Getting Design Right: Systems Engineering for the Non-Engineer

Peter L. Jackson
Cornell University
Systems Engineering Program
Rhodes Hall 206
Ithaca, NY 14853
Telephone: (607) 255-9122
pj16@cornell.edu

Abstract. Our goal is to disseminate the systems engineering process to as broad an audience as possible. This audience includes freshmen engineers, students from non-engineering majors, as well as working managers and staff from a host of different occupations. Reviewing the impediments to the dissemination effort and the success of the Six Sigma movement, we articulate the requirements that a curriculum for non-engineers should satisfy and we propose a particular blended curriculum that satisfies these requirements and highlight its features. We point to three implementations of the curriculum. Experience with this form of the curriculum is still in its infancy.

Introduction

Motivation. Systems engineering professionals frequently express the wish that more people were familiar with the systems engineering process. This sentiment most often seems to surface in reference to their managers and colleagues but the benefits of the process seem so self-evident to them that they see its application in all walks of life. The basic approach of defining a problem by the requirements a solution must satisfy, systematically capturing the relationships between entities in the problem and solution domains, tying tests to requirements, exploring the design space, making the difficult trade-off decisions, defining the interfaces, and using divide and conquer approaches to handle complexity… this approach seems so natural to them that it is frustrating to see that it is not more broadly applied. In academia, after decades in which mathematical analysis and engineering science prevailed in engineering curricula, design and synthesis is now enjoying resurgence. Systems engineering programs and faculty can make an enormous contribution to this revitalization of engineering curricula but only to the extent that the approach is not seen by other faculty as an isolated discipline. The challenge addressed in this paper is how to achieve a greater dissemination of the systems engineering process. The target audience is the non-engineer, or, at least, the non-systems engineer: anyone who would benefit from a basic understanding of the systems engineering process. This audience includes freshmen engineers, students from non-engineering majors, as well as working managers and staff from a host of different occupations.

Requirements for Reaching a Non-Engineering Audience

Impediments to Dissemination. It should be confessed by the systems engineering community

that one of the main impediments to broader dissemination of the systems engineering process is our focus on complexity. The desire for the intellectual respect of our colleagues from other disciplines leads us to emphasize the difficulty of the problems we tackle. The power of the systems engineering process in managing the design of complex systems and our own acquired comfort with the analytical tools and databases for managing complexity lead us to introduce the discipline with its most intimidating aspects first.

Like many of us, I used to describe systems engineering as the process used to design complex systems. Let us reject that definition. Systems engineering is the process by which we understand a complex need, design elegant and harmonious solutions to meet that need, integrate those solutions with solutions to related needs, and marshal the people and resources to build, test, and deploy those solutions. We do not set out to make our systems complex. It is the needs that are complex. We achieve elegance when the solution appears simpler than the need.

Not only do we emphasize the complexity of the problems, but we frequently emphasize the complexity of the process as well. That may be an acceptable pedagogy with systems engineering students; after all, those are the aspects that have likely attracted them to the discipline in the first place. For the non-engineer, however, complexity is the wrong place to start.

There are other impediments to dissemination. The term “engineering” itself is a barrier with the target audience. Engineering has long been associated by the general public with requiring prowess in mathematics and science. But, advanced mathematics is not required for understanding the basics of the systems engineering process. That fact underlies our belief that it can and should be more broadly disseminated and practiced. We may, therefore, have to shed the label “engineering.” Pride of discipline may have to be sacrificed for mission success.

Another impediment is the document-centric nature of the systems engineering process in practice. The volume of documentation supposedly needed as evidence to demonstrate that the systems engineering process has been followed strikes the non-engineer as tedious, wasteful, and uninteresting. The thrill of design has been replaced with the tedium of paperwork.

The Joy of Systems Engineering. In taking on the challenge of disseminating the systems engineering process, we should reflect on what it is that motivates us and excites us about our discipline. The joy of systems engineering is that we do offer an approach to solving problems that is robust, broadly accessible, broadly applicable, and easily communicated. It is the simplicity of the underlying principles that we should be celebrating, not the complexity. Systems engineering for the non-engineer must focus on those aspects of the process that have the greatest leverage: the best assurance of success for the fewest obstacles to implementation.

Systems engineering is also a skill and the non-engineer needs a roadmap and motivation to acquire that skill. Mastery of the basic skills, and, more importantly, success in their application, is another source of joy.

Another of the joys of systems engineering is the habit of systems thinking. By this we mean the mental discipline of identifying a system, its context, its purposes, its entities, the relationships among its entities and between its entities and the outside world, its self-regulating behaviours, and so on. This mental discipline requires abstraction and this can be taught. It is therefore, a joy to be shared.

More fundamentally, the joy of systems engineering is the joy of design, of finding elegant and

harmonious solutions to all manner of problems. It is essential therefore, to view the techniques of systems engineering not as drudgery but as techniques of discovery: discovering needs, discovering requirements, discovering creative alternatives, discovering risks, and so on.

Lessons from the Six Sigma Movement. The systems engineering discipline has much to learn from the success of the quality improvement movement in manufacturing and related industries. Like systems engineering, various implementations of this movement, such as Six Sigma, have emphasized a problem-solving methodology, tracing back to the Plan-Do-Study-Act cycle of Deming and Shewhart (Deming, 1993). Unlike systems engineering, however, the quality improvement movement emphasized dissemination of the methodology throughout the workforce. Quality of product was no longer viewed as the responsibility of the Quality Control department; quality of process was the responsibility of every machine operator. For companies implementing the quality improvement processes, it meant a substantial commitment to training and education. But for the educators, it meant a stripping down of the mathematics of statistical process control to the bare minimum required for success in the hands of a high-school-educated operator. The problem-solving methodology itself was expressed in simple steps.

Consider one example of a Six Sigma process: DMAIC (for Define, Measure, Analyze, Improve, and Control). Table 1 lists the sub-steps involved in each of the major steps of the process. Notice the emphasis on verbs: "identify, prioritize, develop, plot, design," and so on. A comparison with the systems engineering literature is likely to find a much greater emphasis on nouns and adjectives in our papers than on action-oriented verbs. Promulgating the use of the systems engineering process will require leadership and leadership requires the use of action-oriented verbs.

Six Sigma has achieved remarkable penetration both domestically and abroad. Many dozens of companies have claimed to implement it in some form. The challenge for systems engineering is to both maintain its strength as an engineering discipline while disseminating its basic approach.

Table 1: DMAIC Activities (Source: iSixSigma.com)

Define	Measure	Analyze	Improve	Control
Identify objectives	Identify input, output, and process	Stratify process	Design of experiments	Verify reduction in root cause
Identify customers	Develop operational definition and measurement plan	Stratify data	Response surface methods	Are additional solutions necessary?
Identify customer needs and requirements	Plot and analyze data	Develop problem statement	Generate solution ideas	Identify and develop replication and standardization procedures
Identify quality characteristics	Cause and effect analysis	Identify root causes	Determine solution	Integrate and manage

			impacts	solutions in daily work
Prioritize characteristics (Critical to customer)	Failure modes and effects analysis	Design root cause verification analysis	Evaluate and select solutions	Integrate lessons learned
Create a process map	Identify key inputs	Validate root causes	Communicate solutions	
	Identify key process steps	Sources of variation studies	Develop pilot plans	
	Business process charting to track project metrics	Regression analysis	Verify critical inputs	
	Collect baseline performance data	Design of experiments	Optimize critical inputs	
		Process control		
		Process capability		

Blending Systems Engineering with Six Sigma. It is worth asking whether or not systems engineering, when reduced to an elementary problem-solving methodology, is distinct from Six Sigma or Design for Six Sigma (DFSS). Any rational, systematic approach to problem-solving will naturally exhibit a number of common features. It is not surprising, therefore, to find a great deal of overlap between the systems engineering process, the various Six Sigma methodologies (DMAIC, DMADV, IDOV, and DFSS), stage-gate product development processes, Department of Defense system acquisition programs, and general problem-solving techniques. In fact, one of the first assignments we typically give to new systems engineering students is to take tables describing all these different approaches and to come up with their own six- or eight-step process to design.

While acknowledging the overlap, it does appear that systems engineering differs from the Six Sigma approaches (for example, Brue *et al*. 2003, and Yang *et al*., 2003) in its emphasis on system architecture. From its origins in manufacturing, the focus in DFSS appears to be on a decomposition of engineering characteristics, and less on the functional and structural decomposition of a system into entities that must work together. The theme of understanding a system in its context plays greater in the systems engineering approach. The theme of understanding the customer plays greater in Six Sigma. We now adopt a blended approach between the two.

Requirements for a Curriculum: With the preceding as background, we outline the

requirements we adopted in designing a curriculum for educating the non-engineer in the systems engineering process.

1. Emphasize discovery, design, problem-solving, and validation.
2. Defer discussions of complexity until after a basic design methodology has been taught.
3. Describe the basic design methodology as a design cycle of simple steps.
4. Blend the systems engineering approach with other customer-focused and product-development focused approaches.
5. Use action-oriented verbs to describe the steps.
6. Motivate each major design step with "What Went Wrong?" case studies.
7. Restrict techniques to those requiring only secondary-school level mathematics and science.
8. Illustrate each step using a running example.
9. Reinforce systems thinking with repeated opportunities for abstraction.
10. Provide design challenges in both business and engineering applications.

A Blended Approach

Getting Design Right. In naming the curriculum, we chose not to use the phrase "systems engineering" for reasons explained in the background section. Instead, we chose the phrase "Getting Design Right," in hopes that it would have a broader appeal among non-engineers. After several iterations of the curriculum, we settled on a cyclic eight-step process for design as depicted in Figure 1. As in the Six Sigma literature, each major step is broken down into action-oriented sub-steps. The approach is a blend of design steps from multiple sources: systems engineering, software engineering, Six Sigma, product design and development, and project management. Topics covered are shown in Figure 2, an annotated version of the "Getting Design Right" cycle.

Although it is not evident from Figure 2, all topics are covered at a freshman level. The analytic hierarchy process (Saaty, 1980), for example, is used to attach quantitative weights to different product objectives. What is valuable from this approach is the concept that product objectives can be nested and the process of developing weights can be developed in stages. The more complex aspects of the technique (using eigenvalues to extract weights from pairwise comparisons) are not covered.
In the next subsections, we provide a brief description of the curriculum in each of the eight major steps.

Define the Problem. This step walks the student through a process from initial conception of a need, through naming the problem and sketching the product concept, to a more detailed contextual inquiry (Beyer and Holtzblatt, 1998), and finally to a behavioral analysis (use cases and thread description) ending with a list of functional requirements. There is an emphasis on abstraction (extracting the "voice of the customer" from unorganized customer comments) and discovery (unusual functional requirements from use case analysis).

Measure the Need and Set Targets. This step considers two problems: how to measure customer

needs and how to translate vague statements of customer objectives into target technical performance measures. We resolve the first problem using the Goal-Question-Metric method (Basili and Weiss, 1984) and the second problem using the House of Quality technique (Hauser and Clausing, 1988). Since the product concept is now defined by both behavioral and non-behavioral requirements, we introduce the Customer Value Proposition as a test of whether it is worth continuing the design effort.

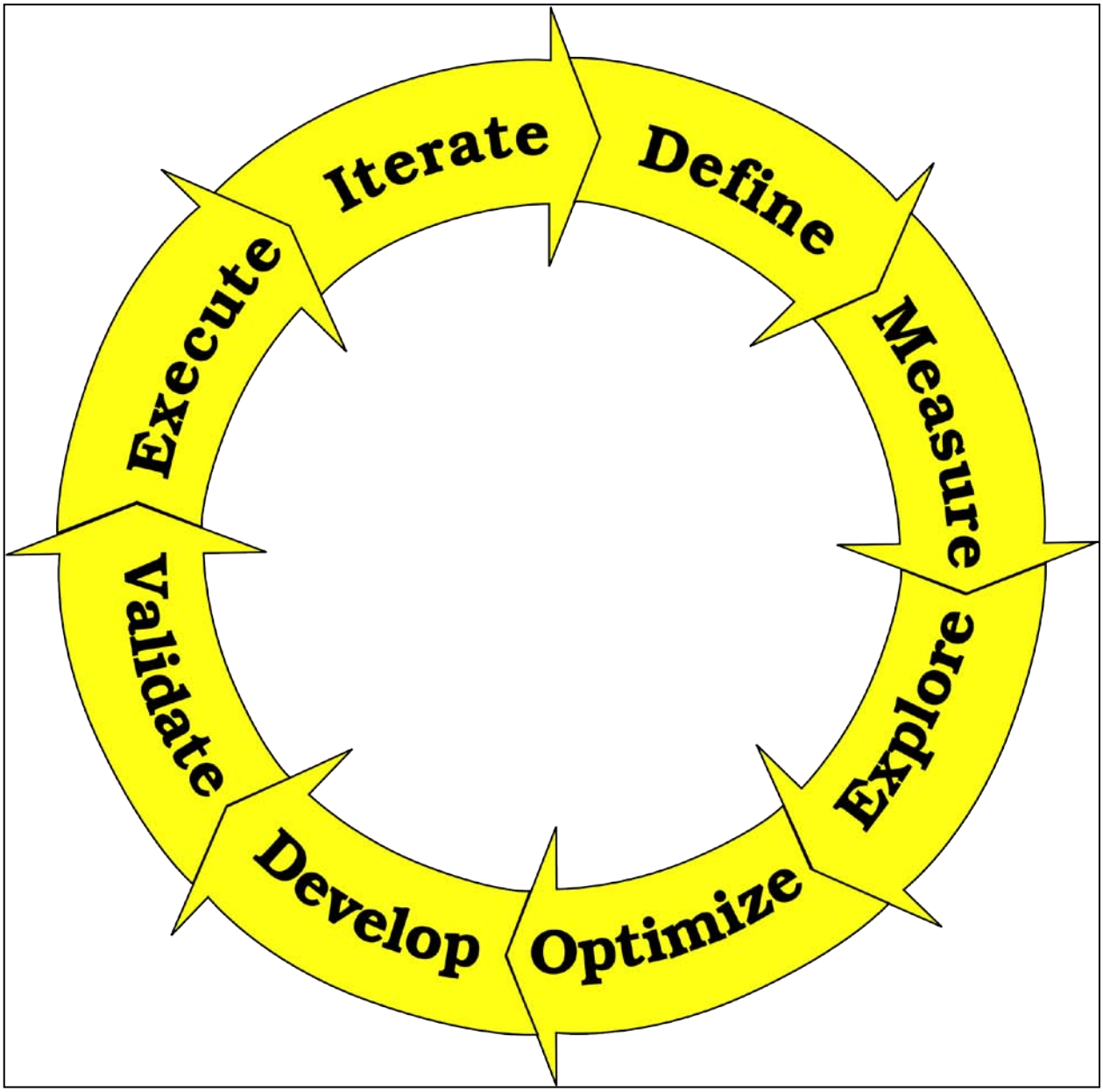

Figure 1. Eight Steps to Getting Design Right (Jackson, 2009. CRC Press)

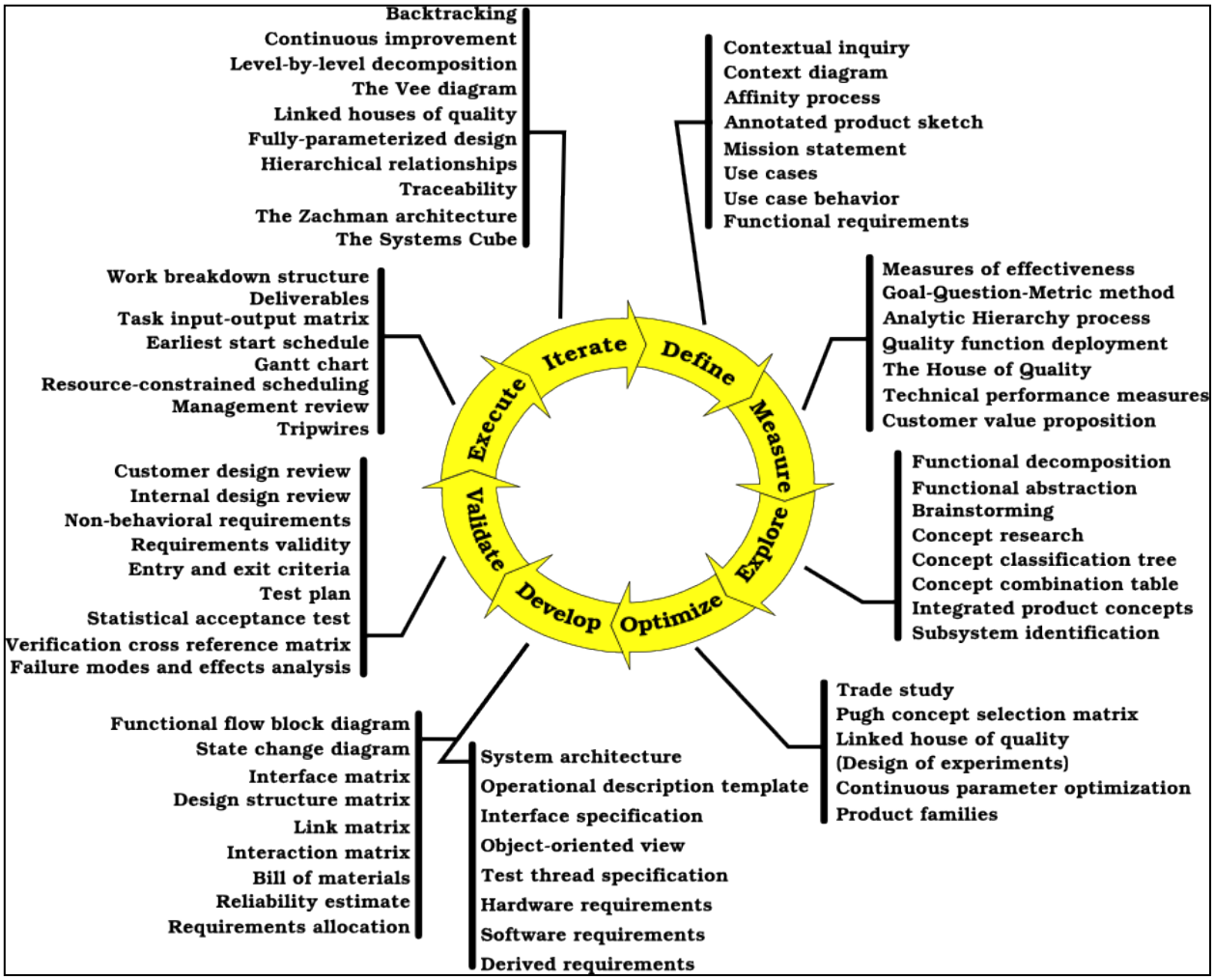

Figure 2. Annotated Cycle of Getting Design Right (Jackson, 2009. CRC Press)

Explore the Design Space. This step is taken from the product design and development literature (Ulrich and Eppinger, 1995) and uses concept classification trees and concept combination tables to suggest innovative integrated product solutions. We introduce the concept of subsystems at the end of this step. We delayed the discussion of subsystems to this point out of a concern that defining subsystems before considering a broad range of possible solutions can unnecessarily restrict the system architecture. We illustrate that radical innovations in system architecture are possible through this step.

Optimize Design Choices. In this step, we describe Pugh analysis (Pugh, 1991) as a technique to select among alternatives when there are many attributes to the choice. We also guide the student into formulating problems of design parameter optimization. The technique used here is the linked house of quality in which system requirements from the house of quality now become constraints to be satisfied at a deeper level of design. This was also a convenient step in which to introduce the concept of product families: a product platform to exploit manufacturing commonality but easily created derivative products to optimize the product for different market niches (Meyer and Lehnerd, 1997).

Develop the Architecture. In this step, the student is led to think of the product at the subsystem

level and to describe how the subsystems will work together to achieve the desired behaviour of the system. For this step, we use the operational description template (Karas, 1987, 1993, 1999, 2001) because, as shown in Figure 3, this tabular description of behaviour across multiple subsystems nicely illustrates the integration of human interface specifications, hardware and software specifications, test thread specifications, and interface specifications. From this, we can also summarize functional flow and state change transitions. A variety of N^2 matrices are used to capture system relationships. Design Structure Matrices (Baldwin and Clark, 2000) are also used to suggest subsystem redesign. We also consider the problem of allocating system level performance measures (cost and reliability) across multiple subsystems. These topics are not typically covered in the Six Sigma literature.

Validate the Design. In this step the process of discovery is extended to include design reviews and a detailed test plan for both behavioural and non-behavioural requirements. The difficulty of testing requirements becomes evident at this point and we use the opportunity to discuss what makes for a valid requirement. This step also includes creation of a verification cross-reference matrix to get a systems view of the verification process. As another check on the design process, we introduce the technique of failure modes and effects analysis (FMEA) to proactively consider the risk of design failure.

Execute the Design. This step is frequently omitted from other design curricula but we prefer a holistic view. We guide the student into collecting all the design, build, and test activities into a work breakdown structure and mapping the inputs and outputs between all the activities. The schedule can be displayed as a Gantt chart and the student adjusts the schedule to account for resource constraints. The concept of management reviews, as distinct from technical design reviews, is also covered.

Iterate the Design Process. Three forms of iteration are described in this step. The first is backtracking to find a feasible solution. Ten backtracking strategies are proposed and illustrated. The second is iterating with improvement, using lessons learned from one design cycle to improve performance on the next. The third use of iteration is iterating by level, applying the design cycle to each level of a complex design. It is here that we finally introduce the systems engineering process, with its level-by-level decomposition of requirements, function, and structure, the Vee diagram (Fossberg and Mooz, 1992), traceability of requirements, and the need for databases to track the explosion of detail.

What's Missing? What is missing from the curriculum, compared with other courses on design, are domain-specific topics, such as design-for-manufacturability (mechanical engineering) or design-for-security (software engineering). Instead, we introduce the concept of design-for-secondary-uses. This topic becomes a placeholder for the instructor to tailor the course to his or her particular expertise and interest.

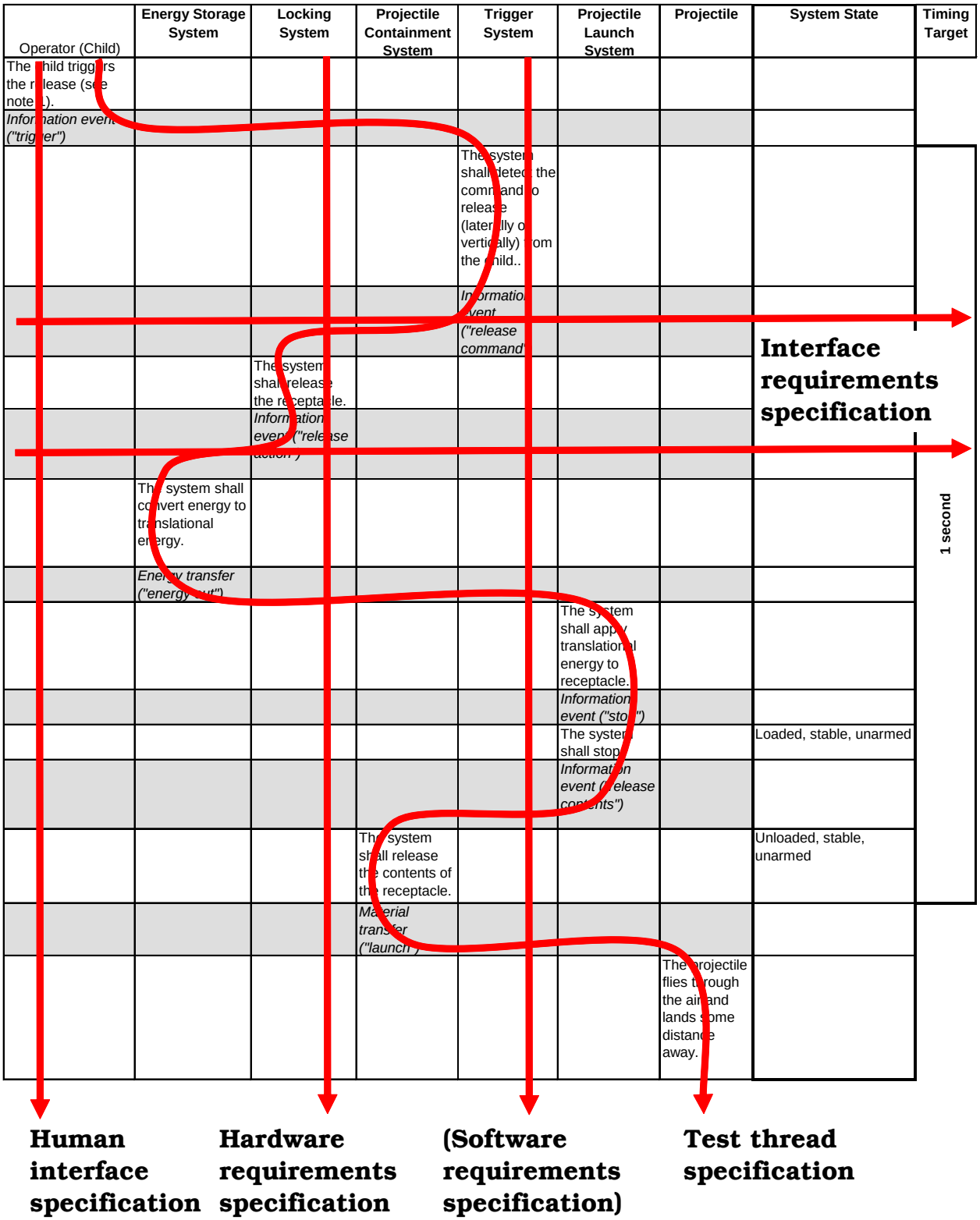

Figure 3. How the operation description template integrates specifications (Jackson, 2009. CRC Press)

Dive and Surface

One of the themes woven into the curriculum is the need to frequently surface from a collection of details and summarize them into a more abstract form. Many steps of the design process are used to illustrate this dive and surface process. For example, customer comments are summarized into the voice of the customer, functional requirements are summarized in more abstract form, random concepts are organized into a concept classification tree, components from multiple product concepts are organized into abstract subsystems, and behaviour threads are summarized by functional flow and state change diagrams. A final illustration of the process is to take all of the concepts introduced in the curriculum and summarize them in a higher view of the design domain. This takes the form of a systems cube, as illustrated in Figure 3. This is a modified version of the Zachman architecture (Zachman, 1987). The concepts fall into one of three domains, Context, System, or Designer, as described in Table 2. Seven views of a domain are described using the suggestive names “Who, Why, How, How well, What, When, and Where” and entities and relationships such as shown in Table 3. The systems cube thus reinforces the dive and surface process of abstraction and serves as an organizing framework for the information that the student has acquired.

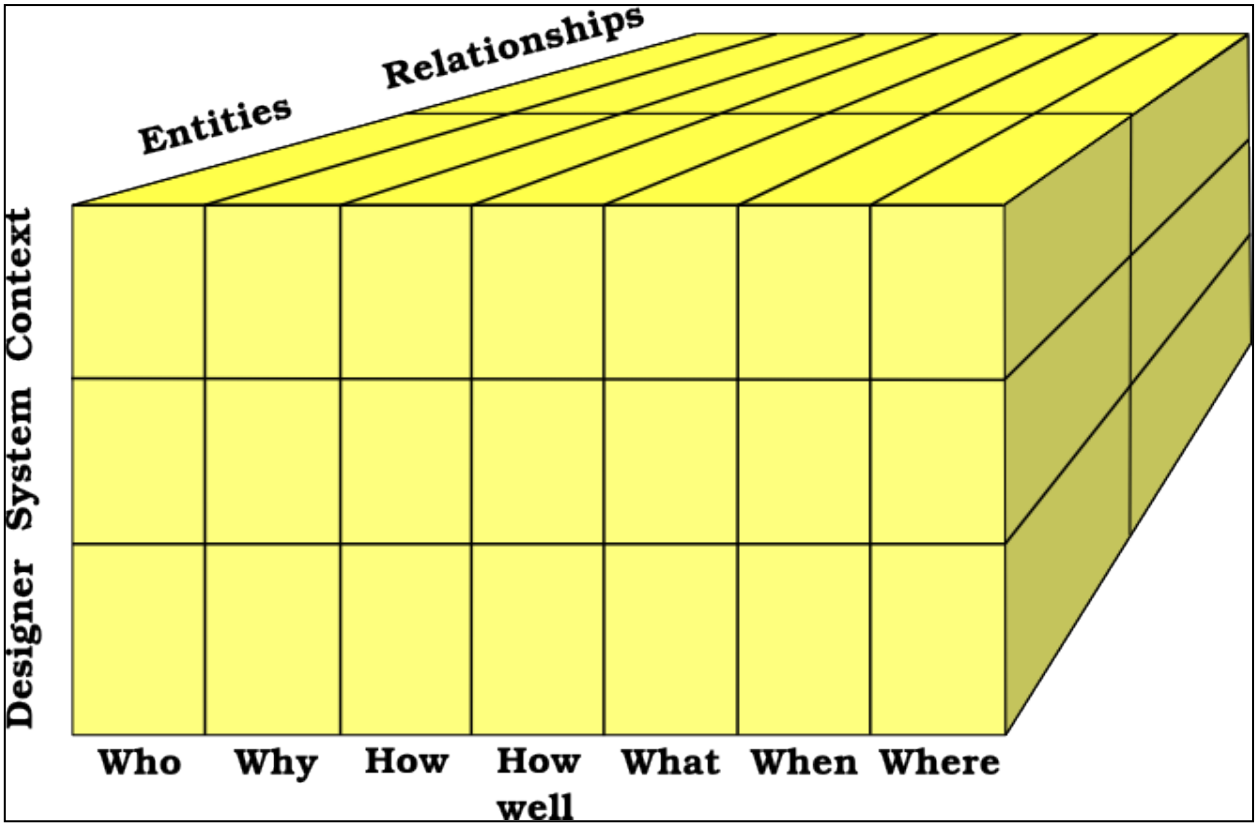

Figure 1. TheSystems Cube (Jackson, 2009. CRC Press)

Table 2: Domains of the Systems Cube (Jackson, 2009, CRC Press)

	Domain
Context	Market opportunity, competition, the value proposition, external entities, system boundary and interfaces with other entities, stakeholders, unintended users, malicious users, threats, risks, strategy, and emergent behavior
System	User and use cases, system requirements (behavioral and non-behavioral), physical architecture, functional architecture, control architecture, design specifications, and system validation
Designer	The design organization: its mission, roles, responsibilities, resources, learning and exploration processes, valuation and decision making processes, verification processes, project management, and risk management

Table 3: Views of the Systems Cube (Jackson, 2009, CRC Press)

	Sample Entities	Sample Relationships
Who	Roles, perspectives, points of view	Organization, responsibilities, reporting relationships
Why	Purpose, mission, goals, objectives, values, uses, constraints, requirements, risks	Tradeoffs, priorities, uses to behaviors, ends to means, conditions to actions (rules), requirements to requirements, requirements to functions, risks to strategies
How	Behaviors, processes, functions, states, tasks	Functions to objects, sequence, iteration, triggers, functional decomposition, inputs and outputs
How Well	Measures of effectiveness, business and technical performance measures, tests and benchmarks, chance and negative outcomes	Measures of effectiveness to engineering characteristics, tests to requirements, allocation of targets to subsystems
What	Artifacts, objects, data, properties	Physical architecture, bill of materials, interactions, interfaces, messages
When	Timing, events, tasks, durations, schedules	State transitions, precedence, tasks to resources, conflicts, gridlock, feedback control
Where	Facilities, geographical or spatial locations, infrastructure	Networks, flow, distances and clearances

Design Challenges

We have been developing product and system design challenges for our graduate systems

engineering students for a number of years. To date we have four design challenges:

- A bathroom-cleaning robot
- A home health-care monitoring station
- A night-vision system for automobiles
- An internet-based meal delivery system

Each design challenge is introduced by means of a fictional case study and a market requirements specification document. We have found that these same design challenges work well with a non-technical audience. The students select one of the challenges at the beginning of the course and then work through the application of the design process to that challenge over the course of the curriculum.

Implementations of the Curriculum

The curriculum has evolved over the years from our development of short courses to teach core practices in systems engineering to managers and engineers in industry. The current curriculum has been implemented in several forms: (1) as a distance learning summer school course for college freshmen, (2) as a distance learning certificate program for working professionals, and (3) as a text for use at other universities and corporate training facilities (Jackson, 2009). Our experience with this form of the curriculum is still in its infancy in terms of the number of students who have completed it. Early indications are that it is meeting its goals.

Conclusion

Our goal is to disseminate the systems engineering process to as broad an audience as possible. Reviewing the impediments to the dissemination effort and the success of the Six Sigma movement, we articulated the requirements that a curriculum for non-engineers should satisfy and we proposed a particular curriculum that satisfies these requirements. The curriculum has been implemented in three forms but there are no results yet to report.

References

Bass, L., P. Clements, and R. Kazman. 2003. *Software architecture in practice*. 2nd Edition. , Indianapolis, IN: Addison-Wesley Professional.

Baldwin, C.Y. and K.B. Clark. 2000. *Design rules, Vol. I: The power of modularity*. Cambridge, Mass: The MIT Press.

Basili, Victor R. and D. Weiss 1984. A methodology for collecting valid software engineering data. *IEEE Transactions On Software Engineering* (November):728-738.

Beyer, H. and K. Holtzblatt. 1998. *Contextual design: Designing customer-centered systems.* San Francisco: Morgan Kaufmann Publishers.

Brue, Greg, and Robert G. Launsby. 2003. Design for six sigma. New York: McGraw-Hill.

Deming, W. Edwards. 1993. *The new economics: for industry, government, education.* MIT Cambridge Center for Advanced Engineering Study.

Fossberg, K. and H. Mooz. 1992. The relationship of systems engineering to the project cycle. *Engineering Management Journal* 4 (3): 36-43.

Hauser, J.R. and D. Clausing, 1988. The house of quality. *Harvard Business Review* (May-June):

63-73.

INCOSE 2006 systems engineering handbook - a guide for system life cycle processes and activities, Version 3. ed. Cecilia Haskins.

iSixSigma. 2003. DMAIC definition. http://www.isixsigma.com/dictionary/DMAIC-57.htm (Downloaded November 8, 2009).

Jackson, P.L. 2009. *Getting design right: A systems approach.* New York: CRC Press.

Karas, Leonard and Donna Rhodes. 1987. Systems engineering technique. NATO Advisory Group for Aerospace Research & Development Conference Proceedings No. 417.

Karas, Leonard and Donna Rhodes. 1993. Enabling concurrent engineering through behavioral analysis. In Proceedings of the Third International Council on Systems Engineering, July.

Karas, Leonard and Donna Rhodes. 1999. The common denominator. In Proceedings of the Ninth International Council on Systems Engineering, June.

Karas, Leonard, 2001. The systems engineering environment (SEE) Tutorial. International Council on Systems Engineering Conference.

Oliver, D.W., T.P. Kelliher, and J.G. Keegan, Jr. 1997. *Engineering complex systems with models and objects*. New York: McGraw-Hill.

Meyer, M. and A.P. Lehnerd. 1997. *The power of product platforms.* New York: The Free Press.

Pugh, Stuart. 1991. *Total design: Integrated methods for successful product engineering.* Indianapolis, IN: Addison-Wesley.

Saaty, T.L. 1980. *The analytic hierarchy process*. New York: McGraw Hill.

Ulrich, K.T. and S.D. Eppinger. 1995. *Product design and development.* New York: McGraw-Hill.

Yang, Kai, and Basem El-Haik. 2003. Design for six sigma: A roadmap for product development. New York: McGraw-Hill.

Zachman, J.A. 1987. A framework for information systems architecture. *IBM Systems Journal* 26 (3).

BIOGRAPHY

Peter Jackson is a Professor in the School of Operations Research and Information Engineering, Cornell University, and was Director of the Cornell Systems Engineering Program (2003-2009). He holds degrees in economics (B.A. 1975, University of Western Ontario), statistics (M.Sc., 1978, Stanford University), and operations research (Ph.D. 1980, Stanford University). Jackson's research and consulting interests include manufacturing systems design and supply chain management. Jackson is the recipient of several awards for curriculum innovation in addition to numerous student-voted awards for teaching excellence. He is the recent author of a text used to introduce systems engineering to a non-technical audience.