

THE ROLE AND LIMITATIONS OF MODELING AND SIMULATION IN SYSTEMS DESIGN

Jason M. Aughenbaugh

Systems Realization Laboratory
G.W. Woodruff School of Mechanical Engineering,
Georgia Institute of Technology
Atlanta, GA 30332-0405, USA.
Email: gtg224k@mail.gatech.edu

Christiaan J.J. Paredis

Systems Realization Laboratory
G.W. Woodruff School of Mechanical Engineering,
Georgia Institute of Technology,
Atlanta, GA 30332-0405, USA.
Email: chris.paredis@me.gatech.edu

ABSTRACT

To design today's complex, multi-disciplinary systems, designers need a design method that allows them to systematically decompose a complex design problem into simpler sub-problems. Systems engineering provides such a framework. In an iterative, hierarchical fashion systems are decomposed into subsystems and requirements are allocated to these subsystems based on estimates of their attributes. In this paper, we investigate the role and limitations of modeling and simulation in this process of system decomposition and requirements flowdown.

We first identify different levels of complexity in the estimation of system attributes, ranging from simple aggregation to complex emergent behavior. We also identify the main obstacles to the systems engineering decomposition approach: identifying coupling at the appropriate level of abstraction and characterizing and processing uncertainty. The main contributions of this paper are to identify these short-comings, present the role of modeling and simulation in overcoming these shortcomings, and discuss research directions for addressing these issues and expanding the role of modeling and simulation in the future.

KEYWORDS

Systems engineering, modeling, simulation, simulation-based design, uncertainty

INTRODUCTION

As engineering systems are becoming more and more complex, it is impossible to solve the associated design problems in one step. Even individual subsystems are no longer dominated by one technology [1] and require cooperation among engineers from various disciplines. At the same time, an emphasis on *concurrent engineering* [2] broadens the focus of design to a product's entire lifecycle. This introduces more objectives into decisions, and it requires knowledge about factors such as

manufacturability and disposability that designers traditionally have not considered during the early stages of design.

The increased complexity and expanded focus of design each place large informational needs on the designer. Most individual designers lack the range of experience and knowledge required for modern systems design and concurrent engineering. Consequently, designers must work in multidisciplinary teams and rely on modeling and simulation to share their knowledge with the rest of the team.

Without modeling and simulation, design relies on implicit knowledge. Implicit knowledge is unreliable in that designers do not know the assumptions and uncertainty in the knowledge *explicitly*. When decisions are coupled and require input from several experts, there is no way to make tradeoffs using only implicit knowledge about uncertainties. First, an individual cannot consider all of the interactions at once. Second, uncertainties from different sources cannot be measured, aggregated, or compared. Third, with different people having knowledge about only part of the system, a joint decision cannot be made. Under these conditions, systems design in a collaborative, multidisciplinary setting cannot be supported without the representation of uncertainty and the use of modeling and simulation.

One way to organize a team of designers is through *systematic design* [3]. Systematic design is based on several strong principles, but its central belief is in a need for a systematic approach to engineering design and decision making. Without a systematic approach, it is unlikely that designers will choose the correct path, and without the correct path, it is unlikely that the correct solution will be found.

The systematic design approach takes designers through a series of steps that work well for design tasks that are simple in function, such as a hinge mechanism for the bay door of a space vehicle (nominally it must open, close, and lock the door in various positions), even if the physical realization is quite complex. However, systematic design is not as suitable for

designing an entire spacecraft to carry men to Mars. A spacecraft is a very complex system with many functions that evolve over time, span many disciplines, and are strongly coupled. While many of the core principles of systematic design are relevant in the design of such a complex system, the systematic design *process* does not adequately address subsystem interactions, emergent functional behaviors, system integration, and the division and coordination of tasks between non-located, interdisciplinary design teams. A holistic, hierarchical decomposition approach to the design process that addresses these problems is provided by *systems engineering* [4, 5].

In this paper, we summarize the systems engineering approach to the product development cycle. We identify four types of system attributes. Because a system's requirements are expressed in terms of these attributes, designers must consider the effects that design decisions have on the system's attributes in order to meet the system requirements. Designers can explore these effects and estimate attributes through modeling and simulation. Although current modeling and simulation methods focus on the latter stages of systematic design (*embodiment* and *detail design*), there are clear opportunities to use modeling and simulation to support the earlier design phases such as decomposition.

Designers face several challenges to the requirements definition and decomposition process. Specifically, design is characterized by uncertainty. Designers lack knowledge about a system's true requirements, its environment of operation, future design decisions, and emergent system behaviors. We present several ways in which modeling and simulation can help designers overcome these challenges.

We illustrate the role and limitations of modeling in the systems engineering process through the example of a passenger conveyance system for an airport. We assume a general layout of several buildings that the system must connect. The primary attributes of the system are passenger wait times and cost. Naturally there are many more, but these suffice as an example in this paper. For the example design, we choose the concept implemented at Atlanta's Hartsfield-Jackson airport. This concept is an underground train system that links the terminals with multiple trains running on two tracks.

SYSTEMS ENGINEERING OVERVIEW

Systems engineers take a holistic approach to the product development cycle. They recognize the interaction between the *product system* being designed and the *human system* that designs the product system. The goal of the human system is to design a product system that meets the needs and desires of stakeholders. In this context, systems engineers coordinate and supervise the transition from stakeholder needs to the specifications that discipline engineers use to embody a physical entity. The systems engineers also plan and oversee the integration and qualification of the system.

Several models of the systems engineering product development process have been developed [4, 5]. The most commonly adopted model of the development process is the *Vee* model [6, 7] shown in Figure 1. The left side of the Vee represents the decomposition of the system into subsystems and

the preliminary definition of those subsystems. The right side represents the integration of the fully detailed subsystems and system qualification. The base of the Vee is discipline design, the phase in which engineers fully define the system components.

The Vee is really a three-dimensional model. The decomposition of the system at each level of the Vee is represented by parallel blocks running into the page. For each of these blocks, designers repeat the process of developing solutions for that block and defining the requirements for the next level in the decomposition [6]. This process involves a repetition of the *clarification of task* and *conceptual design* phases of systematic design. The system concept is recursively refined until there is little value in further decomposition.

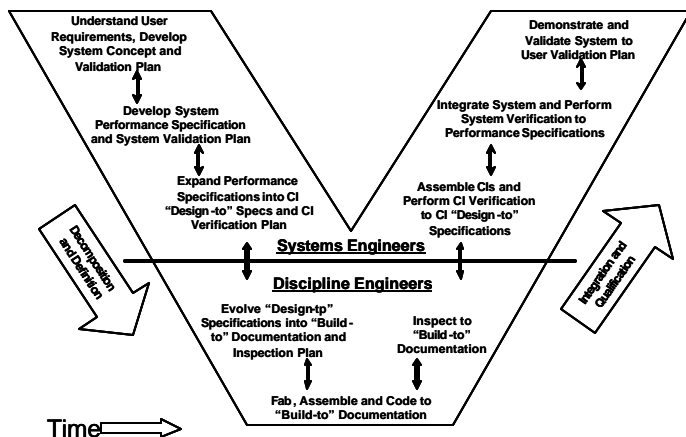


Figure 1: The Systems Engineering Vee Model [6, 7]

The horizontal line dividing the top branches of the Vee from its base represents the point at which the systems engineers hand the specification to the discipline engineers. Discipline engineers, such as mechanical, electrical, chemical, aerospace, or computer engineers, complete the *embodiment* and fully *detailed design* of physical components or software. Communication across the division between systems engineers and discipline engineers, as well as the simultaneous execution of decomposition, design, integration, and qualification tasks, will significantly impact the success of the system design process [4]. In addition to well structured management and documentation, modeling and simulation can help to bridge this communication gap.

In order to reduce the likelihood that requirements cannot be met later in the product development cycle, the Vee model encourages exploratory design and analysis work during the early steps of the cycle [7]. As we illustrate in this paper, modeling and simulation help reach this goal by supporting exploration of the design during the decomposition process.

DEFINING REQUIREMENTS AND ESTIMATING ATTRIBUTES

Systems engineering and the Vee model are characterized by a hierarchical decomposition of the system. It is valuable to use modeling and simulation to characterize the relationships between the different parts of the decomposition.

As shown in Figure 2, when designers make decisions at one level in the decomposition, they establish requirements for the next level—*requirements flowdown*. In these decisions, designers constrain the design variables for subsequent decisions and levels of decomposition. Each decision also affects attributes (such as cost or passenger wait times) that propagate back up the hierarchy. It is in terms of these attributes that system requirements are expressed. Therefore, the attributes in a way measure the success of the design. Designers can estimate the effects that possible decisions have on these attributes using modeling and simulation. Designers can use these estimates to make better decisions and to design a system that is more likely to meet its requirements.

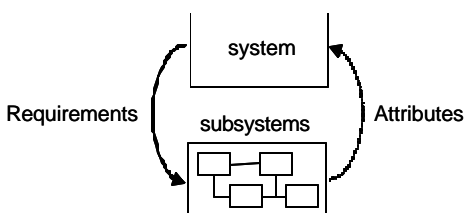


Figure 2: Requirements and attributes

In an example functional architecture for the airport passenger conveyance system (Figure 3), the main function *transport passengers* is composed of three subfunctions—*load people*, *move people*, and *discharge people*. We will focus on the decomposition of the *load people* subfunction.

After requirements flow down ($1r$ in Figure 3) from the top level design decisions (*step 1*), designers can take decisions on the definition and decomposition of the function *load people* in *step 2*. We assume that the designers decide to include doors to meet safety requirements. This decision defines the decomposition of *load people* and the requirements ($2r$) for each of its subfunctions. By choosing to have doors, the designers also affect the attributes ($2a$) that flow back up to the system level. Before taking a decision, designers can estimate its effects on attributes with modeling and simulation. The attributes already have been estimated at a higher level, but with each decomposition decision, the estimates can be improved. For example, a model can help answer how the inclusion of doors will affect design costs and passenger wait times.

Designers cannot always treat subsystems independently because the subsystems interact, such as $3int$ in Figure 3. In the example system’s operational architecture, the functions *open doors* and *close doors* are mapped to the same mechanism in the physical architecture. The two functions therefore constrain

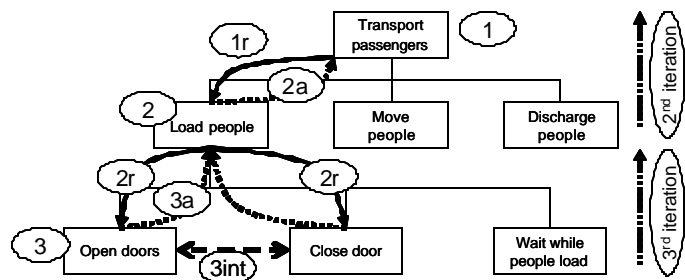


Figure 3: System decomposition

each other; the mechanism chosen for *open doors* affects how the *close door* function can be implemented.

Another example of an interaction is illustrated by the selection of the door actuator. If designers select an electric motor, they necessitate an appropriate power supply, e.g. electric battery. Here the interaction is not a result of the chosen mapping between the functional architecture and the physical architecture. The interaction instead results from the fundamental physics of the design.

Under certain conditions, designers will have to iterate while decomposing the system. One reason for iteration is if the requirements flowdown is in error. This can happen in two ways:

- The requirements established higher in the hierarchy are overly restrictive and a feasible subsystem does not exist; an iteration to relax the requirements is necessary.
- Even when all subsystems meet their allocated requirements, the system requirements may not be met; this could be due to unanticipated interactions between subsystems or due to uncertainty in the models of the individual subsystems

In the first case, designers lack information about feasibility. In the second, they lack information about the attributes. We focus on this second case.

Most system-level attributes cannot be known exactly before the system is fully specified. However, as designers take decisions and trim the design space, they bound the range of the systems attributes. To help estimate the attributes that would result from a particular decision, designers can use modeling and simulation. The better the bounds on these estimates, the less iteration will be required. An important goal of modeling and simulation should therefore be to reduce and characterize uncertainty in models for system attributes at all stages of design. Naturally, uncertainty cannot be eliminated, so some iteration may be required whenever additional information is revealed.

TYPES OF SYSTEM ATTRIBUTES

In order to model attributes appropriately, it is valuable to identify different types of attributes. System attributes can be organized by the degree to which the complexity of subsystem interactions affects how the attributes aggregate from subsystems to higher levels in the decomposition. We identify four types of attributes, characterized in order of increasing complexity:

1. Depending on system composition, e.g. mass
2. Depending on system structure, e.g. cost
3. Depending on system operation, e.g. reliability
4. Resulting from complex emergent behavior, e.g. passenger wait times

The first three types are characterized by the source of complexity. The fourth is more loosely defined and recognizes the complexity resulting from a sufficient number of interactions in the system. Designers need to estimate each of these types of attributes during system design. As the complexity increases, modeling and simulation become more essential for making good estimates.

Composition dependent attributes

Some system attributes easily aggregate from the components to the system level. For example, the mass of a system is simply the sum of the masses of the subsystems. This simple relationship depends only on the decomposition of the system into subsystems and is independent of how these subsystems are configured. To estimate the attributes of the subcomponents accurately, designers often rely on a budget approach; they impose constraints on the masses of the subsystems. Due to these budget constraints, the range of possible system masses is limited.

Structure dependent attributes

Cost is a more complex attribute. Naturally, the system's total cost will depend on the costs of the individual subsystems, but the total cost will also include costs that depend on the system's structure. For example, costs are incurred when the system is assembled. The system's structure affects the cost of assembling two or more subsystems.

Designers currently use parametric cost estimation during conceptual design. The basic models of parametric cost estimation are cost estimating relationships (CER's) that relate particular design parameters to total design cost, based on correlations in historical data and expert opinion [8]. The overall parametric cost model is a group of CER's used together to estimate the costs of the project. This method correlates an unknown attribute with more accurately known values. This approach could be extended beyond cost to any attribute for which such correlations can be found.

Parametric cost estimating is a heuristic method based on correlations in past data. Typically it involves a great deal of uncertainty, especially because it rests on the assumption that future designs will be similar to past designs. Nevertheless, prudent design engineers look for ways to explore tradeoffs in the design process through any method available, and in certain domains, parametric cost estimating has been used successfully. When there is ample historical data from which to draw relationships, parametric cost estimates are usually better than blind guessing. What designers currently do not know is how good these estimates are because the uncertainty of the models is not captured explicitly. This is an important research challenge in predictive modeling that still needs to be addressed.

Operation dependent attributes

The next type of attribute depends on the system's operation and can be represented by reliability. Assume that, for now, we consider only situations in which each component of the system is assumed to be either functioning properly or failed. The system reliability depends not only on the reliability of each component, but also on the system's *operation*. From a reliability perspective, many systems operate in combinations of series and parallel components. The reliability of series/parallel systems can be calculated from the individual component reliabilities through multiplication and addition. Other systems operate in a more complex manner and require techniques such as enumeration, conditional probability, and cut-set approaches to aggregate reliability [9]. In any of these cases, the total reliability of the system differs from the simple sums and products of the component reliabilities.

Emergent attributes

Mass, cost, and reliability are very common drivers in systems design, but paramount is the actual behavior of the system. If a system does not perform its intended functions, it is unsuccessful regardless of how light or inexpensive it is. After all, a system is built to *do* something. Unfortunately, the attributes that describe the operational behavior of a system are not as easy to aggregate as other system attributes.

Once operational attributes are included with system composition and structure, the complexity can increase even more through interactions between subsystems and components. For many operational attributes, these interactions are just as important as the internal workings of the individual subsystems. When several interactions are involved, the operational behavior is characterized by *emergent attributes*. The system behavior *emerges* during operation because the total is not merely the sum of the parts [10]. In order to make decisions that constrain the attributes of the subsystems in a way that yields the desired system level operational attributes, designers must explore the emergent behavior of the system.

One way to model and explore the intended emergent operational behavior of the system is with discrete event simulations such as those offered by using executable specifications. Originally developed for use in software engineering, an executable specification [11, 12] is a specification designed as a discrete event simulation model of the operational architecture as defined to that point in the design process. Executable specifications can be simulated directly, without humans needing to implement a separate discrete event simulation. This eliminates an extra problem of simulation model verification—the simulation model *is* the specification.

A key advantage of using executable specifications is that designers can explore the emergent behaviors before the physical embodiment of the system is defined. This means that designers can use them to explore options while decomposing the system. The simulations reveal the operational behavior of the functional architecture as defined in the specifications. The designers can use this information to specify more appropriate designs and requirements for subsystems.

LIMITATIONS OF THE DECOMPOSITION APPROACH

In a perfect world, all systems could be decomposed into uncoupled subsystems and designers would have access to perfect information and knowledge about the subsystems. In such a world, systems design would be a simple task. Subsystems could be designed independently and designers could make perfect decisions every time. Unfortunately, in the real, imperfect world, designers face significant challenges during the decomposition process, including:

- Design decisions are coupled and involve tradeoffs
- Design decisions are made under uncertainty

Because of these challenges, designers often have to backtrack in the design process. They may have to revisit earlier decisions when new information is revealed or when changes to one subsystem affect another subsystem. This iteration is undesirable because it costs time and money. The nature of

these challenges is therefore important to improving the design process. Designers need to recognize the ways in which decisions are coupled and the types of uncertainty under which these decisions are made.

Coupling of design decisions

Most design decisions require tradeoffs between subsystems and different attributes. The tradeoffs made are important to the success of the design. Early in the design process, designers are really characterizing and specifying a family of solutions. The family of solutions spans a range of specifications that could actually be realized. Within this family, the different attributes that can be realized may be correlated.

Designers have to deal with coupling between attributes. For example, a system's specification bound both the mass and cost of the members of a solution family. Designers need to estimate these bounds and understand the relationship between different attributes in order to take decisions. In some cases the coupling between two attributes can be beneficial because it allows synergies that improve the overall system. In order to take advantage of such opportunities, designers need to understand the tradeoffs available at design time. Modeling and simulation can support this process.

Another type of coupling is subsystem *interactions*. We discussed earlier how decisions about the *open door* function affect decisions about the *close door* function in the train example. The interaction between the functions results in a coupling of the design decisions for each of the two subsystems.

Types of uncertainty

To make rational decisions, designers also need to understand the uncertainty involved in the information and knowledge on which they base their decisions. It is therefore necessary for designers to recognize that there are two types of uncertainty: *aleatory uncertainty* and *epistemic uncertainty* [13].

Aleatory uncertainty is a potential deviation from reality in a prediction or model due to natural random (stochastic) behavior. Aleatory uncertainty is also known as variability, stochastic uncertainty, objective uncertainty, and irreducible uncertainty [14]. Because it results from inherent *randomness*, aleatory uncertainty can be represented using classical probability theory such as probability distribution functions. Examples of phenomena that involve or exhibit aleatory uncertainty include machining errors, material property variations, errors in communications systems due to noise, many measurement errors, and any other truly stochastic processes.

Epistemic uncertainty relates to lack of knowledge about reality. For example, epistemic uncertainty in a model is a potential deficiency in the modeling process due to a lack of knowledge or information which results in a deviation from reality [13, 14]. Epistemic uncertainty is also called imprecision [15], ignorance, reducible uncertainty, or subjective uncertainty.

It is generally incorrect to represent epistemic uncertainty using probability density functions. In some cases there is not enough information to describe the relative likelihoods of

events, and in other cases a probabilistic representation is altogether invalid. An example of the latter case is the uncertainty that results from a conscious modeling decision such as ignoring frictional energy losses in a dynamics model of the train. Although the modelers know the approximation that they made, the model still lacks knowledge about the true phenomenon. This lack of knowledge is epistemic uncertainty. An ignored frictional energy loss in general results in a *systematic* deviation from reality. Because the deviation is not stochastic, it cannot be properly described by a probability distribution.

Currently, designers lack appropriate methods for representing and computing epistemic uncertainty. In the discussion section, we explore this limitation in more detail, reference current work, and suggest future research. Here we emphasize that just the presence of epistemic uncertainty limits the decomposition process. The ultimate success of a decision depends on uncertain factors. Without a means to estimate or reduce this uncertainty, designers cannot factor the uncertainty into their decisions.

CURRENT PRACTICE FOR HANDLING COUPLING AND UNCERTAINTY

To deal with the existence of coupling and uncertainty both in the design process and in simulations, several strategies have been adopted. Although these approaches help designers complete the design process, they leave room for significant improvement through greater use of modeling and simulation and better representations of epistemic uncertainty.

Use of margins and budgets

With perfect information, designers would know exactly which designs were feasible and could allocate appropriate requirements. Without coupling, the requirements for subsystems could be allocated separately. In reality, neither of these is possible. In order to still move forward in the design process, designers often work with *budgets*. For example, in a train system the overall mass budget may be divided and further allocated to each of the subsystems. To account for the possibility that some of these allocations cannot be met (e.g., some subsystems may turn out to be infeasible or overly costly given their assigned budget), designers often withhold a margin (e.g., 20% of the total system mass) that is not allocated to any subsystem. Subsystems that cannot be realized under the initial budgets can be revisited and allocated a greater budget later.

The budget approach is not ideal because it often results in an over-designed system. For example, assume that the train's motor design team can only meet its mass budget if it implements an expensive design for its subsystem. A much cheaper (half price) but more massive design is feasible. In this example, the team designing the doors has two material options. Material A weighs twice as much as material B and is 5 percent less expensive. Either design fits within the mass budget, so the less expensive, heavier option is chosen. Essentially, the door team had a surplus in its mass budget. That surplus could have been better allocated to the motor subsystem. With a small increase in door cost, this extra motor mass allowance would have enabled the motor designers to implement a significantly less expensive solution, thereby decreasing the total system cost.

Designers usually assign margins based on implicit knowledge or generic rules of thumb. A conservative approach is to make the margins larger than is probably necessary. At first this may sound good, because it hints that more design freedom is preserved. The problem is that several subsystems usually compete for a particular budget, such as mass above. Given the same total budget, a larger margin means smaller individual budgets for each subsystem. This results in subsystems being even more over-designed.

If the uncertainty in a model were better represented and more formally handled, designers would be able to reduce the margins and relax the bounds on other specifications without increasing the system budget. If these bounds were in a region of large marginal effects on utility, the relaxation of the bounds might result in a significantly better final design.

Marginal utility and decision tradeoffs

The previous example of a mass budget also illustrates the importance of considering the *marginal effects* of varying a design parameter and making tradeoffs between attributes. Assume that the satisfaction of all requirements can be summarized in an algebraic utility function of attributes.

Designers often lack information about exactly how a design variable affects the system attributes. Ideally when designers lack information, they would postpone a decision until more information is available. This is not always possible because the design has to move forward before everything is known. Sometimes designers cope with this by ignoring unknown relationships; they simply do not factor them into the decision.

This tactic is only acceptable when the design variable under consideration has little impact on system utility. Using cost as the example attribute, a design variable's effect on cost can only be ignored if small changes in the variable have relatively little effect on cost, or if cost has little effect on utility. The parameter can be ignored only if the *marginal utility* of changing the variable is small. This is true even if designers do not know the exact relationships. As long as the variables stay away from the feasible boundaries of existing technology—knowledge that is usually implicitly defined in the head of the engineers—then this marginal *cost* of the parameter will be small. This allows designers to base decisions on other, possibly more important attributes. Modeling and simulation can be used to support the exploration of these other attributes when making tradeoffs.

An example illustrates this point. Assume that the designers are considering three target values for the time it takes the train's doors to close: one second, two seconds, and five seconds. The costs and operational performance for each alternative are coupled through the design variable *closing time*. In this example, a closing time of around 5 seconds is clearly feasible with standard actuators, because we've all seen it done before. The marginal cost of reducing this to 4 seconds or even 2 seconds is relatively low. However, a 1 second closing time is pushing the boundaries. A reduction from 2 seconds to 1 second might require costly development of new actuators and sensors in order to operate safely; the marginal cost is probably high. Designers can then use simulation models to determine if a requirement of 2 or 5 seconds (and their likely lower costs) is operationally sufficient or whether the faster 1 second (and

likely much higher cost) closing time is necessary. While the cost knowledge is still implicit, the use of modeling and simulation makes the operational knowledge explicit. This benefit is currently limited because the uncertainty in the model is not adequately represented.

Current treatment of uncertainty in simulations

Because analysts often fail to recognize the existence of epistemic uncertainty, current practice in discrete event simulations is to use probability density functions to represent all uncertainty. However, the representation of epistemic uncertainty as probability distributions implies knowledge that is in reality unavailable. For example, currently analysts would probably represent the likelihood that a train operator fails to take a preventative action by using probability distributions. However, we do not really know how someone will act because we do not fully understand the intricacies of the human brain. There is epistemic uncertainty in human behavior that results from the hidden phenomena that are not included in the model of the operator [13].

To some extent, nearly everything involves epistemic uncertainty at some level. There is almost always something that we do not know about a system. In many cases things appear completely random, but we cannot be sure that there is not some underlying order that we do not yet understand. For example, the failure rate of mechanical components is often represented as a random process. If we could fully analyze every component, we might find the underlying causes of the differences in failure rates, but this is impractical for many applications due to the number of parts in a system. It is much more practical to treat failures within a population of components as a random process.

THE ROLE OF MODELING AND SIMULATION DURING SYSTEM DECOMPOSITION

We have so far introduced the systems engineering approach to design. We have identified different types of attributes that designers must consider, and we have discussed some of the limitations of the decomposition approach. In this section, we present four ways in which modeling and simulation can help designers overcome challenges in the design process.

The four challenges we discuss are:

- Lack of knowledge about the system's requirements
- Lack of knowledge about the system's environment
- Lack of knowledge about future design decision
- Lack of knowledge about emergent attributes

Lack of knowledge about system's requirements

At the start of the design process, the designers do not know what the stakeholders really want. However, a good understanding of the true needs of stakeholders is crucial for a successful design project. The first step in the Vee model of systems engineering (Figure 1) is *understanding user requirements*. Initially, the designers lack knowledge of these requirements.

The management approach of quality function deployment (QFD) shares the belief that products should be designed to reflect customer desires, and the house of quality serves as a conceptual map that provides a context for communication and

planning [16]. However, it is a major challenge to discover what a stakeholder considers *quality*. Often stakeholders only know quality when they see it in the design.

In order to reduce uncertainty about requirements, designers can model the requirements in more detail. Executable specifications can serve as a prototype of the specifications that is accessible to all concerned and active parties, giving stakeholders and engineers a *touch-and-feel* experience [17] in which they can interactively *explore* the design. By using executable specifications interactively with stakeholders, designers can demonstrate early in the design process what they believe the stakeholders want. The designers can work with the stakeholders to improve their knowledge about system requirements.

The stakeholders can assess the specified operational behavior at this high level, decide if they are pleased with the operation, and even clarify their true needs. They can decide for themselves the quality of the initial specifications and give feedback to the designers. The designers can discover subtle conceptual errors and validate that the system will do what it is intended to do [18]. This process of ensuring that the early specifications match the stakeholders true needs is called conceptual and requirements validation [4].

More generally, interactive simulations are a terrific means of communication, not only between engineers and stakeholders who are not necessarily trained in engineering, but also across different engineering disciplines. Executable specifications can bridge the major communication gaps in the process of identifying requirements and translating them into product specifications. By improving communication, executable specifications improve the knowledge flow between designers and stakeholders, or between systems engineers and discipline engineers. Executable specifications complement QFD and other quality assurance methods. Used together, they help the stakeholders and designers see whether the design process is starting and continuing down a path that will meet stakeholder requirements.

Lack of knowledge about the system's environment

The environment in which a system must operate is not always fully known at the start of the design process. This is especially true when the system creates or changes its own environment. For example, the exact number of passengers who will ride the airport train is not known before the system is built. Before the system is implemented, no one rides it. As soon as it is built, people change their behaviors, and at least some people will ride the train. Designers may have a rough idea of the future passenger loads, but they cannot eliminate the uncertainty completely. What designers can do is to use models and simulations to study the sensitivity of the system's operation to these uncertain environmental factors.

We approximated passenger arrivals as a random process. This is an example of using a representation of aleatory uncertainty to approximate epistemic uncertainty. Designers do not know how passengers will act in the future. Even if the process is truly random, designers do not know the nature of the process, such as true mean, exactly. We assumed that arrivals are a Poisson process and explored rates with means of 1, 3, and 10

seconds. The simulation revealed a near linear relationship between mean arrival rate and maximum queue length.

Similar trials could be repeated in a formally designed experiment (possibly based on design of experiments [19] and robust design [20]) with different values for other system parameters, such as the time allowed for boarding or the time it takes the trains to move between the stations. The system's emergent behavior under different conditions would be difficult, if not impossible, to explore interactively without a simulation model. The simulation results indicate the system's sensitivity to various environmental parameters. This information helps designers to design a system that is robust to this uncertainty.

Lack of knowledge of future decisions

A design's description is inherently incomplete at the beginning of the design process. Essentially, designers cannot know their future decisions before they are taken. This necessary incompleteness of the system description at the time the system is modeled is a form of epistemic uncertainty. These future design decisions are not a stochastic process that can be represented by a probability distribution. For example, what is the probability that a particular gear will be used in the door opening mechanism? In most cases, no such distribution can be known.

Certain final design parameters and attributes are well known because they are requirements. A model of these will match the final product (with probability near one) because the product is *required* to match the model, and a successful product design meets the requirements. In this way, designers can use executable specifications to reduce uncertainty. When designers take decomposition decisions, they do not know what the system will *look* like. However, they do know how it will *behave*, because they are specifying the intended behavior. If the system is built according to its requirements, the system will behave as specified. This behavior can be modeled without knowledge of the system's fully detailed design.

Assuming the specified system is realizable, feedback can ensure that the specifications are met under normal use cases. For example, if the specifications state that it takes the doors five seconds to close, feedback can ensure the train will not depart until either the doors are closed or after five seconds, whichever is longer. This removes uncertainty of the lower bound.

Lack of knowledge about emergent attributes

Stakeholders and design engineers can explore the effects of their requirements and decisions on the system's emergent behavior using executable specifications. The executable specifications include behavioral relationships that could be difficult to understand without simulation. For example, the simulation could help with the following roles:

- Reveal contradictions, timing problems, or deadlocks in the control logic and operation
- Verify that the system as specified satisfies the nominal (intended) system behavior
- Explore the operational impact of the decision about design parameters and subsystem requirements

Revealing problems in operation. Executable specifications can help identify the necessary coordination of various functions, such as timing and control rules for the system. For example, the system will need to guarantee that the *move people* and *open door* (Figure 3) functions never occur simultaneously, as this would be a major safety hazard. While this constraint is rather obvious, other interactions may not be.

Another potential problem with the systems operational control is competition for resources. For example, a wireless network has a limited bandwidth. Average network needs may be well below this level. However, there may be particular scenarios in which many network components request bandwidth simultaneously and exceed the available bandwidth. If the network is part of an emergency service agency, this may happen when there is a disaster that demands significant coordination of emergency resources. Such resource competition can be identified with modeling and simulation.

Verifying behavior. Executable specifications also help designers to verify that the current specification can yield the intended system behavior. This is similar to the conceptual validation discussed earlier, but this verification can continue farther into the design process. The model can indicate if particular errors were made in the design process. For example, if a subsystem requires a particular input from the environment and that input is not defined for the subsystem's parent level, then either the model or subsystem design needs to be corrected.

Exploring decisions. In the context of our example, designers can use the executable specifications to explore how different values for design parameters affect the operational behavior. This is especially important when the attributes in a tradeoff or two subsystems are strongly coupled.

The use of executable specifications will not eliminate uncertainty, but it can help designers to estimate the effects of their decisions on attributes better. For example, we simulated the train system for two and three trains, with all other parameters held constant across the trials. The results of the simulation estimated that the addition of a third train reduced the maximum queue lengths by about 30 percent. Designers could use such estimates of the operational behavior to make better design decisions.

DISCUSSION

The systems engineering approach to design involves clarifying user requirements, decomposing and defining the system, completing detailed component designs, and integrating and qualifying the system. Under conditions of perfect information and uncoupled subsystems, this process would be relatively simple. In reality, subsystems cannot always be decoupled, and the design process involves significant amounts of uncertainty.

There are great opportunities for use of modeling and simulation in systems design. Models can help designers to handle and, in some cases, reduce uncertainty in the design process. With this increased knowledge, designers can make better decisions including more appropriate tradeoffs between attributes. However, some factors still prevent the true value of modeling and simulation from being realized. We have organized our discussion into the following topics:

- Limits to uncertainty reduction
- Need for uncertainty representations and propagation methods
- Predictive versus prescriptive models
- Types of models for systems design
- Integration of models
- Validity of models
- Communication of complex ideas
- Verification of our ideas

Limit to uncertainty reduction. Theoretically, epistemic uncertainty can be reduced through additional study. For example, going to a more detailed model allows the designers to reduce the bounds on the uncertainty. The higher fidelity and more complete models also allow for errors in the higher level representation, such as missing inputs or outputs, to be identified and corrected. They also enable more complete exploration and communication. However, at some point it is impractical to reduce uncertainty by further study or more detailed modeling because the costs will significantly outweigh the benefits. The motivation for modeling is to abstract the essence of a system, not to reproduce the system in detail.

Early in the system design process, fully detailed modeling is actually impossible. Design knowledge is inherently incomplete during system decomposition because there are still decisions to be taken, and designers cannot know *a priori* what the final design will be [15]. Despite the ability of modeling and simulation to help designers reduce uncertainty, it will always exist.

Need for uncertainty representations and propagation methods. Given the presence of inherent uncertainty during design, designers need formal ways of representing this uncertainty. Designers face both aleatory and epistemic uncertainty and therefore must recognize both types and consider them individually [13]. Aleatory uncertainty can be representing using classical probability theory and probability distributions. In some cases, epistemic uncertainty can be approximated by probability distributions, but, in general, it is unacceptable to make this approximation. Instead, other techniques should be used.

Formal approaches for representing and making decisions under epistemic uncertainty are current research topics. Researchers have explored several alternatives to classical probability theory, including intervals, convex sets [21], possibility theory [22], fuzzy set theory [23], Dempster-Shafer evidence theory [24], probability-bounds analysis [25, 26], interval analysis [27] and information gap decision theory [21]. Future research should seek to relate these approaches specifically to modeling during systems design. A related question is how to compute with both types of uncertainty and compare results appropriately.

Since practical methods for dealing with both epistemic and aleatory uncertainty are currently still missing, we suggest continuing the current practice of representing all uncertainty using probability density functions, even if such an approximation may lead to conclusions that are overly confident. The designers should however be made aware of this caveat so that they can rely on implicit knowledge to judge

the quality of the simulation results. Until better tools for uncertainty quantification have been developed, using the above approximation is still desirable over ignoring the uncertainty all together.

Predictive versus prescriptive models. We have advocated the use of executable specifications during systems decomposition. Designers should think about executable specifications differently from how they may be used to thinking about simulations. Simulations are usually used as *predictive* models. Executable specifications are in one way predictive of the operational attributes that will result from the specified behavior. Executable specifications are also a *prescriptive* model of how the system should behave once built. The results of the executable specification simulations represent the behavior of the system subject to the feasibility of building the system as specified. The feasibility of the specification must be studied using other techniques, and such studies currently still rely greatly on the implicit knowledge and experience of the design team.

Types of models for systems design. The iterative, hierarchical nature of systems engineering requires that a model at an appropriate level of abstraction be available for each of the levels of detail in the design description. Discrete event simulations give direct insight only into operational attributes. They focus on the flow of signals, the duration of functions, and the sequence of functions. Subsystems can also be coupled through their energy flows. Therefore, designers should also model the energy exchange between simultaneously operating functions using ordinary differential equations or differential algebraic equations. These models help designers analyze the emergent attributes relating to energy flow. Finally, to model the complex nonlinear behavior of individual components, finite element analysis (FEA) is used. Finite element models are necessary to model how a component's behavior emerges from complex interactions within the component. For example, in a spring, the molecules interact with their neighbors in a way that gives the spring its characteristic stiffness and restorative abilities. Designers can use FEA based on Hooke's Law applied to each finite element to find a global spring constant that describes the spring's system behavior.

Integration of models. In general, engineering models are not integrated. The models for different types of attributes are separate. There is an even bigger gap between the types of models used in conceptual design and those used in the later phases of design. In software engineering, this gap is smaller, thus enabling the *transformational approach* to software engineering in which the executable specifications can be gradually replaced with the final code as portions of the final code are completed [11].

An open research question is: What are the fundamentals for an analogous *transformational approach* to engineering design of physical systems? This is a challenge because the coupling between the models and products is not obvious for physical systems. A formal representation of the information and knowledge in models at different levels of abstraction may enable designers to move smoothly between the types of models, or even use them simultaneously in a consistent, integrated manner. At a minimum, a formal representation of

uncertainty would at least allow designers to compare the results with an acceptable amount of confidence.

Validity of models. An additional problem is that a model can never be proven to be valid [28]. In the context of design where experimental validation is infeasible, the predictive value of a model will always rely on the modeler's ability to anticipate all the physical phenomena and interactions that significantly impact a system's behavior. For complex systems with emergent attributes, it may be difficult to identify all the interactions that are important, especially for attributes that depend on the system behavior in exceptional circumstances (such as reliability analysis). Developing systematic methods for creating valid predictive models at different levels of abstraction is therefore an important research issue that still remains to be addressed.

Communication of complex ideas. Because engineering systems designers are distributed across geography, time, cultures, disciplines, and nomenclatures, communication during systems design can be quite challenging. Before the use of modeling and simulation is expanded in systems design, the question as to whether it mitigates or compounds this communication problem must be answered. We believe that modeling and abstraction reduce the amount of information that needs to flow between teams of designers. For example, in an executable specification, the team designing one function block often can look at all of the other blocks as black boxes with well defined inputs and outputs. Even in cases with significant interactions, the designers of one subsystem can use the simulation to see how their decisions affect the system's attributes at higher levels of abstraction without understanding the individual implementations of other subsystems. In summary, as long as the interfaces in the models are clearly defined and uncertainty is rigorously represented, the use of models reduces the amount of information transfer between design teams.

Verification of our ideas. At this time, we have not completed a detailed case study of these ideas. There are many references available on the use of the systems engineering approach [4, 5] and the Vee model [6, 7]. Simulation models of function structures are implemented in several software packages, such as CORESim [29] or Statemate [30]. We recognize that this paper is a starting point. We have identified particular roles for modeling and simulation in this established process, as well as existing limitations. Future work will include not only verifying the suggested roles for modeling and simulation, but also creating formalisms for dealing with uncertainty in the models so that they can be used appropriately in these roles.

SUMMARY

Systems engineering relies on a hierarchical decomposition of a system into subsystems. Decisions during the decomposition result in a flowdown of requirements through the hierarchy and an upward flow of different types of attributes. In this paper, we have identified the different types of system attributes that need to be estimated during system decomposition and have surveyed the role of modeling and simulation in that process.

Modeling can help overcome some of the obstacles to the decomposition process such as coupling and uncertainty. Modeling allows for a more certain and accurate specification

of the originating requirements, a more complete exploration of possible operational architectures, better analysis of emergent attributes, more accurate predictions of subsystem attributes, and design for robustness to various environments. Together these applications allow designers to make better tradeoffs in design decisions.

However, the current treatment of uncertainty in these models limits their usefulness. Designers must recognize and treat appropriately both aleatory and epistemic uncertainty. We expect that a more formal representation of uncertainty and corresponding tools for simulation under uncertainty will have a dramatic impact on the fields of systems engineering and engineering design.

ACKNOWLEDGMENTS

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. Additional support is provided by the G.W. Woodruff School of Mechanical Engineering at Georgia Tech.

REFERENCES

- [1] Wiese, P. R., and John, P., 2003, *Engineering Design in the Multi-Discipline Era: A Systems Approach*, Professional Engineering Publishing Limited, London.
- [2] Hoffman, D. R., 1998, "Overview of Concurrent Engineering," *Proceedings of the 1998 Reliability and Maintainability Symposium*, Jan 19-22 1998, IEEE, Piscataway, NJ, USA, Anaheim, CA, USA, pp. 1-7.
- [3] Pahl, G., and Beitz, W., 1996, *Engineering Design: A Systematic Approach*, Springer Publishing, London.
- [4] Buede, D. M., 2000, *The Engineering Design of Systems: Models and Methods*, John Wiley & Sons, Inc., New York.
- [5] Blanchard, B. S., 2004, *Systems Engineering Management*, John Wiley and Sons, Inc., Hoboken, NJ.
- [6] Forsberg, K., and Mooz, H., 1992, "The Relationship of Systems Engineering to the Project Cycle," *Engineering Management Journal*, 4(3), pp. 36-43.
- [7] Forsberg, K., Mooz, H., and Cotterman, H., 2000, *Visualizing Project Management: A Model for Business and Technical Success*, Wiley, New York.
- [8] 1999, *Parametric Costing Handbook*, Defense Logistics Agency, Defense Contract Management Command and Defense Contract Audit Agency, <http://www.ispa-cost.org/PEIWeb/newbook.htm>.
- [9] Rao, S. S., 1992, *Reliability-Based Design*, McGraw-Hill, Inc., St. Louis.
- [10] Sage, A. P., 2003, "Conflict and Risk Management in Complex System of Systems Issues," *IEEE International Conference on Systems, Man and Cybernetics*, 2003., 4, pp. 3296-3301 vol.3294.
- [11] Gaskell, C., and Phillips, R., 1994, "Executable Specifications and CASE," *Software Engineering Journal*, 9(4), pp. 174-182.
- [12] Winton, I. A., 1994, "Multiple Domain Experience in Systems Engineering Using an Executable Specifications Tools," *14th Digital Avionics Systems Conference*, pp. 56-63.
- [13] Parry, G. W., 1996, "The Characterization of Uncertainty in Probabilistic Risk Assessment of Complex Systems," *Reliability Engineering and System Safety*, 54(2-3), pp. 119-126.
- [14] Oberkampf, W. L., DeLand, S. M., Rutherford, B. M., Diegert, K. V., and Alvin, K. F., 2002, "Error and Uncertainty in Modeling and Simulation," *Reliability Engineering and System Safety*, 75(3), pp. 333-357.
- [15] Antonsson, E. K., and Otto, K. N., 1995, "Imprecision in Engineering Design," *Journal of Mechanical Design*, 117, pp. 25-32.
- [16] Hauser, J. R., and Clausing, D., 1988, "The House of Quality," *Harvard Business Review*, 66(3), pp. 63-73.
- [17] Arano, T., Chang, C. K., Mongkolwat, P., Liu, Y., and Shu, X., 1993, "An Object-Oriented Prototyping Approach to System Development," *Seventeenth Annual International Computer Software and Applications Conference*, Phoenix, AZ, USA, pp. 56-62.
- [18] Harel, D., 1992, "Biting the Silver Bullet: Toward a Brighter Future for System Development," *Computer*, 25(1), pp. 8-20.
- [19] Neter, J., Kutner, M. H., Nachtsheim, C. J., and Wasserman, W., 1996, *Applied Linear Statistical Models*, WCB McGraw-Hill, St. Louis.
- [20] Taguchi, G., 1987, *System of Experimental Design: Engineering Methods to Optimize Quality and Minimize Cost*, UNIPUB/Kraus International Publications, Dearborn, Michigan.
- [21] Ben-Haim, Y., 2001, *Information Gap Decision Theory*, Academic Press, London.
- [22] Dubois, D., 1988, *Possibility Theory*, Plenum Press, New York.
- [23] Klir, G. J., 1997, *Fuzzy Set Theory: Foundations and Applications*, Prentice Hall, Upper Saddle River, NJ.
- [24] Yager, R. R., Kacprzyk, J., and Fedrizzi, M., 1994, *Advances in the Dempster-Shafer Theory of Evidence*, John Wiley and Sons, New York.
- [25] Tucker, W. T., and Ferson, S., 2003, "Probability Bounds Analysis in Environmental Risk Assessment," *Applied Biomathematics*, Setauket, New York.
- [26] Ferson, S., 2000, "Probability Bounds Analysis Solves the Problem of Incomplete Specification in Probabilistic Risk and Safety Assessments," *Ninth Conference on Risk-Based Decisionmaking in Water Resources*, Santa Barbara CA, USA.
- [27] Ferson, S., and Ginzburg, L., 1996, "Different Methods Are Needed to Propagate Ignorance and Variability," *Reliability Engineering and System Safety*, 54(2-3).
- [28] Malak, R. J., Jr., and Paredis, C. J. J., 2004, "On Characterizing and Assessing the Validity of Behavior Models and Their Predictions," *ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Salt Lake City, pp. DETC2004-57452.
- [29] CORE, Vitech, <http://www.vtcorp.com/>.
- [30] Statemate, I-Logix, <http://ilogix.com/>.