Using the Design Structure Matrix (DSM) for Process Integration

Tyson R. Browning
Lockheed Martin Aeronautics Company
P.O. Box 748, MZ 2274
Fort Worth, TX 76101 USA
tyson.browning@lmco.com (or) *tyson@alum.mit.edu*

Abstract. The new standards advocate integrated engineering processes. A process is a kind of system. As such, it derives its added value from the relationships among its parts (e.g., activities). For a group of activities to be truly integrated (versus merely aggregated), their interfaces must be well defined. In engineering processes, these interfaces usually indicate a flow of information. Engineering processes are extremely complex because of the large number of interfaces, as many types of information flow to many destinations. This paper reviews a powerful technique, the *design structure matrix* (DSM), for representing and analyzing complex processes. The DSM is extended to account for external inputs and outputs, providing the basis for process "puzzle pieces" that can be assembled to form large, integrated processes.

INTRODUCTION

Emerging standards for engineering, design, and product development processes such as CMMi, EIA/IS 731, ISO 15288, etc. advocate the inclusion of a number of "good practices." Essentially, these practices are activities that should be part of any development process so that it can be capable, mature, repeatable, etc.—with the implication that such processes provide the maximum value to their customers and users. Unfortunately, processes for the development of large, complex systems are already complex, and the inclusion of additional activities does not make them any simpler.

One of the major problems in complex system development projects is the difficulty coordinating the contributions of a number of activities, such that each of these contributions comes at just the right time. In product development, many of the contributions come in the form of information that is consumed, transformed, and supplied by activities. The value of the process is compromised when information is "out of sync," forcing those executing activities to make assumptions in the absence of real data [3, 4]. This problem is exacerbated as more activities and contributions must be managed. No one can keep track of everything. We need better tools that will give us visibility into these situations, highlight problems, suggest solutions, and be able to handle increasing complexity.

A classic means to address and reduce complexity is through modeling. A model is an abstract representation of reality that is built, analyzed, and manipulated to increase understanding of that reality. A good model is helpful for testing hypotheses about the effects of certain actions in the real world, where such actions would be too disruptive or costly to try in the real situation. Here, we are interested in models that will help us represent, understand, manage, and improve complex processes. Such models would also facilitate process integration.

Process modeling, like many other types of system modeling, is often approached through process decomposition into simpler elements.[1] But a complex process is more than just a grouping of activities. It exists for a purpose—to produce something. Especially in product development contexts, that something typically requires the activities to collaborate, not simply to make a unilateral contribution. Process complexity is a function of (1) the number of elements, (2) the individual complexity of each of those elements, (3) the number of relationships between the elements, and (4) the individual complexity of each of those relationships. Rechtin [7] reminds us that relationships among elements are what give systems their added value, and that the greatest leverage in systems architecting is at the interfaces. This is no less true for processes. Hence, a good process model must account for the interfaces between its activities. Unfortunately, what often passes for a process model in industry fails to say much about the relationships between activities,

[1] Of course, decomposition presents the danger of incorrect abstraction—failing to represent the characteristics of the process that provide its full value and capability.

as evinced in flowcharts where the modelers label the boxes but not the arrows.

While many organizations that develop complex systems have made efforts to "document their processes," very few have actually built (and committed to improve and learn from) useful process models. Furthermore, the process models or descriptions that do exist are often unintegrated, and it is left to those attempting to plan, execute, and manage projects to wicker disparate process models into a coherent whole. Unfortunately, what emerges is seldom understandable by or useful to many people, and it often has little impact on actual project planning, execution, and management. On the other hand, an integrated process model would be very valuable to a number of users.

Some people confuse the *real process* (how work is really done) with the *process model* or definition, which is only an abstract representation of the real process. Standard processes, tailored processes, flowcharts, etc. are all just process models. Models can be improved by adding more detail (making them less abstract) and by verifying that they adequately and accurately represent reality. It is usually the case—especially in complex processes like engineering—that the models never fully represent the way work really occurs.[2] Thus, a prerequisite to process improvement (changing the real process) is increasing the adequacy and accuracy of the process model.

How do we know when a process model is sufficiently adequate and accurate? It depends on the intended use of the model. The list in Table 1 covers a variety of potential uses for (and users of) process models. No one user would need all of the listed capabilities. Various types of users require different views of process model data. However, all of the information represented by any view of a process model must come from a consistent source (typically a database). (When data are added or updated, those data should automatically update in the views of all users.) The goal is to provide a single process model that is sufficiently adequate and accurate for all of these uses and users at once. This is possible because it is easy to extract only the relevant portion of a detailed process model and provide it to a user with limited needs. But it is extremely undesirable to maintain a variety of disparate, less-detailed models of the same process. Unfortunately, this is the current situation in many companies: several unintegrated representations of how work gets done are maintained by various users for various uses—e.g., cost and schedule accounting by one group, planned work flow and assignments by others, potential activities to add to the plan by others, the "standard" view of the process by others, etc.

• **Program Planning:** A process model helps determine the statement of work (SOW), the work breakdown structure (WBS), the integrated master plan (IMP) and schedule (IMS), and therefore the systems engineering master/management plan (SEMP). It also helps to estimate cost, schedule, effort, resources, and risk. It is therefore useful for proposal preparation.
• **Program Execution:** A process model helps identify the critical path, determine what to work on this week, evaluate progress, coordinate deliverables, analyze the impacts of changes and the value of options, and replan the remainder of a program.
• **Baseline for Continuous Improvement:** A process model helps analyze potential process changes in terms of net value (investment costs vs. value added benefits) and helps isolate root causes of problems.
• **Knowledge Retention and Learning:** A process model captures lessons learned when the process does not work as expected.
• **Process Visualization:** A process model helps people visualize where they are in a process and what they need and must produce and when.
• **Training:** A process model can help new hires get oriented, see what they need to do and why, and see where to go for more information.
• **Framework for Metrics:** A hierarchical process model serves as the framework for organizing low-level measures and for rolling them up to feed high level metrics that tie directly to business goals.
• **Compliance/Audits/Assessments:** A process model helps an organization comply with audit requirements and assessments.

Table 1: Uses of a Good Process Model

This paper presents an approach to process modeling and integration that provides the foundation for all of these uses. The next section discusses the building blocks of a process model. Then, a process representation and analysis technique called the *design structure matrix* (DSM) is reviewed. The paper then discusses the importance of process synchronization, which must be prefaced by process integration. The DSM is used to demonstrate process integration applications.

PROCESS MODELING OBJECTS AND THEIR ATTRIBUTES

A basic process model can be built from two types of objects, *process elements* and *deliverables*. Process elements represent processes, subprocesses, activities, tasks, or any package of work that produces an output.

[2] Modelers often describe the way things have always been done (which probably needs to be improved) or the way they would like things to be done (which is unverified and may be infeasible).

Most process elements will require some kind of input(s) as well. Both inputs and outputs are deliverables. This paper will mainly refer to process elements as processes or activities, in the relative sense that processes are "parents" and activities are "children." The paper refers to deliverables as either internal or external, where internal deliverables are produced and consumed within the boundary of a process, and external deliverables are either produced or consumed outside the boundary of a process. Some typical process element and deliverable attributes are listed in Table 2. Process elements are the same basic type of object as IPO (input, process, output) and ETVX (entry, task, verify, exit) objects.

These objects will often be defined in a decentralized fashion. For example, a survey can be sent to a person with expertise in a particular activity to capture that activity's attributes. A model integrator can then assimilate multiple survey responses into a process model such as the one in Figure 1. Model integrators will spend most of their time resolving the deliverable objects that link the process elements, since activity experts will seldom agree at first on deliverable names and flow paths. After the initial model is built, with all of its inconsistent deliverables highlighted, all of the activity experts can be brought together to reach consensus. Despite the effort required, a somewhat decentralized approach to model building allows the people doing the work to contribute, yielding a more accurate process description that users will accept.

THE DESIGN STRUCTURE MATRIX (DSM)

Complex processes quickly become challenging to represent, present, and understand. Flowcharts with all kinds of boxes and arrows ("spaghetti and meatballs") do not simplify the problem. Fortunately, a technique for representing complex systems and their relationships in a concise and visual manner can be applied to processes. As shown in Figure 2, a DSM is a square matrix with corresponding rows and columns. The diagonal cells represent the activities, which are listed from upper left to lower right in a roughly temporal order. Off-diagonal cells indicate the dependency of one activity on another—e.g, information flow. Reading down a column shows information sources; reading across a row shows information sinks.[3] For example, Activity 1 provides information to Activities 2, 4, 5, and 6. Activity 2 depends on information from Activities 1 and 6 and provides information to Activities 3 and 4.

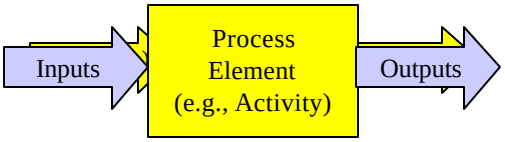
Figure 2 shows how the DSM displays dependent, independent, and interdependent activity relationships. Since Activity 2 depends on information from Activity 1, these two activities will probably be executed sequentially in the workflow. Activities 3 and 4 do not depend on each other for information, so they may safely proceed in parallel (barring other resource constraints). Activities 5 and 6 both depend on each other's outputs. These activities are said to be interdependent or coupled and are discussed below.

Of particular interest are the cases where marks appear in the lower-triangular region of the DSM. Such marks indicate the dependence of an upstream activity on information created downstream. If project planners decide to execute the activities in the given order, Activity 2 will have to make an assumption about the information it needs from Activity 6. After Activity 6 finishes, Activity 2 may have rework if the assumption was incorrect. The DSM conveniently highlights iteration and rework, especially when it stems from activities working with potentially flawed information.

When we see a mark in the lower-left corner of the DSM, we know that there is a chance of having to return to the beginning of the process, which could have a catastrophic impact on cost and schedule. The marks in the lower-left corner of the DSM may represent key drivers of cost and schedule risk. Rearranging the activity sequence (by rearranging the rows and columns in the DSM) can bring some subdiagonal marks above or closer to the diagonal, thereby reducing their impact. Simple algorithms automate this exercise. Adding quantitative information to the DSM and using simulation can quantify the impacts of process architecture changes on cost and schedule risk [5].

Sometimes a subdiagonal mark cannot be brought above the diagonal without pushing another mark below the diagonal. This is a case of interdependent activities, such as Activities 5 and 6. Each activity depends on the other. They must work together to resolve a "chicken and egg" problem. Coupled activities might work concurrently, exchanging preliminary information frequently. If a subset of coupled activities must begin before the rest, the more robust (less volatile and/or sensitive) information items should be the ones appearing below the diagonal in the DSM. If coupled activities are functionally-based, an opportunity may exist to fold the activities into a single activity assigned to a cross-functional team.

[3] Some DSMs use the opposite convention—rows for sources and columns for sinks—resulting in feedback appearing above the diagonal. The two conventions convey equivalent information.



Process Element Attributes	Deliverable Attributes
• Name • Description • Inputs (and sources) • Outputs (and sinks) • Parent process element • Constituent subelements (“children”) • Metrics (duration, cost, risk, etc.) • Resources (critical skills, roles, tools, facilities, etc.) • Organizational assignment • Entrance criteria • Exit criteria • (And others in advanced model)	• Name • Description • Supplier(s) • Consumer(s) • Parent deliverable • Constituent deliverables (“children”) • Format • Medium • Requirements • Verification and Validation criteria • (And others in advanced model)

Table 2: Fundamental Building Blocks of Process Models

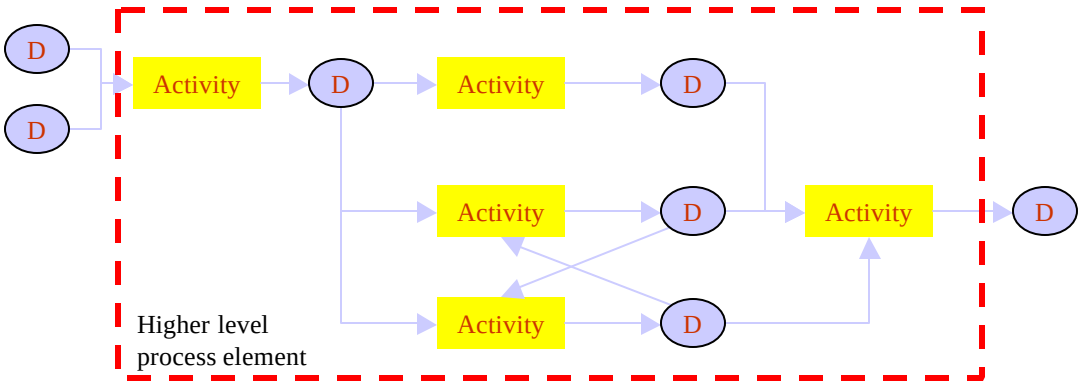


Figure 1: Linking of Process Model Objects

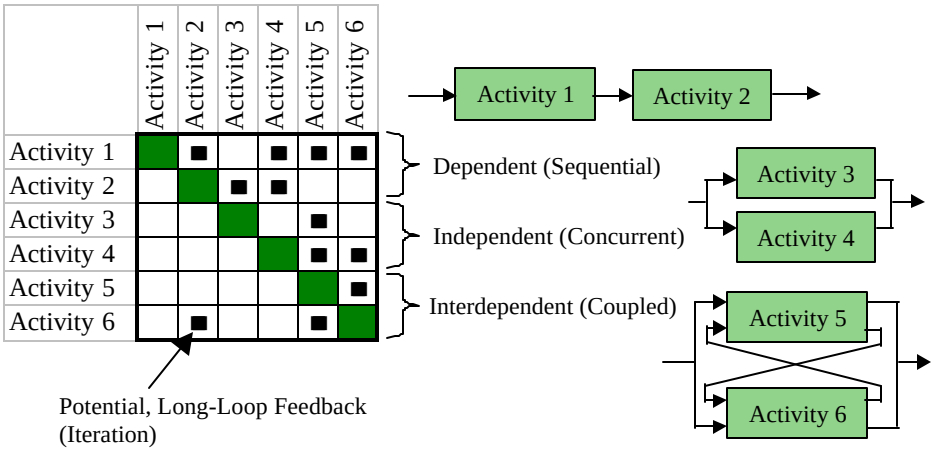


Figure 2: Example DSM

Integration, test, and design review activities typically have marks in their rows to the left of the diagonal. These activities create information (including results of decisions) that may cause changes to (and rework for) previously executed activities. Unfortunately, most process planners "plan to succeed" and their process models fail to account for these possibilities. Fortunately, the DSM provides an easy way to document potential "process failure modes" and their effects on other activities. The simple marks in the DSM can be replaced by numbers indicating the relative probability of information change, iteration, etc. This enables an analysis of process failure modes and their effects on cost, schedule, and risk. Process improvement investments can then target mitigation of the biggest risk drivers. The process model thus becomes the repository for organizational learning (lessons learned), and future projects that follow the process receive the benefit of best practices.

For a real-life example, Figure 3 displays a DSM of the Preliminary Design process for an *uninhabited combat aerial vehicle* (UCAV).[4] The first dozen activities comprise the Conceptual Design phase. In this phase, design requirements and objectives (DR&O) are prepared, a configuration concept is proposed, it is analyzed by a variety of discipline perspectives, and then these results are assessed. The assessment may reveal a need to alter the DR&O, to create a new configuration concept, and/or to alter the current configuration concept. This cycle repeats until the design space is sufficiently understood and/or time and money are exhausted. (All of these activities are condensed into a single row and column in Figure 3.) The design process then moves into the Preliminary Design phase, where the configuration is developed and analyzed in more detail and the objective is to prepare a proposal to acquire funding for additional phases. Figure 3 shows the process "as is," without any attempt to resequence the process to eliminate feedback. This basic model served as the basis for additional process analysis, evaluation, discussion, and improvement.

Figure 3 contains an additional extension: the regions above and to the right of the main matrix account for external inputs and outputs, respectively. Since we can look down a column of the DSM to see where an activity receives its information from, we simply continue looking above, along the extended column, to see external inputs. Similarly, we read across the extended row to see external outputs. The first row in the external inputs region is a summary row, as is the first column in the external outputs region.

[4] The UCAV example comes from The Boeing Company and is fully documented in [1].

The DSM provides a concise, visual format for representing processes. A process flowchart consuming an entire conference room wall can be reduced to a single-page DSM. After a quick orientation, everyone can see how his or her activity affects a large process. People can see where information comes from and where it goes. They can see why delaying the activities they depend on forces them to make assumptions, which may trigger rework later. It becomes apparent that certain information changes tend to cause rework. Such situation visibility and awareness leads to improved process design and coordination. The DSM also provides a process knowledge base from which the foundations of process plans and risk assessments can be drawn. Moreover, the DSM is amenable to some simple yet powerful analyses.

As a metaphor, consider the pictures made when fans in the stands at a football game each hold up a colored card. None of those doing the work may be able to see the big picture. The DSM provides the view from the blimp.

DSMs have been developed for planning and managing projects in the building construction, photographic, semiconductor, automotive, aerospace, telecom, and electronics industries [2]. More detailed and quantitative models based on the DSM have been developed by several researchers [e.g., 5, 6, 8-10].

Perhaps the greatest barrier to DSM usage is the amount of information required to characterize the structure of a design process. DSMs representing complex system development processes call for integrating the expertise of a number of people. Building a DSM also forces some people and groups to think in terms they may not be accustomed to. But this is good, and it should be made to happen anyway. A great amount of benefit is often realized simply by participating in the DSM construction process.

THE IMPORTANCE OF PROCESS SYNCHRONIZATION

The DSM provides important benefits for process representation and understanding, which are prerequisites for process improvement. But process improvement requires looking outside as well as inside the process's boundary. The external regions of the DSM help with this challenge. This section discusses why external inputs and outputs are essential to account for in a process model.

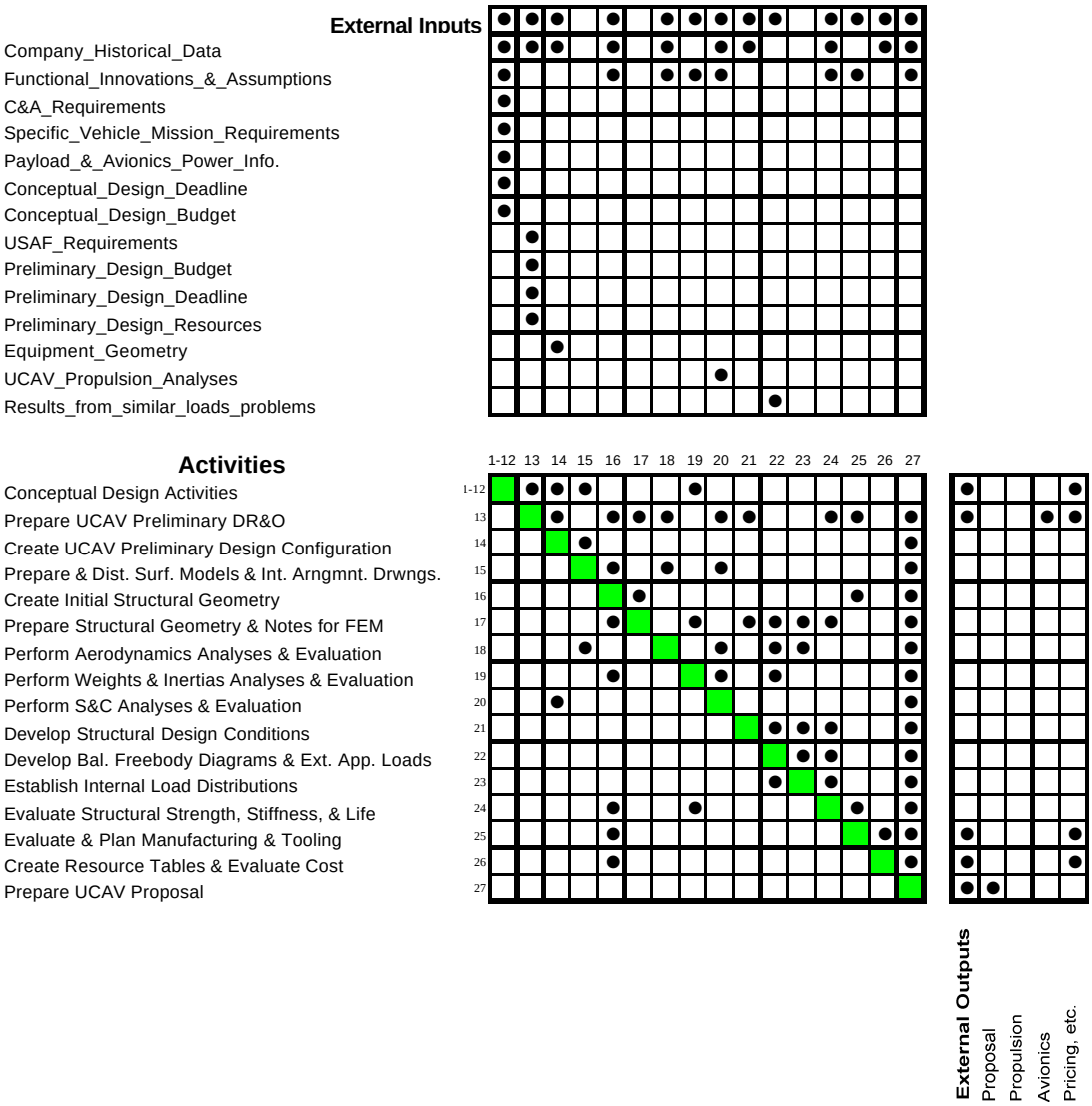


Figure 3: Example DSM for UCAV Preliminary Design Process

Process efficiency depends not only on the efficiencies of the process's constituent elements but also on the timing of interactions with external processes—i.e., process synchronization. To illustrate, consider the following example, beginning in Figure 4. A process contains three activities, executed in sequence. Each activity depends on input from both its predecessor activity (except for Activity 1) and an external process. The result of the process, provided by Activity 3, is a final deliverable (D). Figure 5 shows this process on a time line.

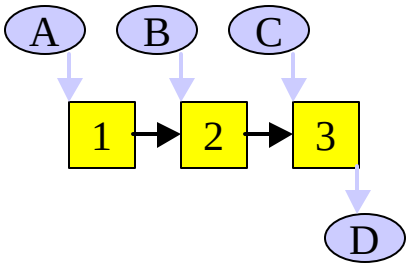


Figure 4: Example Process (3 Activities, 3 Inputs, 1 Output)

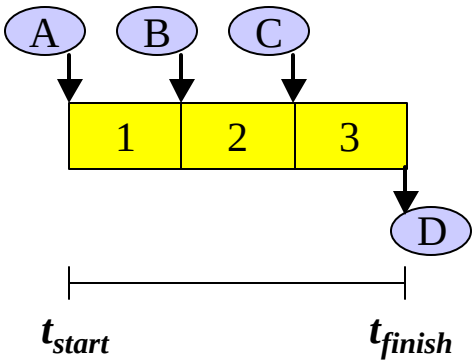


Figure 5: Example Process Time Line

Now, suppose the process is the beneficiary of a "lean initiative" that cuts each activity's duration in half, as shown in Figure 6. Unfortunately, the external inputs still arrive at their same, old times. If the activities have to wait on their external inputs, then the resulting time savings for the whole process is only the savings from the last activity—much less than the process's managers probably expect.

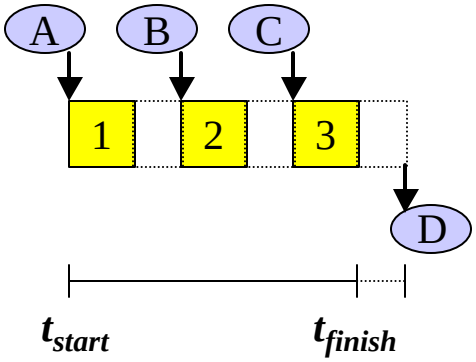


Figure 6: Process Improvement Cuts Cycle Time in Half for Each Activity

Instead of waiting for the external inputs to arrive, suppose activities 2 and 3 each go ahead and begin work as soon as their predecessor finishes, as shown in Figure 7. For example, Activity 2 begins when Activity 1 provides outputs, without waiting for external input B. In this case, Activity 2 is making an assumption about B. If that assumption turns out to be incorrect, then Activity 2 will have to do rework (Activity 2r). If activities make many assumptions and begin earlier than they should, the resulting rework can offset any supposed time savings for the process.[5] In fact, the time savings may not be any more than if the activities had just waited for their inputs, as in Figure 6. And the costs may be higher, because waiting resources can do other tasks (and get paid by other budgets), but reworking resources are getting paid for doing the same task twice. Unfortunately, it is all too common in many development programs for activities to begin before they should, because the people assigned to those activities do not want to be seen "sitting around." (What this really says is that the organization's resource allocation mechanisms and policies are not flexible enough.)

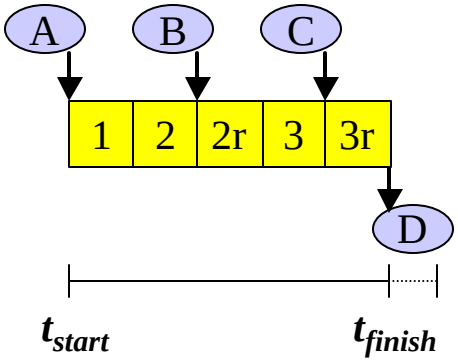


Figure 7: Activities Beginning Without Their Inputs Often Have Rework

By now, hopefully it is obvious that actually realizing the savings expected in the process requires coordinating the availability of the external inputs—i.e., process synchronization—as in Figure 8. It is not enough for each individual process to be lean and efficient: all processes (i.e., the entire product development process, including the supply chain) must be synchronized and coordinated, or else there are few real time and cost savings that will actually get passed along to the bottom line. The essence of lean product development is to get "the right thing in the right place at the right time."

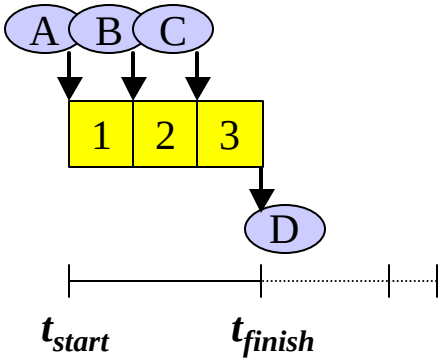


Figure 8: Synchronizing the Arrival of External Inputs Allows Realization of More Savings

[5] Rework is an even worse problem in activity webs or networks than it is in activity chains, because rework in one activity cascades throughout all activities that depend on the reworking activity's output(s).

PROCESS INTEGRATION

Before we can coordinate interfaces, we must be aware of and understand all the interfaces: process integration is a prerequisite to process synchronization. Some organizations begin their process modeling efforts by documenting their existing work methods in a decentralized fashion. That is, they tell various managers in the organization to document their own processes. The collection of processes that results is often stored in a process asset library of some kind. The collection may be dubbed the "standard process set" or something similar. However, in many cases this collection is merely an *aggregation*, not an *integration*, of the constituent processes. That is, the relationships between the processes may not be well defined (correct, sufficient, etc.). Process integration is the work required to wicker the individual processes together into an integrated process. This section demonstrates this exercise using the DSM.

For example, consider a set of three processes, each defined by a DSM: Process A (Figure 9), Process B (Figure 10), and Process C (Figure 11). Each process comes from a separate organization and has a separate owner. Each DSM is augmented with regions above and to the right to show external inputs and outputs, respectively. External interfaces exist with each of the other two processes and with processes external to this set of three.

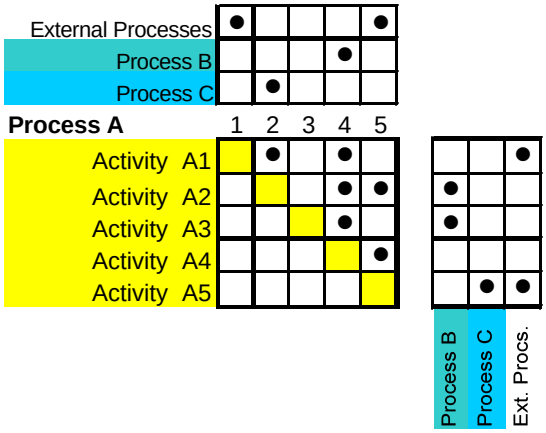


Figure 9: Example DSM for Process A

At a high level, the way these processes relate to each other is shown by the DSM in Figure 12. Note that each of the three processes provides something to and receives something from the other two. How should these processes be sequenced? Should they all just proceed simultaneously? There is no way to resequence the DSM that will eliminate subdiagonal marks. Do the subdiagonal marks imply that this set of processes doomed to experience lots of rework?

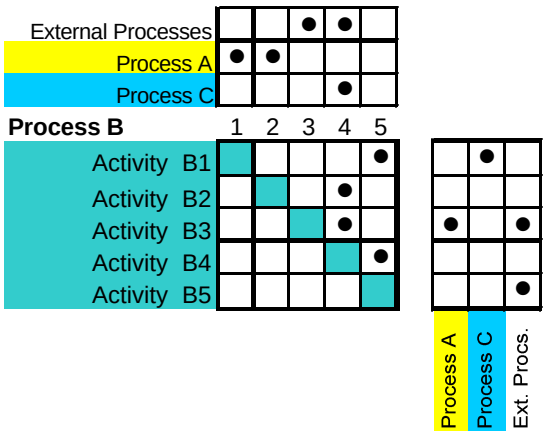


Figure 10: Example DSM for Process B

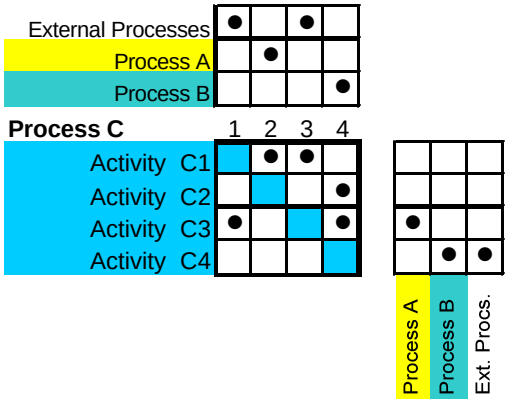


Figure 11: Example DSM for Process C

The first thing we should try to do is understand this top-level, aggregate process better. This requires more information about each lower-level process and its interfaces: What are the particular activities that are requiring and producing the deliverables? What are the particular deliverables that are being exchanged? By taking the individual processes' DSMs and aggregating them, we arrive at a DSM like the one in Figure 13. Each diagonal element of the DSM in Figure 12 has been replaced by a lower-level matrix; each off-diagonal element has also been "zoomed in." Using the external regions around the DSM from each process, we can determine which specific activities are exchanging deliverables.

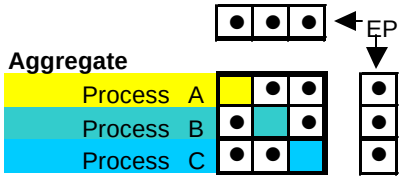


Figure 12: Aggregate DSM for Integrated Process

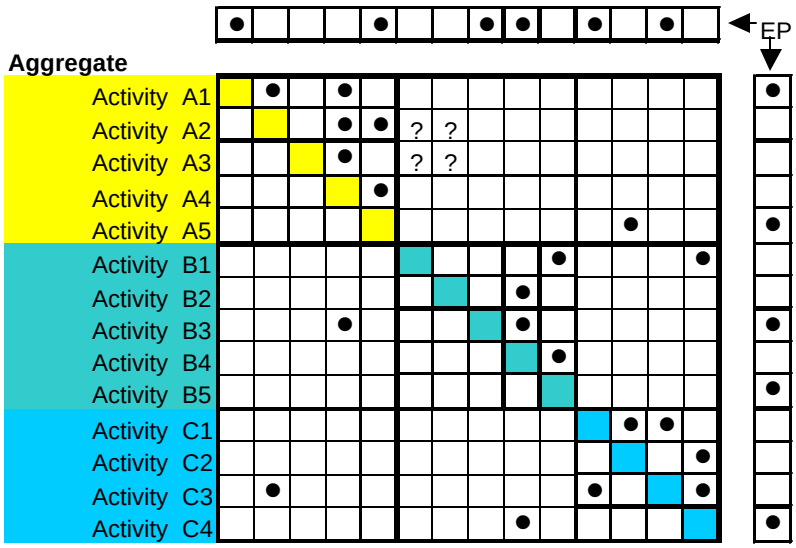
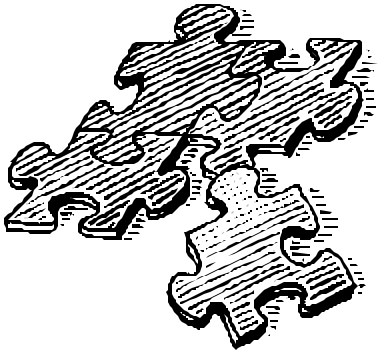


Figure 13: Expanding the Aggregate DSM by Decomposing Activities and Deliverables

In cases where processes are sending or receiving a single deliverable, it is easy to determine which activities are involved. For example, Process A provides a deliverable to Process C, and we can see from Figure 9 and Figure 11 that, actually, Activity A5 is providing the deliverable to Activity C2. In other cases, where more than one deliverable is being exchanged, actual knowledge of the particular deliverables is needed to complete the integrated DSM. For instance, Process A provides two deliverables to Process B. From Figure 9, we know that the deliverables are provided by activities A2 and A3; from Figure 10, we know that the deliverables are consumed by activities B1 and B2. But just looking at the individual DSMs will not tell us which goes where. (Of course, it is possible that both deliverables go to both places.) Knowledge of the deliverable objects themselves is necessary to determine the integrated DSM (Figure 14). Adding detail to a process model requires an amount of digging and puzzling.

Once an integrated DSM has been built, we can analyze the set of processes in more detail and gain new insights. For example, by block diagonalizing[6] the lower-level DSM, the integrated set of processes can be sequenced to minimize feedbacks—e.g., to minimize the need to make assumptions—as shown in Figure 15. By intermingling the activities from the three processes in the DSM, a reasonable sequence for executing the activities is revealed.



As the example shows, process integration requires an understanding of the deliverables that must be exchanged between processes and their constituent activities. If these interfaces are to be coordinated and the processes synchronized, they must first be understood and resolved so that both parties agree about the deliverable's content, timeliness, format, medium, etc. Understanding processes requires digging deeper into their activities and deliverables. How deep is enough? We should understand and document processes to the level we wish to effectively manage, control, coordinate, and synchronize processes.

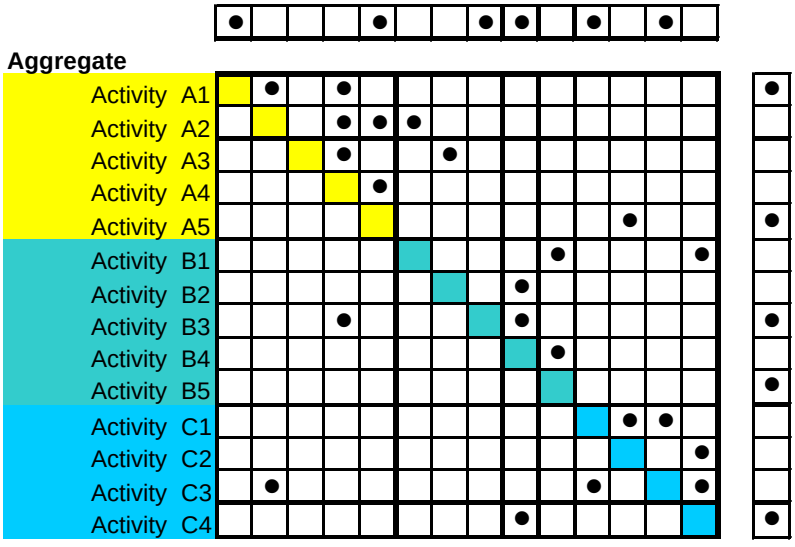


Figure 14: Complete Aggregate DSM

Once the DSM has been manipulated to achieve an acceptable activity sequence, this sequencing can be input to scheduling tools such as Microsoft Project® for further analysis, resource loading, etc. The DSM is

[6] Block diagonalization, block triangularization, partitioning, sequencing: see [2]

useful as the basis for structuring workflow in product data management (PDM) tools; DSM output can be the basis for a truly integrated master schedule (IMS).

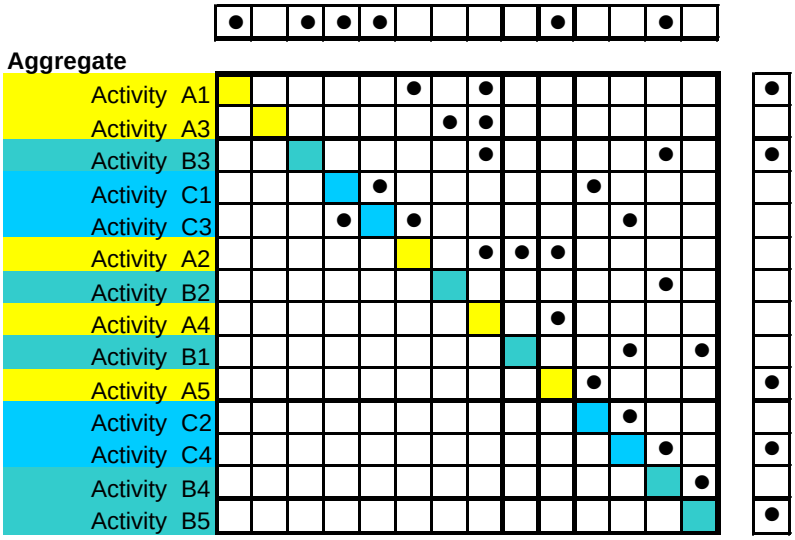


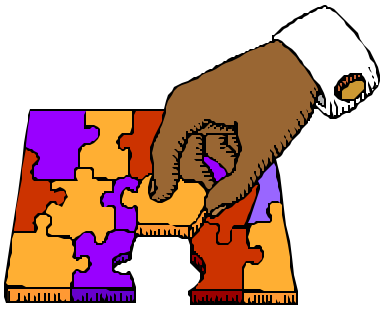
Figure 15: Aggregated DSM After Sequencing

CONCLUSION

This paper shows how process modeling can serve an important role in process understanding and improvement. Processes are complex systems. A generic tool for managing complexity, the DSM, can help. Large, detailed DSMs can be built by integrating smaller DSMs. Smaller DSMs, including regions to account for external inputs and outputs, are the "puzzle pieces" from which integrated processes are built. Process integration is especially important when there are large numbers of interdependent activities to coordinate. Process integration and synchronization lead to reduced variance in process execution time and cost, which can translate directly into competitive advantage for a perceptive organization.

AUTHOR BIOGRAPHY

Tyson R. Browning provides internal consulting and conducts applied research on engineering process development for Lockheed Martin Aeronautics Company in Fort Worth, Texas, USA. He previously worked with the Product Development Focus Team of the Lean Aerospace Initiative at MIT. He earned a Ph.D. in Technology, Management and Policy (interdisciplinary technology management and systems engineering) from MIT.



REFERENCES

[1] T. R. Browning, *Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development*, Ph.D. Thesis (TMP), Massachusetts Institute of Technology, Cambridge, MA, 1998.

[2] T. R. Browning, "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," Lockheed Martin Aeronautics Company, Fort Worth, TX, Working Paper, July 2000.

[3] T. R. Browning, "Value-Based Product Development: Refocusing Lean," presented at IEEE EMS International Engineering Management Conference (IEMC), Albuquerque, NM, 2000, pp. 168-172.

[4] T. R. Browning, J. J. Deyst, S. D. Eppinger, and D. E. Whitney, "Complex System Product Development: Adding Value by Creating Information and Reducing Risk," presented at Tenth Annual International Symposium of INCOSE, Minneapolis, 2000, pp. 581-589.

[5] T. R. Browning and S. D. Eppinger, "Modeling the Impact of Process Architecture on Cost and Schedule Risk in Product Development," Lockheed Martin Aeronautics Company, Fort Worth, TX, Working Paper, Apr. 2000.

[6] 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.

[7] E. Rechtin, *Systems Architecting: Creating & Building Complex Systems*. Englewood Cliffs, NJ: PTR Prentice Hall, 1991.

[8] J. L. Rogers, "Reducing Design Cycle Time and Cost Through Process Resequencing," presented at International Conference on Engineering Design, Tampere, Finland, 1997

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

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