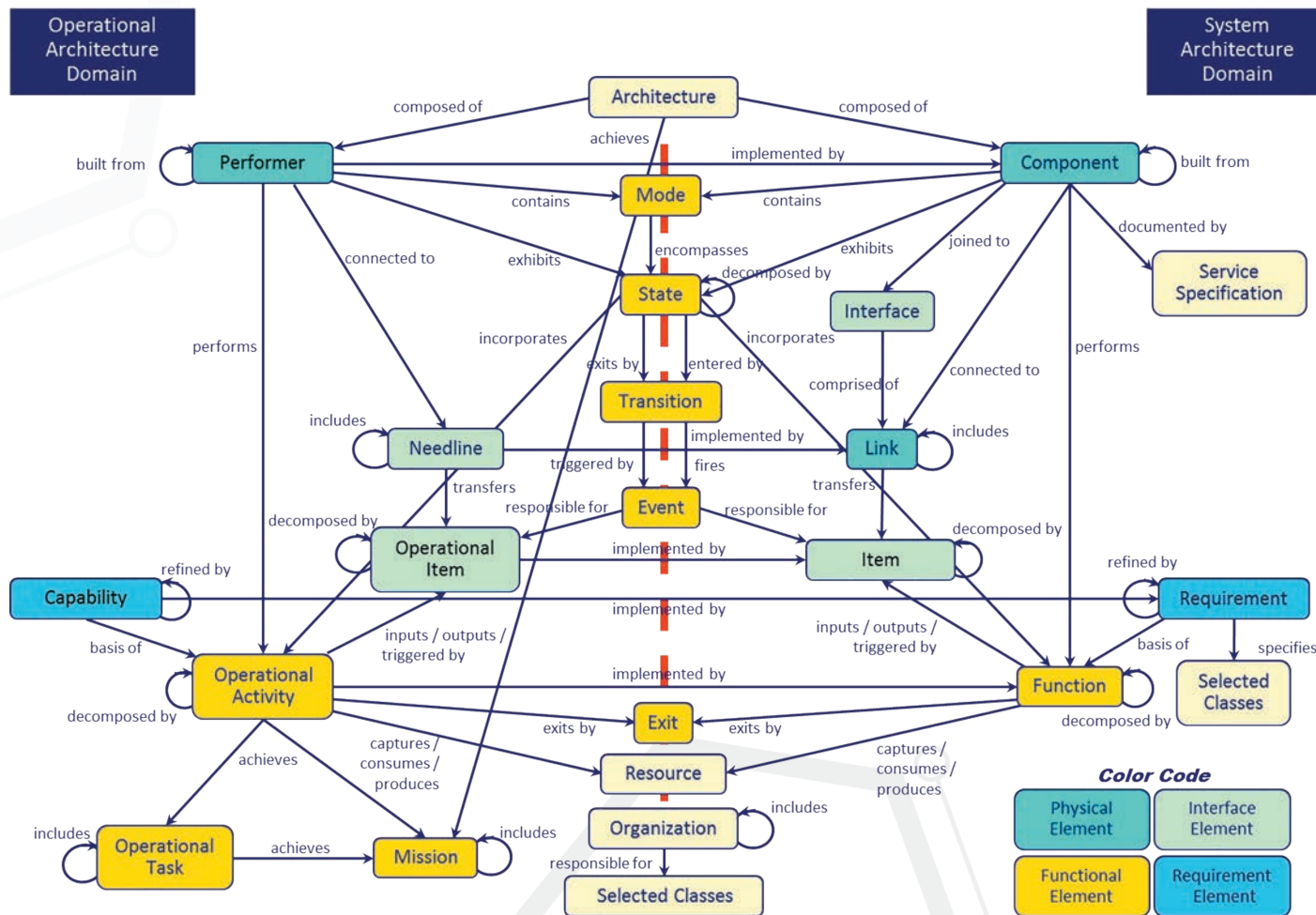


Figure 3

System models encompass operational and system domains with all entity relationships.

Just as a human body without a skeleton and other body systems to hold its parts in relationship would simply be an amorphous mass of tissue, so a system model without relationships would be a disjointed collection of elements. The essence of the system is contained in the relationships.

Law #7 - Even a Set of Views is not a Model

With the rise of model-based systems engineering, we run the risk of inadvertently substituting a decoy and dangerous approach — diagram-based SE — in place of the powerful models we need. However, much as top, left, and front views in mechanical engineering are simple projections of an underlying model, the myriad of traditional and object representations are limited views of the underlying system model.

A true model depends upon control constructs and relationships (see Law #6) webbing the system entities together into a model of the system solution. It is the entities, their properties and relationships, and the definition of their interactions that make up the model. Individual views provide valuable analytical insight and aid in communication, but they are defined from a singular viewpoint. When seeking to integrate multiple views, one quickly learns that the views are overlapping and intersecting. Without a coherent model at the foundation, diagrams are simply static representations from a fixed viewpoint. While it is important to be able to see and represent this in order to understand and evaluate the design, the representations are no more the model than a schematic of a model airplane is the model plane itself.

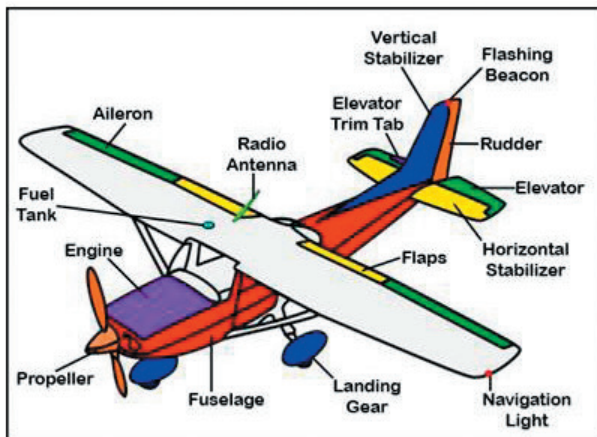


Figure 4

Flat diagrams are not models.

Often, teams will rely upon an underlying data dictionary to “align” an independent library of pictures and assert that the result is a model. But remember law #6 — it’s all about the relationships. Classically, different representations focus on different perspectives (and different relationships) of the system in question. Data dictionaries may align object names and properties, but the relationships — the critical aspect in question — is left to vary from diagram to diagram. Predictably, the result is an inconsistent, incomplete, and often incoherent picture as the complexity of the system and the desired viewpoints overwhelm our human ability to manually align these disjointed artifacts.

That means that no collection of views, no matter how robust and fit for their purpose, can become the model itself just as no collection of photographs and assembly directions can become a model airplane. They “picture” and describe the model but are not the model in reality. In both cases, the views are a representation of the underlying reality (the model plane) and cannot become the reality itself.

With the rise of model-based systems engineering, we run the risk of inadvertently substituting a decoy and dangerous approach — diagram-based SE — in place of the powerful models we need.

Law #8 - Choose the Representation that Best Suits the Audience

The role of any representation or view is to convey a particular subset of information to the intended audience in order to enhance their understanding of the system solution. Most often, representations are used by the design team to gain perspective and an understanding of the model and its interrelationships. By selecting the desired viewpoint and representation, the team can gain insight and understanding of the model suited to their particular purpose.

Representations can also be used by the team to communicate information about the model to others. Communication requires meeting the audience where they are and bringing them to the desired understanding. By considering both the situation of the audience and the team’s need for audience understanding of the model, it is possible to choose the view or views that will achieve this goal.

For example, a group of process owners will usually resonate more with functional flow representations than the physical block diagrams that reach those tasked to physical architecture design. Sequence diagrams, where time flows from top to bottom, may confuse an audience accustomed to flows plotting time from left to right. The goal is to present information in a way that is most likely to reach a specific audience. The first consideration then is what view or views the audience is most likely to understand, given their roles and experience.

Communication requires meeting the audience where they are and bringing them to the desired understanding.

The second consideration is what the audience needs or wants to know. By providing information that the audience is looking for, the communication channel is opened. Additional information can flow through that channel and be received along with the information the audience is seeking. It is helpful to meet the audience's need for information if for no other reason than to remove the obstacle of open question loops that may obstruct the flow of other information. Although the question, "What do you want/need for the audience to know?" is the reason for initiating the communication, being aware of and responsive to audience needs is certain to pave the way to accomplishing the presenter's purpose.



Figure 5

The right information in the right format for each particular audience

9 Laws of Effective Systems Engineering

The final consideration is what the design team wants the audience to know. Choosing the representation rests on the combination of the information that is needed and the ability of the particular representation to communicate that information. Too much information — meaning information not germane to the purpose of the representation — can be as bad as too little. It distracts from the purpose and blurs the message. Although Dragnet's Sgt. Joe Friday did not say the oft-quoted phrase, "Just the facts, Ma'am," the words he actually spoke were even better: "All we are interested in are the facts, Ma'am." A good representation follows Sgt. Friday's principle. It includes all of the facts needed to get the points across and adds no more to obscure the message.

For example, if the audience needs to focus on just the process flow, a functional flow block diagram will be the choice. If they need to consider triggers and data stores as well, it will take an enhanced functional flow block diagram or an activity diagram to show them in context. Views are most effective when they serve the needs of the presenter and the viewer alike. They enable the team to meet the audience where they are and bring them to where they need to be in understanding the system design.

Law #9 - Systems Come in Threes

Every system design involves three systems: the system being designed, the system it will "live" in, and the system used by the team to design it. Below is the example of designing a subway train. To the right of the train is the subway system in which it will operate, and to the left is the process chart depicting the design process.

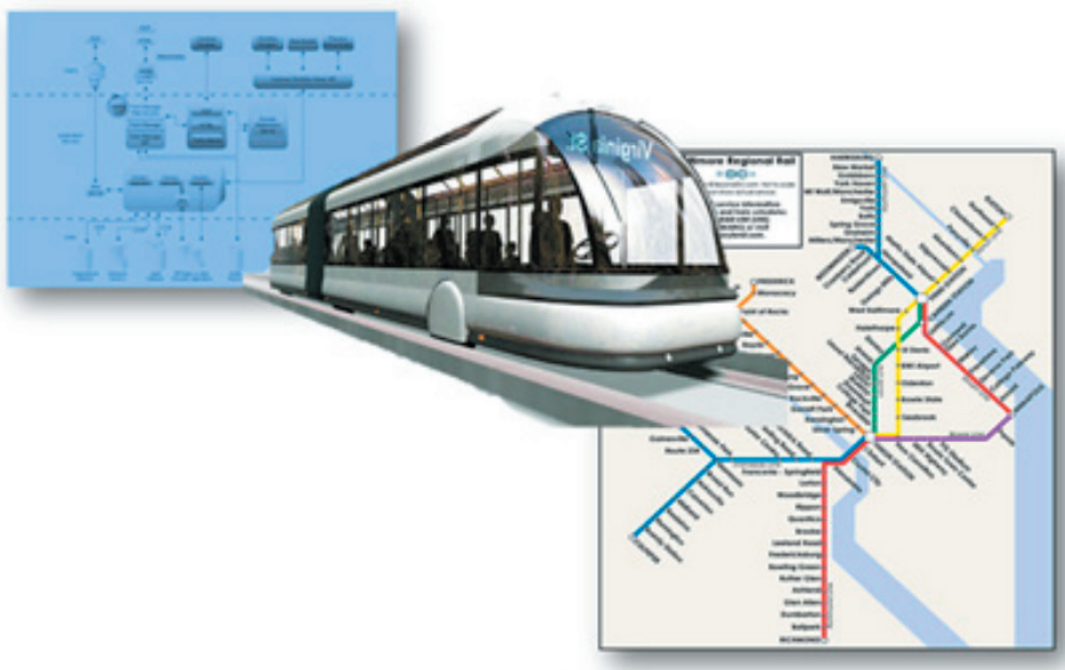


Figure 6
3 kinds of systems

Almost all design teams understand that they must focus on the first system. Its design is, after all, the purpose of their efforts. The system under design is the subject of consideration throughout the design process itself.

Many teams, however, fail to adequately consider the second system — the context in which the new system will operate. This failure can lead to unintended consequences and/or inadequacies in the design solution. This system is becoming increasingly important as we design into existing systems and environments. The opportunity to design truly “clean sheet” or “top down” unprecedented systems is becoming increasingly rare. Not many organizations can scrap all the existing technology and processes to accommodate a truly new system. They must retain systems and technology already in place and use the new designs in conjunction with their existing environment. Ignoring this second system represented by the operating environment is a recipe for design failure and implementation problems.

Often the design team mistakenly blends the solution being designed and the design process.

Likewise, most teams do not intentionally factor in the third system — the system they use to design the solution that is the subject of their project — in their design effort. This system typically grows ad hoc from their experience and can be disjointed and uncoordinated. Often the design team mistakenly blends the solution being designed and the design process. That can result in a disintegrated design that impairs real systems thinking (see Law #3). The systems become confused and get lost in the engineering process. Without the rigor and discipline of a well-thought-out system design process, the subject system is placed at risk. The conscientious design team must be intentional about the disciplinary structure they bring to their own processes. This is the system that provides the rigor and process that will guide their design efforts. A failure to be disciplined and intentional here can hurt the design process throughout.

NOTE: There are other models of this three-system environment. James Martin, in an excellent and detailed treatment of this subject, https://www.researchgate.net/publication/229057077_312_The_Seven_Samurai_of_Systems_Engineering_Dealing_with_the_Complexity_of_7_Interrelated_Systems, describes seven systems. Whether one chooses to think of seven systems or three, the critical concept remains: There must be an intentional and rigorous treatment of these aspects of developing the system solution.

CONCLUSION

As we have seen, the job of the systems engineer is at once critical and imposing. The responsibility of keeping the system design on track without losing sight of the context or the goals of the design can be a heavy one. Meeting that responsibility requires a disciplined and well-thought-out approach.

The nine laws discussed here provide a good look at the parameters of a successful systems engineering process. They offer sound guidance for the practitioner seeking to maintain the systems perspective and drive a disciplined process to a successful systems solution for the problems at hand. By following these laws, the systems engineer will have a sound path charted and a reasonable assurance of success.

Vitech offers a strong leveraged approach to following these laws. Vitech's STRATA™ approach offers a disciplined, rigorous way to keep the design process on course. The system design proceeds from layer to layer, increasing the granularity with which the system is described. This allows the design team to maintain the system view (Law #5) and converge on a solution to the right problem (Law #2). The STRATA methodology — the system used to design the solution — calls for an understanding of the system being designed AND the context in which it will live (Law #9). This approach yields insight into the nature and structure of the problem and the possible solutions (Law #3).

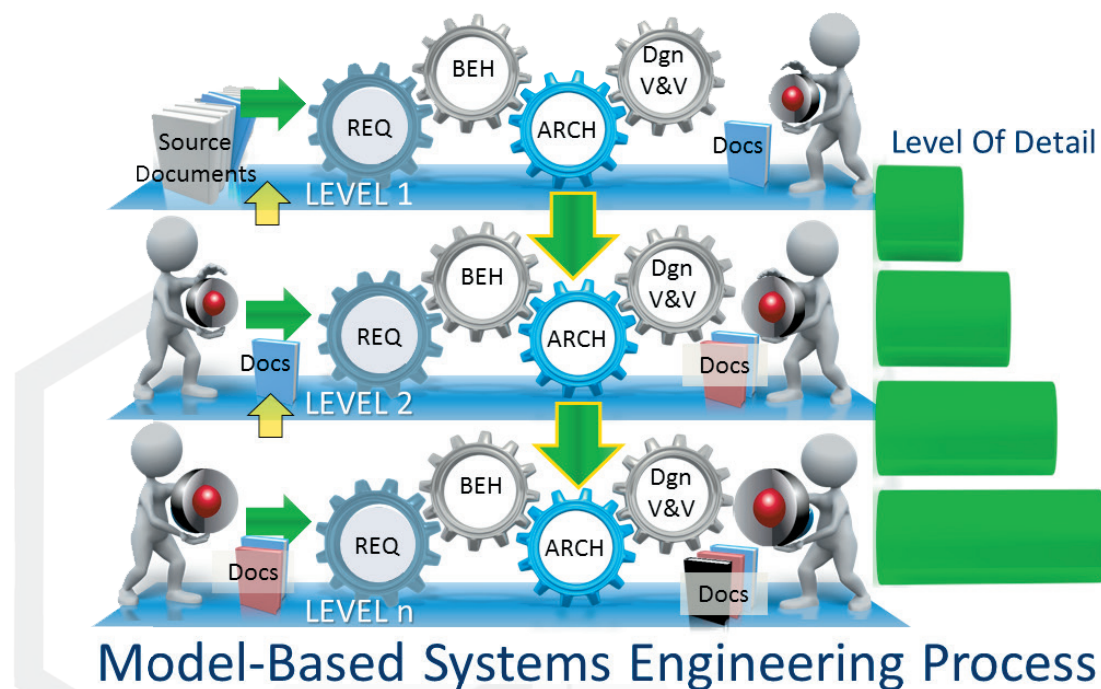


Figure 7
STRATA – Strategic layers for systems engineering

STRATA is best leveraged by a single model, single repository tool such as CORE™ or GENESYS™. Where there are multiple tools constructing multiple models of a system under design, the relationships between the models can become very difficult to maintain. When this happens, engineers serve as tool integrators or consistency checkers rather than focusing on the value-add of systems engineering. At one level, there are problems making sure that changes made to one of the piecemeal models are translated and tracked into the other partial models. Even where the changes are translated from one to another, there is a danger of data loss, misinterpretation or corruption between models and tools.

The use of a single tool in an integrated repository maximizes the power of the model under construction. Work on any aspect of the model is dynamically reflected throughout the model. The tool tracks any changes into all related aspects of the model through the web of relationships which are established in the model in the MBSE design process (Law #6). A single coherent model contained in a single repository and constructed and maintained with a single tool offers powerful system design (Law #4).

With this context available to the engineer in real time, design decisions can be made efficiently by following the logical progression of related and resulting changes. Nothing is lost or forgotten, and all the ramifications of each development step are taken into account. Whether at the macro or micro level, the context can be clearly seen by the design team, and a high-quality design is assured.

In the Vitech approach, the model is the basis for, rather than the product of, the documentation. Views, reports and other documents are generated from the model through the tool. This allows the team to select the view or document best suited to their purpose (Law #8), and generate it automatically for interactive design or team communication. The resulting views and/or documents are the reflection of the model but are not the model itself (Law #7). Although information from other sources can be incorporated into the model, the model is not to be found in the documents or the views. They are simply the descriptive artifacts of the model.

In the end, the discipline and rigor of the methodology supported by the comprehensive scope of a tool help deliver a high-quality solution to the system design problem. Because of the ability to address problem and solution complexity in real time through intentional design choices, the risk of unintended consequences is minimized. The tool and method lead the team along an efficient path to a convergent solution. The investment of time and resources to position the design team garners a significant return in the form of process efficiency and design quality.

Using STRATA leveraged by CORE or GENESYS, the design team is positioned to most positively impact the customer's value proposition with the system solution (Law #1). Following STRATA keeps the team on track and aimed at the right problem, avoiding Yogi Berra's warning that "If you don't know where you're going, you'll end up someplace else." Instead, the solution solves the right problem in the best way possible.

About the Author



Zane Scott, as Vice President of Professional Services at Vitech Corporation, manages the company's consulting and training division. For the past 25 years, Zane has built a skill set which enables him to provide insight and guidance to individuals and companies as they improve organizational processes and methodologies. He has taught systems engineering methodology, is a frequent blogger and webinar presenter on model-based systems engineering, and is co-author of Vitech's *A Primer for Model-Based Systems Engineering*.

With a professional background in litigation, Zane is also a trained negotiator, labor management facilitator, and mediator. He has practiced tactical negotiation and interventional mediation, and taught communications, conflict management, and leadership skills at both the university and professional level. Before joining Vitech, Zane worked as a senior consultant and process analyst assisting government and industry clients in implementing and managing organizational change.

About Vitech

For over two decades, Vitech has helped organizations raise their systems engineering proficiency through a tailored combination of training, services, and software. By engaging with Vitech, organizations around the globe increase their productivity, enhance agility, and reduce project risk.

Unlike siloed approaches and products that mask critical context and system interactions, Vitech's approach and its GENESYS™ and CORE™ software embrace the holistic aspects of systems engineering. These solutions enable teams to clearly capture and address systems concerns from problem identification through requirements, architecture, and testing in an integrated model. These solutions manage critical interrelationships to guarantee consistency and design integrity. The result is a team empowered to engineer with confidence, free to focus on creativity, innovation, and analysis to effectively deliver against stakeholder needs.