Proceedings of the Fifth ISPE International Conference on Concurrent Engineering: Research and Applications, Tokyo, Japan, July 15-17, 1998.

Use of Dependency Structure Matrices for Product Development Cycle Time Reduction

Tyson R. Browning
Massachusetts Institute of Technology
77 Massachusetts Ave., Room 33-407
Cambridge, MA 02139
tyson@mit.edu

Abstract

Cycle time reduction is an important aspect of integrated product and process development *(IPPD). This paper outlines some of the challenges that impede efforts to reduce cycle time for complex system development projects. Several facets of these challenges can be summarized using the general concept of design iterations. After reviewing this concept, the paper provides an overview of a process modeling approach utilizing* dependency structure matrices *(DSMs—also known as design structure matrices). DSMs allow a simple, visual representation of processes and highlight potential iterations. The paper concludes with a discussion of the ways DSM-based methods help manage some of the cycle time reduction challenges.*

1 Introduction

Tremendous pressure on today's companies to achieve and sustain competitive advantage has led to heightened efforts to reduce product development cycle time and cost while providing superior products. Indeed, improved management of the product development process contributes to sustainable competitive advantage for companies [1-3]. Many have recognized that the key to process improvement lies in better process understanding. Towards this objective, a complex design process can be viewed as a set of simplier activities with discernible interrelationships. Choices as to the decomposition and integration of these activities have ramifications for the competitiveness of the process [4]. The realization that many product development processes and activities are procedural and repeatable has given rise to modeling efforts that seek to describe current processes and prescribe appropriate policy adjustments.

This paper highlights the capabilities of *dependency structure matrix* (DSM)[1]-based methods to manage projects for reduced schedule risk and shorter cycle time. Following a presentation of some cycle time reduction challenges, I provide overviews of the concept of design iterations and of DSM-based, process modeling techniques. In closing, I discuss how DSM methods meet the cycle time reduction challenges covered earlier. This paper's main contribution is to lay out several cycle time reduction challenges and show the ways DSM methods help meet them.

2 Cycle Time Reduction Challenges

Several characteristics of most complex system development projects make cycle time reduction especially challenging. These characteristics and challenges include:

- Inefficient distribution of personnel and resources—including facilities, tools, and especially information—results in parts of the project spending "down" time waiting for resources when they could be working. While this is inevitable given limited resources, appropriate planning can insure that at least critical path activities are not delayed due to resource constraints. Lack of and failure to use appropriate planning techniques complicates cycle time reduction strategies.
- Unstable product requirements cause indecisiveness and foster redesign. If requirements remain uncertain too late in the project, extra rework

[1] Or, *design structure matrix*

becomes inevitable and cycle time reduction becomes more difficult. Making some decisions as late as possible (a tenet of set-based design) is sometimes the right thing to do, but certain decisions need to be made early to properly set the scope and direction of a project.

- The existence of long and varied activity "pipelines" means shorter duration activities finish and wait for longer duration activities to produce needed results. When these results finally become available, the activities that finished early often have to go back and change their work based on the new information. Meanwhile, if the activities that finished early had proceeded to more detailed design work (perhaps because the work culture demands they "look busy") before the later results became available, much of this work could end up squandered.
- Lack of activity coordination results in wasted time, doing the wrong work, and inappropriate communication. If activities are not well integrated to the point of knowing what information each needs, when, and in what format, they will have trouble reducing the cycle time of their overall process.
- Highly interdependent or coupled activities increase the chances of iteration or rework. This effect is discussed in more detail in the sections that follow.
- Overly ambitious initial schedules make further cycle time reductions even more difficult. If an original plan turns out to be high risk from a schedule standpoint, reducing that schedule any further will cause even greater risk. In other words, one cannot reduce cycle time simply by shortening the planned schedule; one must insure a commensurate reduction in schedule risk.
- Many seemingly feasible cycle time reduction solutions have adverse side effects which reduce anticipated effectiveness. Thus, successful cycle time reduction requires a systems perspective, taking into account the feedbacks that will tend to diminish overall process efficacy.

Some have approached the cycle time reduction problem by simply taking once serial activities and executing them in parallel. However, doing so without paying attention to the interdependencies that may exist between such activities can actually waste resources without providing much cycle time advantage. Activities that depend on other activities for information will often have to redo some portion of their work if those inputs change or arrive late. Hence, indiscriminately placing once serial activities in parallel can lead to additional rework or iteration. A more effective approach requires a systems view of the process and the capability to analyze the complex relationships between activities. The DSM methodology meets these requirements.

Together, the challenges above account for a large portion of the uncertainty or variance that leads to overruns in project schedules. Increasing complexity and scope of the system under development exacerbates these hazards. These challenges exist because they are difficult to plan for and manage on a real-time basis. The goal of this paper is to show how the concept of design iterations and the DSM modeling methodology help address some of these problems.

3 The Concept of Design Iterations

Many of the challenges noted above are captured in the concept of iterations in the design process. Here I provide an overview of this concept. Realizing the role of iteration as a schedule risk driver and as a barrier to cycle time reduction motivates the need for tractable methods that account for iteration and help managers focus resources appropriately.

Design iteration implies rework or refinement, returning to previously worked activities to account for changes. This rework can stem from new information and/or failure to meet design objectives [5]. New information for an activity comes from:

1) Upstream activities causing an activity's inputs to change (often the result of an iteration causing a second pass through some part of the process, or a change in externally supplied requirements and/or assumptions),
2) Concurrent, coupled activities, causing an activity's inputs to change, often as shared assumptions change, and
3) Downstream activities causing an activity's inputs to change, as errors and incompatibilities are discovered. [6]

Work done later (downstream) in the process—particularly verification and validation activities—often reveals aspects of seemingly completed (upstream) activities that must be reworked [7-10]. Rework can also be generated by changes in the information provided to and received from concurrent, interdependent (coupled) activities. For example, team A needs to know what values team B has set for parameters x and y; team B needs to know what values team C is using for parameters w and z; but team C needs to know the result of team A's activities to determine w and z. These design issues are sometimes called "chicken and egg" problems, and the groups that work to solve them are called coupled teams. The phenomenon of engineers trading technical information and thereby creating rework for each other has been observed in several cases by Clark and Fujimoto [1]. Also, when downstream or coupled activities create rework for upstream activities, the resulting changes may cause second order rework for interim activities (those between the upstream and downstream activities directly involved in the iteration). Hence, input changes to activities can generate new information and force iteration in the process.

Iteration also results from not meeting the desired design objectives. Iteration has been defined as "the repetition of activities to improve an evolving design" [11]. Failure to converge to design specifications can require reworking upstream activities tied to the area(s) of shortfall. Note that such insufficiency in the design is more likely when requirements and objectives are unstable or otherwise prone to modification. Such instability results from changing emphasis on known objectives and/or the addition of new objectives.

Several studies have documented iteration effects as key drivers of overall development cycle time [2, 3, 12-18]. In particular, Osborne found that iteration accounted for between 13% and 70% of total development time for semiconductor development activities at Intel. He also found iteration to be the chief cause of cycle time variability at the firm.

The iteration perspective helps capture and quantify drivers of cost, schedule, and performance (design quality) variability in design programs. Tightly coupled, highly iterative product development processes can expect greater difficulty converging to an acceptable design under a given schedule and budget. To move towards consistent success, product development projects should strive to insure that, to a greater extent, their design iterations are strategically planned rather than the result of seemingly random chance.

In light of this objective, it is useful to divide iteration in the product development process into two categories:

1) **Intentional iterations**, purposely performed in a coupled design process to converge to a desirable solution (i.e., improve design quality to a desired level), and
2) **Unintentional iterations**, resulting from new information arriving late in the process (caused by out of sequence activities, fluid requirements or design goals, mistakes, etc.).[2]

[2] Similarly, Clausing [19] proposes the categories "creative iteration" and "disciplined iteration" as intentional and "dysfunctional iteration" as unintentional.

The first step towards reducing product development cycle time and variation lies in minimizing unintentional iterations. This consists of insuring that the right information is available at the right place at the right time; activities are properly sequenced, given relevant constraints; resources are available to the activities when needed; requirements are firm as quickly as possible; and mistakes are minimized.

The next step towards accelerating the design cycle involves two basic options for managing intentional iterations:

1) **Faster iterations**, and
2) **Fewer iterations**. [20]

Faster iterations can be achieved by, e.g., CAD systems that accelerate individual activities, simulation and analysis tools that reduce dependence on time-consuming prototype/test cycles, improved intragroup coordination for teams assigned to individual activities, concurrent engineering or IPPD, integration of engineering analysis tools used for disparate activities, and removal of extraneous activities from the process [20]. A study by Eisenhardt and Tabrizi [18] and a model by Gebala and Eppinger [13] find that additional (because they are faster) iterations are significantly correlated with faster product development. Fewer iterations may be possible by, e.g., improved activity sequencing, improved interteam (interactivity) coordination, co-location of those executing highly interdependent activities, and improved assumptions (learning) about other activities. However, fewer iterations can mean less design quality, or at least a greater risk that the design will not converge to the desired targets. Faster iterations reduce this risk, since more iterations improve the chance that the design will converge to acceptable multiattribute performance levels (all else being equal). In both cases, however, the quality, gain, or *productivity* of each iteration is important [21]. Fewer iterations might make sense if each one is of sufficient productivity to ensure acceptable output. Faster iterations will only be advantageous if each activity can be accelerated while continuing to produce satisfactory outputs.

4 Introduction to DSM-Based Process Modeling Methodology

This section provides an extremely brief overview of the origins of DSMs and the keys to DSM methodology. DSMs can trace their development to several sets of roots. Perhaps the primary source of DSM concepts grows from efforts to solve systems of equations in the late 1950s and 1960s.

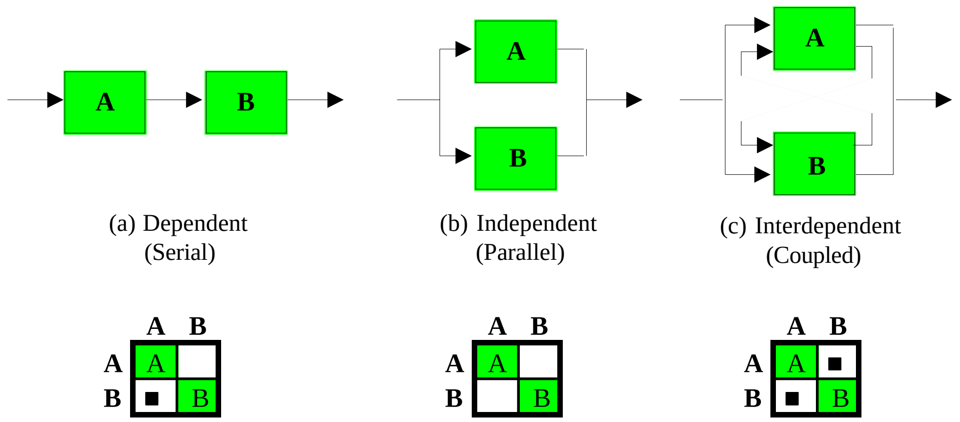

Figure 1: Activity Networks and DSM Equivalents

These efforts expanded into areas such as matrix mathematics [e.g., 22]. However, DSM concepts are also related to network precedence diagrams [e.g., 23, 24] and network relationship diagrams [e.g., 25]. Systems engineers familiar with interface-to-interface (N-to-N or N^2) diagrams [26] will also recognize their similarity to DSMs.

Formal definition and application of these concepts to design process issues came in 1981, when Steward described the “design structure matrix” [27, 28]. In 1989, Rogers at NASA built software to aid in analyzing such matrices [29]. Actual use in industry settings began about 1990, as several professors and students at the Massachusetts Institute of Technology (MIT) built research around the design structure matrix and expanded its applications,[3] as summarized in [39]. In the last few years, this work has expanded beyond MIT to industry and other universities,[4] and the resulting need for a more general name has led to the term *dependency structure matrix*, which retains the initialism “DSM” left over from Steward’s work.

The DSM is a square matrix with one row and column per activity. The activities are listed in roughly chronological order, with upstream or early activities listed in the upper rows. Diagonal elements are placeholders in a simple DSM, and off-diagonal elements indicate activity interfaces. Figure 1 shows the network flowchart and DSM representations of basic activity relationships.[5] Because of the temporal ordering of the activities, subdiagonal matrix elements show feedforward information (such as the flow from activity A to activity B in the serial case). Superdiagonal elements indicate feedback, the potential for iteration and rework in the process. Thus, if activities in rows (and corresponding columns) *i* and *j* of the DSM have no direct interfaces, entries *ij* and *ji* in the matrix will be zero or empty. If, on the other hand, *both* entries *ij* and *ji* are filled, this indicates two-way interdependency or coupling between the activities. Usually this points to some kind of “chicken and egg” problem, of which many exist in complex system design. Traditional PERT/CPM methods and Gantt charts do not adequately represent these types of relationships and their effects.

The following is a simple example, showing a DSM for a familiar process, putting on socks and shoes (Figure 2).[6] Information and precedence relationships flow in a counter-clockwise direction in the matrix, so the marks below the diagonal imply, for instance, that “get socks” must precede “put on socks,” and “get shoes” must precede “put on shoes” and “inspect shoes.” The mark above the diagonal in the DSM indicates that, once shoes have been inspected, they may be found wanting (e.g., too scuffed up or the wrong color for the clothes), requiring an iteration, “get (new) shoes.”

Now, the goal in DSM analysis is to resequence the activities so as to minimize iterations and their scope. Since the activities “get shoes” and “inspect shoes” are coupled, however, there is no way to reorder the rows (and columns) of the DSM to get all the marks below the diagonal. Failing this, we change our goal to

[3] [5, 12, 16, 17, 20, 30-38]
[4] [e.g., 40, 41]
[5] Diagrams adapted from [31, 42].

[6] This five activity DSM example is adapted from a presentation by Stephen Denker, “A New Way to Think About Problems,” Presentation to Project Management Institute (PMI), Boston Chapter, 6/19/97.

getting any superdiagonal marks as close to the diagonal as possible, minimizing the scope of the iteration. In Figure 2, once we "get shoes," we go ahead and "put on socks" and "put on shoes" before we "inspect shoes." If, instead, we moved the inspection step upstream, as in Figure 3, we minimize the *impact* of a need to "get (new) shoes." (Essentially, this act of moving activities upstream demonstrates concurrent engineering: we shorten the feedback loop in the process in hopes that we will decrease the variance in total process lead time.) While in this example we could not eliminate the iteration entirely, often it is possible to reduce the number of potential iterations substantially by resequencing rows and columns.

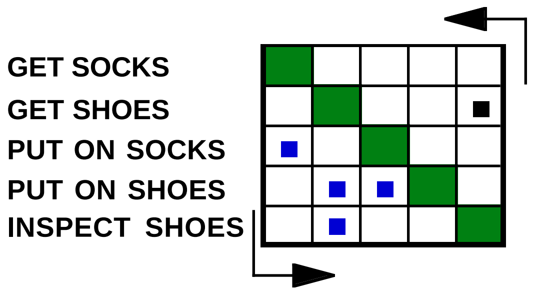

Figure 2: DSM for the Process of Putting on Socks and Shoes

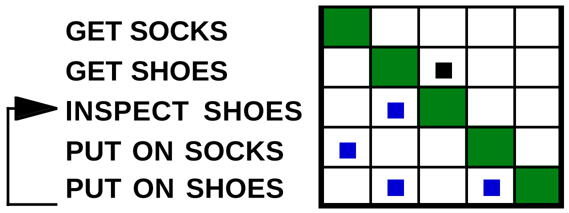

Figure 3: Resequenced DSM Shows Improved Process

The DSM also indicates which activities can be accomplished in parallel without causing additional iteration. For example, in Figure 3, "get socks" and "get shoes" can be done simultaneously, as can "inspect shoes" and "put on socks" (if we have enough resources!). Sometimes planners choose activities to work in parallel without first considering their information dependencies, which can result in additional iteration and thus, more, not less, cycle time.

The DSM introduction in this section is very simple. While time and space do not permit a full discussion and assimilation of all applicable theory and practice, the following points will help guide further inquiry and more sophisticated use of DSM models:

- Off diagonal elements do not have to binary (i.e., present or not): they can be numbers from 0 to 1, from 0 to 9, etc. These numbers can convey any of a number of bits of information regarding the interfaces, such as probability of iteration, percent rework, amount of data flow, type of data flow, sensitivity to change in input to downstream activity, etc. For instance, in the example above, if the probability of having to redo the "get shoes" step after the results of the "inspect shoes" step are in is estimated to be 0.4, this number could replace the superdiagonal square.
- Activities need not be limited to "finish-to-start" type relationships; i.e., activities can be partially overlapped in certain cases, although these decisions should be based on the nature of information production and use by the upstream and downstream activities, respectively [43]. For example, "put on socks" and "put on shoes" could be overlapped, perhaps by putting on one sock, then one shoe, then the other sock, then the other shoe.
- Activity durations may be placed in the diagonal elements of the DSM, and this along with the knowledge of serial, parallel, and coupled activities can lead to rough critical path calculations.
- Activities capable of execution in parallel from an information flow perspective may yet be held up by resource constraints. A more complete analysis must account for these constraints. For example, the DSM above indicates that "get socks" and "get shoes" can occur in parallel. But if we only have one person available for both activities, we may not in fact be able to do both at once.

5 DSM Capabilities and Advantages for Cycle Time Reduction

The DSM has numerous advantages, but perhaps the two greatest ones are:

- Concise representation of complex processes, providing a systems view, and
- Clear rendering of potential iteration in such processes.

Other advantages and capabilities include:

- A description of a process that can be analyzed and modified to provide a prescription for a process with reduced schedule risk and cycle time.
- A means to more accurately manage schedule and anticipate schedule risk.
- A systems view of project activities and their relationships which drive cycle time.

- A model demonstrating appropriately concurrent activities.
- A means to quickly examine potential activity and decision sequence changes ("what if" capabilities) for their effect on schedule.
- A view from which to deploy resources to reduce unintentional iterations.

From the description and examples above, we can see how the method's capabilities help meet the challenges noted earlier:

- Inefficient distribution of personnel and resources, including facilities and tools and especially information, results in parts of the project spending "down" time waiting for resources when they could be working. The DSM shows which activities depend on others for important resources and when these resources will be needed, helping managers insure that the right resources are available at the right place and at the right time.
- Unstable requirements make decisiveness difficult and foster redesign. The DSM can be used to trace the effects of changing information. More sophisticated DSM analysis can even help quantify the effects of such changes on project cost and schedule. This kind of information is valuable to those deciding if, when, and how much to change requirements.
- Long and varied activity "pipelines" mean shorter duration activities finish and wait for longer duration activities to produce needed results. By examining the durations of activities in a DSM, one can quickly isolate these situations and account for them in project plans.
- Lack of activity coordination results in wasted time, doing the wrong work, and inappropriate communication. Using a DSM allows activities to see their place in an overall process, enabling them to see what other activities need their results and when.
- Highly interdependent or coupled activities increase the chances of iteration or rework. The DSM highlights these situations, enabling planners to more appropriately account for them.
- Overly ambitious initial schedules make further cycle time reductions even more difficult. By showing the amount of iteration in a process, the DSM provides a quick check of schedule feasibility and risk. This brings additional, important information to cycle time reduction efforts.
- Many seemingly feasible cycle time reduction solutions have adverse side effects which reduce anticipated effectiveness. The systems view provided by the DSM decreases the chances that process reengineering efforts will miss their mark in reducing cycle time.

In conclusion, the DSM method—which has existed in the literature for 17 years but is only now becoming widely recognized—is shown to provide utility to project managers who seek to reduce schedule risk and cycle time.

Acknowledgments

I would like to acknowledge funding provided by MIT's Lean Aerospace Initiative (LAI) <http://lean.mit.edu/public/>. Additional support has come from MIT and a NSF graduate fellowship. I am grateful for the helpful comments on this material provided by John Deyst, Steven Eppinger, Earll Murman, Ross McNutt, Eric Rebentisch, Kirkor Bozdogan, and others in LAI.

References

[1] K.B. Clark and T. Fujimoto, *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*. Boston: Harvard Business School Press, 1991.

[2] S.C. Wheelwright and K.B. Clark, *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality*. New York: Free Press, 1992.

[3] K.B. Clark and S.C. Wheelwright, *Managing New Product and Process Development*. New York: Free Press, 1993.

[4] E. von Hippel, "Task Partitioning: An Innovation Process Variable" *Research Policy*, vol. 19, pp. 407-418, 1990.

[5] R.P. Smith and S.D. Eppinger, "A Predictive Model of Sequential Iteration in Engineering Design" *Management Science*, vol. 43, pp. 1104-1120, 1997.

[6] S.D. Eppinger, "Three Concurrent Engineering Problems in Product Development." Cambridge, MA, 1995.

[7] K.G. Cooper, "The Rework Cycle: How it Really Works… And Reworks…" *PMNETwork*, 1993.

[8] K.G. Cooper, "The Rework Cycle: Benchmarks for the Project Manager" *Project Management Journal*, vol. 24, 1993.

[9] K.G. Cooper, "The Rework Cycle: Why Projects are Mismanaged" *PMNETwork*, 1993.

[10] D.N. Ford, *The Dynamics of Project Management: An Investigation of the Impacts of Project Process and Coordination on Performance*, Ph.D. Thesis (C.E.), M.I.T., Cambridge, MA, 1995.

[11] S.D. Eppinger, M.V. Nukala, and D.E. Whitney, "Generalised Models of Design Iteration Using

Signal Flow Graphs" *Research in Engineering Design*, vol. 9, pp. 112-123, 1997.

[12] S.D. Eppinger, D.E. Whitney, R.P. Smith, and D.A. Gebala, "A Model-Based Method for Organizing Tasks in Product Development" *Research in Engineering Design*, vol. 6, pp. 1-13, 1994.

[13] D.A. Gebala and S.D. Eppinger, "Modeling the Impact of Organizational Structure on Design Lead Time and Product Quality", MIT Sloan School of Management, Cambridge, MA, Working Paper no. 3301, May, 1991.

[14] D.E. Whitney, "Designing the Design Process" *Research in Engineering Design*, vol. 2, pp. 3-13, 1990.

[15] K.J. Singh, J.W. Erkes, J. Czechowski, J.W. Lewis, and M.G. Issac, "DICE Approach for Reducing Product Development Cycle", *Proceedings of the Worldwide Passenger Car Conference and Exposition*, Dearborn, Mich., Sept. 28-Oct. 1, 1992, pp. 141-150.

[16] D.S. Cesiel, *A Structured Approach to Calibration Development for Automotive Diagnostic Systems*, Master's Thesis (Mgmt./E.E.), Massachusetts Institute of Technology, Cambridge, MA, 1993.

[17] S.M. Osborne, *Product Development Cycle Time Characterization Through Modeling of Process Iteration*, Master's Thesis (Mgmt./Eng.), M.I.T., Cambridge, MA, 1993.

[18] K.M. Eisenhardt and B.N. Tabrizi, "Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry" *Administrative Science Quarterly*, vol. 40, pp. 84-110, 1995.

[19] D. Clausing, *Total Quality Development: A Step-by-Step Guide to World-Class Concurrent Engineering*. New York: ASME Press, 1994.

[20] R.P. Smith and S.D. Eppinger, "Identifying Controlling Features of Engineering Design Iteration" *Management Science*, vol. 43, pp. 276-293, 1997.

[21] D.G. Pekar, *Modeling Design Time and Quality Tradeoffs in Automotive Component Design*, Master's Thesis (Mgmt.), MIT, Cambridge, MA, 1992.

[22] J.N. Warfield, *Societal Systems: Planning, Policy, and Complexity*. New York: John Wiley & Sons, 1976.

[23] E.P.C. Fernando, "Use of Interdependency Matrix for Expediting Implementation of an Integrated Development Programme in a Developing Country", *Proceedings of the Second International Congress for Project Planning by Network Analysis*, Amsterdam, 1969, pp. 76-85.

[24] M. Hayes, "The Role of Activity Precedence Relationships in Node-Oriented Networks", *Proceedings of the Second International Congress for Project Planning by Network Analysis*, Amsterdam, 1969, pp. 128-146.

[25] J.W. Lorsch and P.R. Lawrence, Ed. *Managing Group and Intergroup Relations*. Irwin Series in Management and the Behavioral Sciences. Homewood, Ill.: Richard D. Irwin, 1972.

[26] DSMC, *Systems Engineering Management Guide*. Defense Systems Management College, 1990.

[27] D.V. Steward, *Systems Analysis and Management: Structure, Strategy, and Design*. New York: PBI, 1981.

[28] D.V. Steward, "The Design Structure System: A Method for Managing the Design of Complex Systems" *IEEE Transactions on Engineering Management*, vol. 28, pp. 71-74, 1981.

[29] J.L. Rogers, "Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Processes", NASA, Hampton, VA, Technical Manual TM-110247, April, 1996.

[30] T.A. Black, C.F. Fine, and E.M. Sachs, "A Method for Systems Design Using Precedence Relationships: An Application to Automotive Brake Systems", M.I.T. Sloan School of Management, Cambridge, MA, Working Paper no. 3208,, 1990.

[31] S.D. Eppinger, "Model-based Approaches to Managing Concurrent Engineering" *Journal of Engineering Design*, vol. 2, pp. 283-290, 1991.

[32] D.A. Gebala and S.D. Eppinger, "Methods for Analyzing Design Procedures", *Proceedings of the ASME Third International Conference on Design Theory and Methodology*, 1991, pp. 227-233.

[33] V. Krishnan, S.D. Eppinger, and D.E. Whitney, "Simplifying Iterations in Cross-Functional Design Decision Making" *Journal of Mechanical Design*, vol. 119, pp. 485-493, 1997.

[34] K.R. McCord, *Managing the Integration Problem in Concurrent Engineering*, Master's Thesis (M.E.), Massachusetts Institute of Technology, Cambridge, MA, 1993.

[35] M.D. Morelli, S.D. Eppinger, and R.K. Gulati, "Predicting Technical Communication in Product Development Organizations" *IEEE Transactions on Engineering Management*, vol. 42, pp. 215-222, 1995.

[36] T.U. Pimmler and S.D. Eppinger, "Integration Analysis of Product Decompositions", M.I.T. Sloan School of Management, Cambridge, MA, Working Paper no.3690, May, 1994.

[37] M.W. Sequeira, *Use of the Design Structure Matrix in the Improvement of an Automobile Development Process*, Master's Thesis (MSE/Mgt.), M.I.T., Cambridge, MA, 1991.

[38] R.P. Smith and S.D. Eppinger, "Modeling Design Iteration", M.I.T. Sloan School of Management, Cambridge, MA, Working Paper no.3160,, 1990.

[39] T.R. Browning, "An Introduction to the Use of Design Structure Matrices for Systems Engineering, Project Management, and Organization Planning", M.I.T. Lean Aircraft Initiative, Cambridge, MA, Working Paper #WP97-001-18, Feb. 18, 1997.

[40] S. Austin, A. Baldwin, and A. Newton, "A Data Flow Model to Plan and Manage the Building Design Process" *Journal of Engineering Design*, vol. 7, pp. 3-25, 1996.

[41] D.L. Grose, "Reengineering the Aircraft Design Process", *Proceedings of the Fifth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, Sept. 7-9, 1994

[42] R.P. Smith, *Development and Verification of Engineering Design Iteration Models*, Ph.D. Thesis (Mgt.), M.I.T., Cambridge, MA, 1992.

[43] V. Krishnan, S.D. Eppinger, and D.E. Whitney, "A Model-Based Framework to Overlap Product Development Activities" *Management Science*, vol. 43, pp. 437-451, 1997.

Author Biography

The author is currently a Ph.D. candidate in the Technology, Management, and Policy Program (interdisciplinary systems engineering and technology management) at MIT and a research assistant with the Lean Aerospace Initiative. Mr. Browning's research focus is on strategies and methods for complex system product development—currently on models of project cost, schedule, and performance risk. He holds a S.M. in Aeronautics and Astronautics from MIT, a S.M. in Technology and Policy from MIT, and a B.S. in Engineering Physics from Abilene Christian University. He has previous work experience at Honeywell, Inc. and at Los Alamos National Laboratory.