Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions

Tyson R. Browning

Abstract—Systems engineering of products, processes, and organizations requires tools and techniques for system decomposition and integration. A design structure matrix (DSM) provides a simple, compact, and visual representation of a complex system that supports innovative solutions to decomposition and integration problems. The advantages of DSMs *vis-à-vis* alternative system representation and analysis techniques have led to their increasing use in a variety of contexts, including product development, project planning, project management, systems engineering, and organization design. This paper reviews two types of DSMs, static and time-based DSMs, and four DSM applications: 1) *Component-Based* or *Architecture DSM*, useful for modeling system component relationships and facilitating appropriate architectural decomposition strategies; 2) *Team-Based* or *Organization DSM*, beneficial for designing integrated organization structures that account for team interactions; 3) *Activity-Based* or *Schedule DSM*, advantageous for modeling the information flow among process activities; and 4) *Parameter-Based* (or low-level schedule) *DSM*, effective for integrating low-level design processes based on physical design parameter relationships. A discussion of each application is accompanied by an industrial example. The review leads to conclusions regarding the benefits of DSMs in practice and barriers to their use. The paper also discusses research directions and new DSM applications, both of which may be approached with a perspective on the four types of DSMs and their relationships.

Index Terms—Design structure matrix, information flow, integration analysis, modularity, organization design, product architecture, product development, project management, project planning, scheduling, systems engineering.

I. INTRODUCTION

PRODUCTS, processes, and organizations are each a kind of complex system. The classic approach to increasing understanding about a complex system is to model it, typically by

1) decomposing it into subsystems about which we know relatively more;
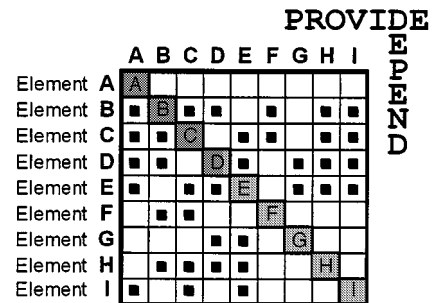2) noting the relationships between (the integration of) the subsystems that give rise to the system's behavior;

Fig. 1. Example DSM.

3) noting the external inputs and outputs and their impact on the system.[1]

With a reasonable model, it becomes possible to explore innovative approaches to system decomposition and integration.

The design structure matrix (DSM) is becoming a popular representation and analysis tool for system modeling, especially for purposes of decomposition and integration. A DSM displays the relationships between components of a system in a compact, visual, and analytically advantageous format. A DSM is a square matrix with identical row and column labels. In the example DSM in Fig. 1, elements are represented by the shaded elements along the diagonal. An off-diagonal mark signifies the dependency of one element on another. Reading across a row reveals what other elements the element in that row provides to; scanning down a column reveals what other elements the element in that column depends on. That is, reading down a column reveals input sources, while reading across a row indicates output sinks. Thus, in Fig. 1, element B provides something to elements A, C, D, F, H, and I, and it depends on something from elements C, D, F, and H.

There are two main categories of DSMs: static and time-based. Static DSMs represent system elements existing simultaneously, such as components of a product architecture or groups in an organization. Static DSMs are usually analyzed with clustering algorithms. In time-based DSMs, the ordering of the rows and columns indicates a flow through time: upstream activities in a process precede downstream activities, and terms like "feedforward" and "feedback" become meaningful when referring to interfaces. Time-based DSMs are typically analyzed using sequencing algorithms.

[1]Building a system model, thus, involves choosing an arbitrary boundary for the system of interest (i.e., determining what is endogenous and exogenous to the system).
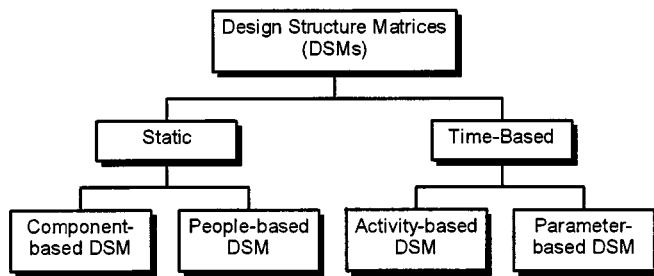
Fig. 2.   DSM taxonomy (adapted from [20]).

DSMs stem from diverse roots. A static DSM is essentially the square matrix called an N^2 diagram, long used by systems engineers to represent architectural components and interfaces (e.g., [12], [40], [63]).[2] The "roof" of a quality function deployment (QFD) matrix [32] exhibits similar interactions. Organization designers also use matrix-based techniques to document communication networks (e.g., [67, p. 107]). Economists summarize the effects of a change in one product's attributes on other products (elasticities) in a matrix (e.g., [34]). Steward [107], [108] used matrix-based techniques to analyze the structure of the system design process, coining the term "design structure matrix" for a time-based matrix akin to a precedence diagram, which had been used to manage projects since the 1960s (e.g., [46], [51]). This paper refers to all of these techniques broadly as design structure matrices (DSMs), although the terms dependency structure matrix, dependency source matrix, dependency map, interaction matrix, incidence matrix, precedence matrix, and others are also used in the literature. The point of the matrix is to illuminate the structure and aid in the design of products, processes, and organizations.

The use of DSMs in both research and industrial practice increased greatly in the 1990s. DSMs have been applied in the building construction [8]–[10], [53], [54], [56], semiconductor [43], [81], automotive [71], [96], [100], [102], [103], photographic [112], aerospace [1], [2], [7], [15], [18], [31], [33], [48], [68], [80], telecom [83], small-scale manufacturing [65], factory equipment [50], and electronics [27] industries.

This paper reviews four DSM applications useful to product developers, project planners, project managers, system engineers, and organizational designers [20].

1) **Component-Based or Architecture DSM:** Used for modeling system architectures based on components and/or subsystems and their relationships.
2) **Team-Based or Organization DSM:** Used for modeling organization structures based on people and/or groups and their interactions.
3) **Activity-Based or Schedule DSM:** Used for modeling processes and activity networks based on activities and their information flow and other dependencies.
4) **Parameter-Based (or Low-Level Schedule) DSM:** Used for modeling low-level relationships between design decisions and parameters, systems of equations, subroutine parameter exchanges, etc.

Fig. 2 shows each application classified as either static or time-based. Each of the four applications is applied to a system

[2]Lano [63] extended N^2 charts to include time-sequenced applications.

decomposition and/or integration problem in the sections that follow. The paper concludes with a summary and a discussion of barriers to DSM use. Relationships between the four types of DSMs are also explored, leading to interesting issues for future research and new DSM applications.

II. MODELING SYSTEM ARCHITECTURE WITH COMPONENT-BASED DSMs

A. Motivation

Product architecture is the arrangement of functional elements into physical chunks that become the building blocks for a product or family of products [112]. Chunks should implement one or a few functions entirely, and interactions between chunks should be well defined. Modular system architectures have advantages in simplicity and reusability for a product family or platform [11], [98]. Research has shown that innovative product architectures can be a source of competitive advantage for product development firms [52]. Where should one look to achieve innovative product architectures? Rechtin reminds us that the relationships among elements are what give systems their added value, and, furthermore, that the greatest leverage in systems architecting is at the interfaces [87]. A prerequisite to innovation is understanding, which can be increased through the use of representative models—in this case, preferably ones that highlight the interfaces or interactions between system elements.

A DSM can represent a system architecture in terms of the relationships between its constituent components. Such a model informs system decomposition into subsystems. Intelligent decomposition or partitioning is important to managing system complexity [3]. The architectural decomposition scheme has ramifications for the ease of system design and integration [63], [87]. The importance of informed architectural decomposition has led to several matrix-based models (e.g., [4], [5], [60], [74]). In general, the system engineering exercise involves the following three steps:

1) decompose the system into elements;
2) understand and document the interactions between the elements (i.e., their integration);
3) analyze potential reintegration of the elements via clustering (integration analysis) [82].

Every complex system development project includes these steps, although they are not always approached systematically or innovatively. A component-based DSM facilitates both systemization and innovation.

B. Method

A component-based DSM documents interactions among elements in a system architecture. An organized taxonomy can help differentiate types of interactions. Pimmler and Eppinger suggest four types, as shown in Table I. The important types of interactions will vary from product to product, and others—such as vibrational or electrical—could also be included. A single (three-dimensional) DSM can represent multiple types of interaction data if each off-diagonal cell contains a vector.

A quantification scheme facilitates weighting interactions relative to each other. Off-diagonal square marks in the DSM are

TABLE I
SIMPLE TAXONOMY OF SYSTEM ELEMENT INTERACTIONS [82]

Spatial	associations of physical space and alignment; needs for adjacency or orientation between two elements
Energy	needs for energy transfer/exchange between two elements (e.g., power supply)
Information	needs for data or signal exchange between two elements
Material	needs for material exchange between two elements

TABLE II
EXAMPLE OF A SPATIAL INTERACTION QUANTIFICATION SCHEME [82]

Required	+2	Physical adjacency is necessary for functionality.
Desired	+1	Physical adjacency is beneficial, but not necessary for functionality.
Indifferent	0	Physical adjacency does not affect functionality.
Undesired	-1	Physical adjacency causes negative effects but does not prevent functionality.
Detrimental	-2	Physical adjacency must be prevented to achieve functionality.

replaced by a number (coupling coefficient)—e.g., an integer $-2, -1, 0, 1,$ or 2 (Table II). Alternatively, the weighting scheme could be exponential instead of linear. Weighting information can be obtained by reviewing architectural diagrams and system schematics. Further clarification comes from interviewing engineers and architectural domain experts.

Integration analysis—via the clustering of off-diagonal elements by reordering the rows and columns of the DSM—can provide new insights into system decomposition and integration. Clustering requires several considerations. The foremost objective is to maximize interactions between elements within clusters (chunks) while minimizing interactions between clusters [11], [87], [98]. It has also been suggested to minimize the size of the clusters [4]. Second, it may be useful to allow for some overlapping of clusters—i.e., recognizing certain elements in more than one cluster. Third, if using a three-dimensional (3-D) DSM, one must decide whether to slice it into several two-dimensional (2-D) matrices and work with each separately, or to perform a composite analysis by weighting the various types of interactions based on their relative importance. (For example, spatial relationships may be more important than data flow associations, since wiring can often be repositioned more easily than larger hardware.) Both procedures have advantages and can reveal significant relationships. However, analyzing 2-D matrices is much simpler, and a composite analysis, while conceptually attractive, might obscure some of the basic insights. Finally, it may be useful to keep integrative elements such as data buses outside of the clusters, noting that these elements must interact substantially with all of the clusters. In some cases, highly interactive components are assigned to a "controls cluster" that interacts with all clusters.

While it is not yet possible to optimize all of these objectives, clustering algorithms are very helpful in integration analysis. By reordering rows and columns, a clustering algorithm seeks a DSM configuration that optimizes an objective function.

For example, the objective could be to minimize the coupling between the clusters while minimizing the size of the largest cluster. In this case, the reordered DSM will have clusters of elements along the diagonal. Several algorithms and heuristics have been offered to aid in determining appropriate objective functions and optimization (e.g., [45], [59], [74]). Altus *et al.* [4], [5] use a genetic algorithm. Pimmler and Eppinger [82] use a distance (from the diagonal) penalty computed for each interaction. Yager [120] discusses advanced clustering algorithms for general applications. A clustering algorithm should account for the importance of both precluding negative relationships and ensuring positive ones. After clustering analysis, any interactions exogenous to the clusters should be noticed as interfaces where special attention and verification may be required. No single clustering approach is a panacea. Fortunately, visual inspection and manipulation are often adequate for small or sparse matrices.

C. Example

Pimmler and Eppinger [82] use a component-based DSM to reveal and explore alternative architectures "to improve the quality of the resulting product design and to ease the substantial coordination demands that are required when subsystems interact" at Ford Motor Company. Fig. 3 shows the materials interaction perspective for an automobile climate control system, where materials exchange is the most crucial architectural design driver out of those listed in Table I. Numerical entries correspond to a quantification scheme like the one in Table II. Not every element of the climate control system interacts with every other element on a materials basis, but all of the materials interactions that do exist are essential to achieve desired functionality.

Using a distance penalty algorithm or by examination, the climate controls system can be clustered into subsystems on the
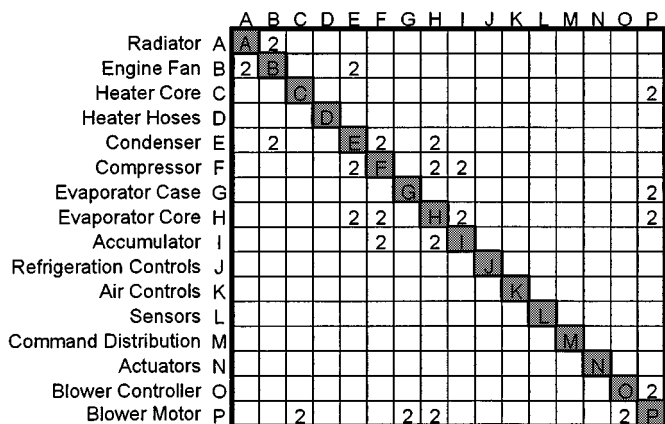
Fig. 3. Component-based DSM showing materials interactions for climate control system (adapted from [82]).
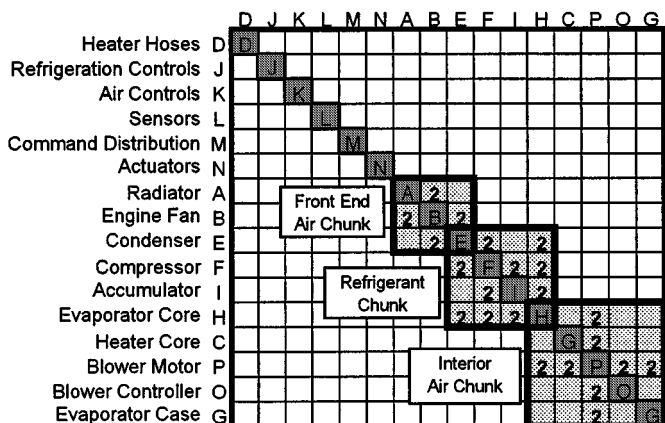

Fig. 4. Clustering of materials interactions in component-based DSM (adapted from [82]).

basis of materials interactions as shown in Fig. 4. The front-end cluster represents the set of components at the fore of the engine compartment involved with heat transfer to the exterior air. The refrigerant cluster consists of the air conditioner components; the interior air cluster represents the components at the front of the passenger compartment involved in modifying interior air temperature. Assigning two of the elements, evaporator core and condenser, to two clusters each forces the clusters to overlap, highlighting areas requiring integration across clusters. Remaining components are assigned to the three existing clusters and a new controls cluster based on spatial, energy, and information interactions (which are not covered here since the materials perspective suffices to illustrate the application). Even this simple analysis revealed to Ford the utility of overlapping what were previously mutually exclusive architectural clusters.

D. Insights

While the above example is simple, the underlying methodology is powerful. When other types of interactions are included, conducting the decomposition and integration analysis with respect to varied objective functions provides alternative architectural perspectives. Integration analysis with a DSM promotes architectural innovation by demonstrating the

rationale behind architecting decisions ([63, pp. 45–48], [118]). Integration analysis also supports modularization [11], [69], [70], [96], which, in turn, enables product platforms and other advantageous approaches to product development. Integration analysis applications are ubiquitous, since clustering facilitates the partitioning of any set of more- or less-related elements into rational groupings.[3]

III. MODELING ORGANIZATIONAL INTEGRATION WITH A TEAM-BASED DSM

A. Motivation

Organizations are extremely complex systems. Better understanding of organizations enables innovation and improvement in organization design. Complex system development requires the exchange of information among various groups or teams. Relationships among people and teams are what give organizations their added value. Building and analyzing a team-based DSM highlights interteam interfaces, which provide the greatest leverage for improving the organization. Better understanding of organizational interfaces supports the application of appropriate integrative mechanisms [17], [21].

B. Method

Modeling an organization as a system requires three steps:
1) decompose the organization into elements (e.g., teams) with specific functions, roles, or assignments;
2) document the interactions between (the integration of) the teams;
3) analyze the clustering of the teams into "metateams."

In conducting the first step, it is often helpful to map the teams to product components such as subsystems (such that the organizational architecture mirrors the product architecture).

Step two usually requires information from the organization's members. Members of each team are asked to note which other teams their team provides information to and receives information from. In addition, one can query the frequency of these interactions. This information is used to fill in the rows and columns of the DSM. Often, modelers will build two DSMs—one for the information "supplier" perspective and one for the "consumer" perspective. Modelers must then follow up to iron out discrepancies between the two DSMs in an effort to converge on a consensus DSM. Unless organizational interfaces have been carefully defined, some teams may see their interaction with another team as merely one of providing information, whereas, the other team may see the relationship as bilateral information exchange. In other cases, one team may note an interface with another team, and this other team may not even recognize that the interface exists. Note that the DSM is static; i.e., it shows interactions between teams existing at a given point in time.[4]

In addition to information dependencies, Thomas and Worren [111] present the possibility of accounting for dependencies of responsibility, accountability, consultation, and commitment.

---

[3]Other applications include, e.g., portfolio segmentation as applied to option packages, product lines, business units, etc.

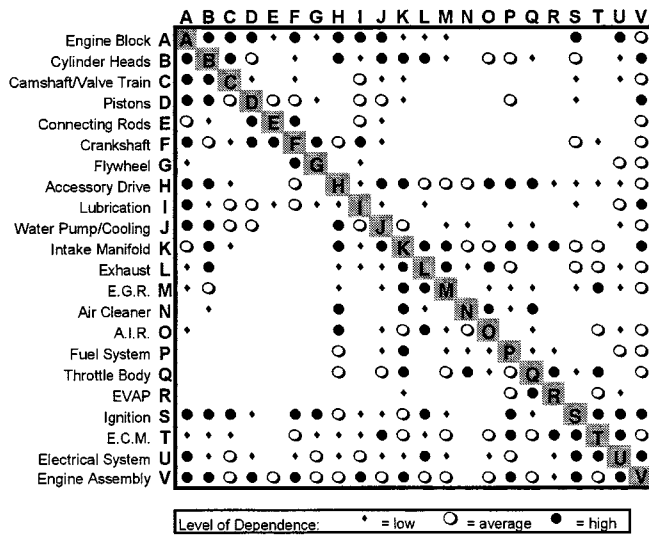[4]Showing dynamic teams and relationships requires a 3-D DSM.

Fig. 5. Team-based DSM showing information flow between PDTs (adapted from [72]).

Fig. 6. Restructured, team-based DSM shows proposed "metateam" organization design (adapted from [72]).

Analysis of these various dimensions of interaction could require multiple DSM planes, akin to the approach presented in the last section (stemming from Table I).

Step three follows the same general integration analysis methodology as presented in Section II-B. The DSM is analyzed with the primary goal of clustering teams into metateams where interactions are most essential and/or frequent. Interactions between clusters or metateams are to be minimized. In addition, perhaps some teams should be aggregated into a single team, or perhaps some teams should be divided. Highly interactive teams might be left outside the main clusters and assigned an integrative role.

C. Example Application

McCord and Eppinger [43], [72] use a team-based DSM to analyze an automobile engine development organization. They captured the frequency and direction of information flow between the product development teams (PDTs) in the project (Fig. 5). Using clustering, they reorganized the teams as shown by the DSM in Fig. 6 to improve interteam integration. The clusters represent four metateams, which overlap to account for the beneficence of some of the PDTs being members of more than one metateam. Two of the PDTs (B and K) are shown twice because they each participate in three metateams. The five PDTs (H, S, T, U, and V) at the bottom of the matrix do not fit neatly into the four metateams; these PDTs need to interface with all of the metateams. Therefore, these PDTs are integrated at a higher organizational level (e.g., in a higher metateam or at the overall project level) with different integrative mechanisms. For example, perhaps a representative from each of these five PDTs will attend each of the four metateam meetings.

The team-based DSM has also supported integration analysis at Boeing [15] and Saab [33] in the context of organizations that develop military aircraft.
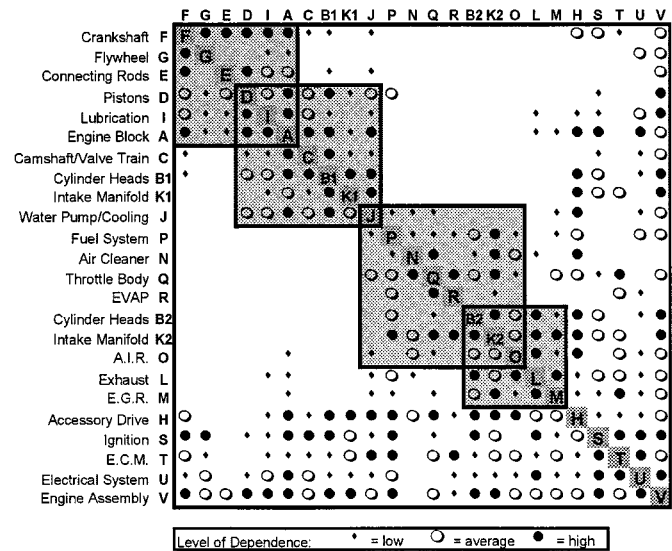
D. Insights

The DSM is an effective representation tool for modeling organizational elements and their relationships. The DSM model then lends itself to integration analysis via clustering approaches. While clustering remains somewhat of an art, several objectives have been identified. The DSM provides a platform for communicating about alternative organizational structures and perspectives. Therefore, its use can improve organizational understanding and innovation. Simply building the DSM encourages disparate people and teams to increase mutual awareness and understanding.

Team-based DSMs can foster creativity and systemization in addressing the following types of questions:

- Why do teams differ in their perceived levels of interaction? Perhaps they do not fully recognize their information suppliers and customers. One team may see itself merely as an information provider or recipient, unaware of exactly how its outputs figure into the overall project or organization.
- Should the teams be organized differently? How should those teams requiring high bandwidth interfaces be divided, combined, and/or integrated? Where might particular integrative mechanisms be prudent? Certain interteam interfaces might indicate the need for colocation, special software tools, common databases, etc. Certain interactions might proceed more efficiently with the aid of a liaison or zone engineer of some sort, or when overseen by an integration team.
- Are currently present integrative mechanisms applied judiciously? Are the liaisons, zone engineers, integration teams, etc. as they now stand appropriately distributed? Are the current plans for software tool, common database, colocation, and other improvements well-founded from an overall organization perspective?
- Are the interactions noted the ones that should be taking place, based on knowledge of the system as a whole? Man-
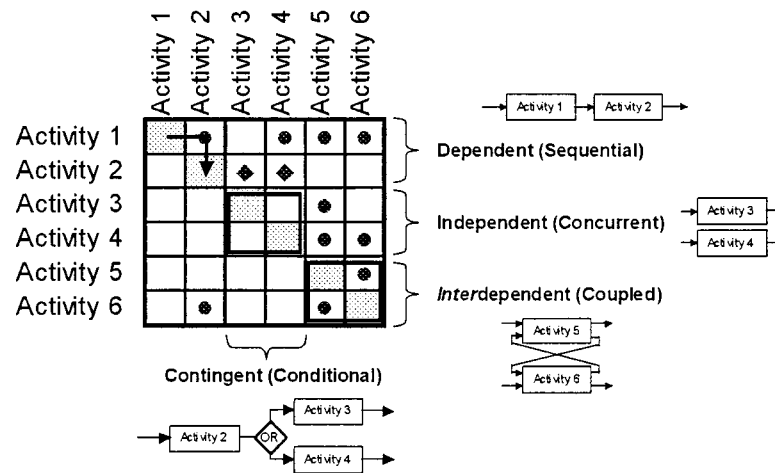
Fig. 7.   Four types of activity relationships in an activity-based DSM (adapted from [20]).

agers should know, based on their own experiences with the teams, whether or not the DSM picture makes sense. If a high level of information exchange is noted, has it really been observed? Are any interactions surprising—ones that may not have been considered before?

IV. INFORMATION FLOW-BASED PROCESS MODELING USING THE ACTIVITY-BASED DSM

A. Motivation

Processes—especially product development processes—are complex systems. A prerequisite to process improvement is process understanding [117]. Process structure or architecture affects process efficiency and effectiveness [23], [25]. Therefore, process architecture can be an important source of competitive advantage [113]. Improved understanding of process architecture can be gained by using process models, particularly ones that support process decomposition and integration analysis. Process decomposition requires an understanding of process activities and their interfaces, because the interfaces are what give a process its added value (versus a mere collection of activities). The greatest leverage in process architecting (and process improvement) is at the interfaces. Therefore, process models must capture flow (an important aspect of Lean principles and value streams) [23], [24], [119].

Product design process activities include analyses, studies, decisions, tests, reviews, etc. Many have found it helpful to think of product development from an information processing perspective, where activities acquire and modify information to produce new or revised information products (e.g., [26], [30], [43], [112], [117]). Product design is characterized by highly coupled, interdependent activities—solving engineering "chicken and egg" problems—which must converge iteratively to an acceptable design solution. When activities begin work without the necessary information, the arrival or change of that information causes rework [18], [19], [25], [37]. Thus, rework results from information arriving at the wrong time, perhaps because of poor activity sequencing, changes, delays, mistakes, etc. Knowing which activities produce and depend on what information can help planners better understand and mitigate

unintentional iteration or rework as a source of schedule risk [19].

The activity-based DSM provides a concise, visual format for understanding and analyzing these issues. Since the activity-based DSM is time-based, it is especially useful for highlighting iteration (feedback) and coupled activities in a design process, a capability traditional PERT/CPM[5] techniques cannot deliver. An activity-based DSM first describes the input/output relationships between activities, showing the dependency structure of a process based on the requisite information flow. Then, a rearranged DSM can prescribe an improved process architecture, such that information is created at the right time and unintentional iteration is minimized.

B. Method

Modeling a process requires two representation steps, followed by integration analysis:
1) decompose the process into activities;
2) document the information flow among the activities (their integration);
3) analyze the sequencing of the activities into a (generally) maximally-feed-forward process flow.

First, the process modeler must determine the boundary of the process to be modeled and how the process will be decomposed. The model grows in size exponentially through successive levels of decomposition. A general guideline is to model a process to the level of detail to which one desires to understand and control the process.

Second, the modeler collects activity data and builds the DSM. Activities are listed in the DSM in roughly chronological order, with upstream or early activities listed in the upper rows, as shown in Fig. 7. Therefore, superdiagonal matrix elements show feedforward information. Subdiagonal elements indicate feedback—the potential for iteration and rework in the process. Information flows in a clockwise direction.[6] If activities in rows

---

[5]Project Evaluation and Review Technique/Critical Path Method

[6]Note that this directionality is the exact inverse of that used in some DSM literature, which displays feedback above the diagonal, i.e., counterclockwise information flow. The two conventions convey equivalent information and are interchangeable by transposing the matrix.
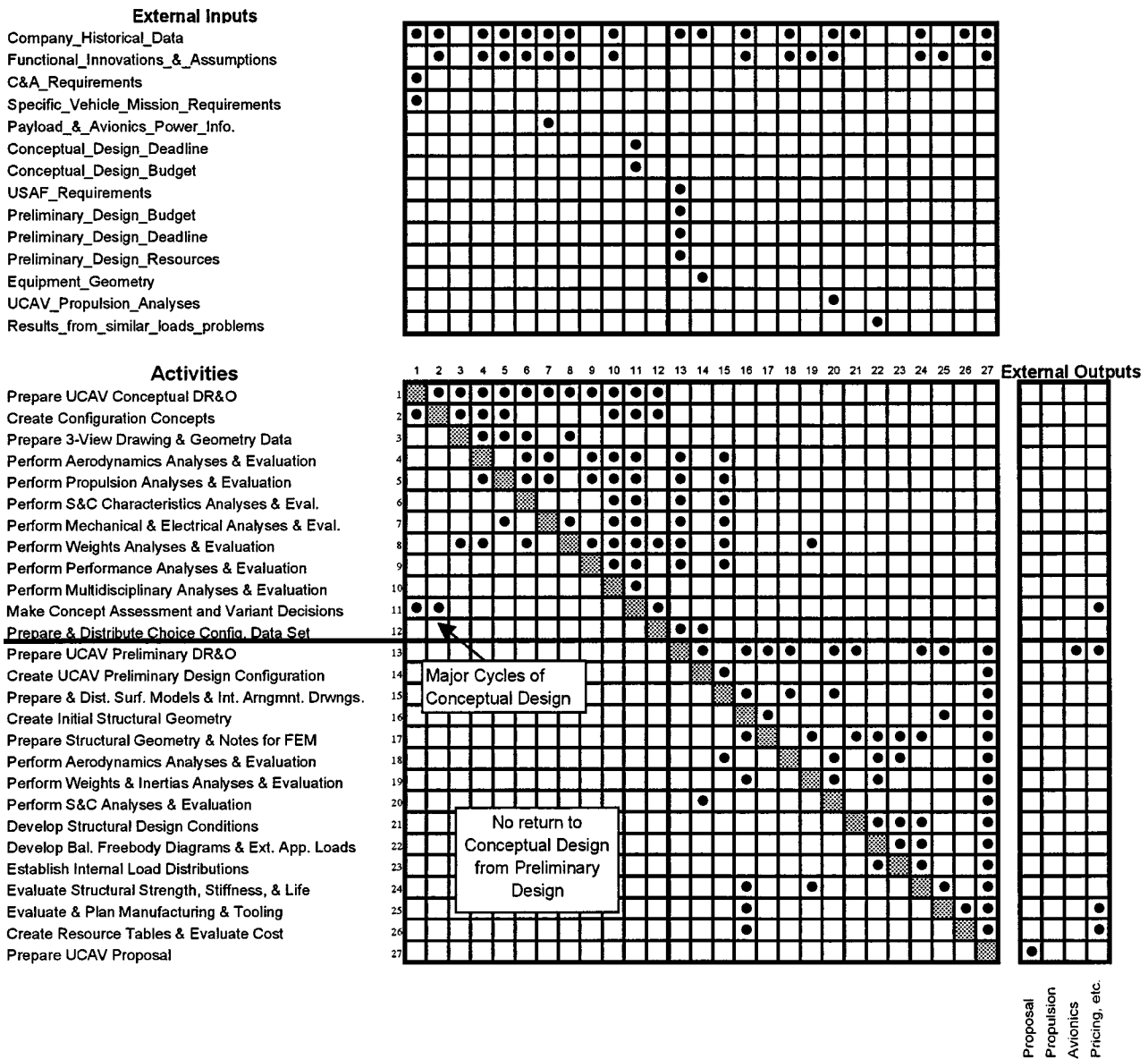
Fig. 8.   Example DSM with external input/output regions (adapted from [18]).

(and corresponding columns) i and j of the DSM have no direct interfaces, they are independent, and 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.

The DSM is built by finding people knowledgeable about each activity and eliciting their expert opinions about the following questions:

1) What outputs or products must the activity produce?
2) Where do these outputs go to (another activity or outside the process)?
3) What inputs does the activity need?
4) Where do these inputs come from (another activity or outside the process)?

The answers to these questions are used to fill in the rows and columns of the DSM. Again, it is often useful to begin by building two DSMs—the first based only on questions one

and two, and the second based on questions three and four. Some effort is usually necessary to achieve consensus between the "supplier" and "consumer" perspectives. However, simply reaching this understanding can benefit a product development organization greatly, since it forces people and teams (who execute activities) to agree on interfaces and deliverables.

For example, a finalized DSM representation might look like Fig. 8, which has been augmented with external input and output regions. These regions allow the model to account for interactions with exogenous elements. By reading across the extended row, one can see where process deliverables go, outside the process. By scanning the extended column, one can see external inputs as well as internal ones. Note that the convenient placement of these external regions depends on the subdiagonal feedback convention. Since DSM literature has had trouble standardizing on one of the two possible conventions, perhaps the usefulness of the external regions will inspire use of the intuitive, clockwise-information-flow convention.

With a reasonably accurate model of a process, one then uses the model to look for improvements, expecting that they can be implemented in the real process. The primary goal in basic DSM analysis is to minimize feedbacks and their scope by restructuring or rearchitecting the process, i.e., by resequencing the rows and columns of the matrix. This widely practiced initial step in analysis is called partitioning, block diagonalization, or block triangularization, and it involves an algorithm for getting the DSM in an upper-triangular form to the extent possible, with a minimum number of subdiagonal marks pulled as close to the diagonal as possible and grouped in blocks. Several researchers have developed partitioning algorithms [47], [55], [61], [108], [110], [114]–[116]. Notice that moving activities upstream to reduce the scope of their feedback loops typifies concurrent engineering [44], [101]. For example, moving a review of the design by manufacturing engineers upstream decreases the scope of the feedback loop that is created if the review reveals issues suggesting rework for upstream activities.

The coupled blocks of activities identified by partitioning the DSM represent several options for process execution. Either all the activities can proceed concurrently, or else certain activities must be chosen to begin before others. Either way, all of the activities will have to converge to a mutually satisfactory solution. To derive an executable process from the information flow model, the project planner must plan for all feedback marks. Executing an activity without a required input from downstream requires making an assumption about that input, which increases the risk level for the activity and the process as a whole. Project planners must be aware of the assumptions they are making and choose those implying the least risk.

Several possibilities exist for resolving the coupled blocks of activities into a feasible execution sequence.

- *Aggregation*: While aggregating two or more activities into a single activity to reduce subdiagonal marks and simplify the DSM may seem attractive, doing so makes the model less useful by "sweeping under the rug" the very issues it should expose.
- *Decomposition*: Decomposing coupled activities can reveal ways to intermingle the lower-level activities that eliminate feedback. For example, see Figs. 11 and 12.
- *Tearing*: Steward [107], [108] introduced the possibility of choosing certain dependencies about which to make assumptions that would allow the process to proceed. The least-damaging assumptions are made first, and their marks are temporarily removed or "torn" from the DSM. Then, the block is repartitioned and, if subdiagonal marks remain, the next least-damaging assumptions are made. This procedure continues until no feedback marks remain in the block of coupled activities.

These basic approaches were augmented by additional mathematical analysis and theoretical extension in Eppinger *et al.* [41], [44]. A common thread in this and Steward's work is the extension of the binary DSM to a numerical DSM by using numbers in the off-diagonal entries to indicate the strength of dependencies, the ease of making assumptions, etc. Rogers built a software tool called Design Manager's Aide for Intelligent Decomposition (DeMAID) to support process structuring based on interfaces of varying strength; DeMAID was subsequently ex-
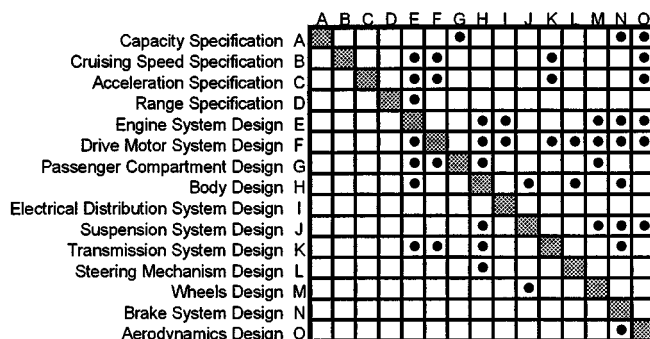

Fig. 9. Activity-based DSM of automobile design process (adapted from [61]).
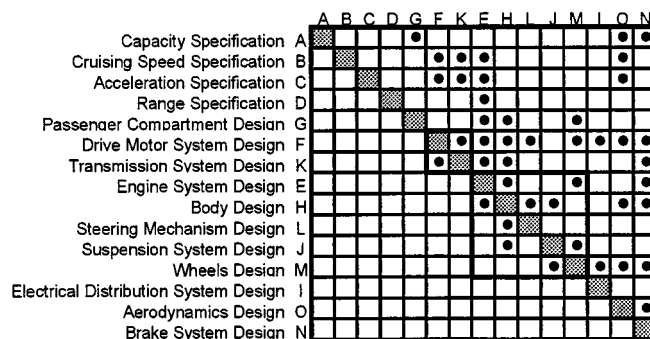

Fig. 10. Resequenced (block-diagonalized) DSM (adapted from [61]).

tended to include a genetic algorithm for sequencing the DSM [39], [73], [89]–[93], [95]. Recently, Denker *et al.* have updated and further explained Steward's work [35]–[38]. Yassine *et al.* [121] use work by Eppinger *et al.* [44], Krishnan *et al.* [57], and Loch and Terwiesch [66] to provide a method for quantifying the off-diagonal dependencies based on information variability and sensitivity. Another paper by Yassine *et al.* [122], provides an updated perspective on tearing. Taking a slightly different approach, Grose [48] attempts to move activities as far upstream as possible in a DSM without creating additional iterations.

More sophisticated analytical models address special issues using the activity-based DSM. Smith *et al.* provide sequential [102], parallel [103], and hybrid [29], [104] iteration models, where design activities are assumed to occur one at a time, all at once, or some of each, respectively. These models analyze project cycle time and highlight which activities contribute the most to delaying design convergence. Others have developed cycle time models for analyzing in detail the information exchange characteristics of a small number of activities [27], [62]. Browning [18] uses a dynamic simulation of the DSM to quantify the level of cost and schedule risk inherent in various process architectures, finding that minimal iteration does not always lead to minimal project duration. Tacconi *et al.* [109] use the DSM as the basis for simulation of a manufacturing process. Ahmadi *et al.* [1] use a Markov model based on the DSM to explore iteration and process structure.

C. Example Application

Kusiak and Wang [61] demonstrate the use of an activity-based DSM to provide a high-level description of a simple, automobile design process (Fig. 9). Fig. 10 shows the restructured
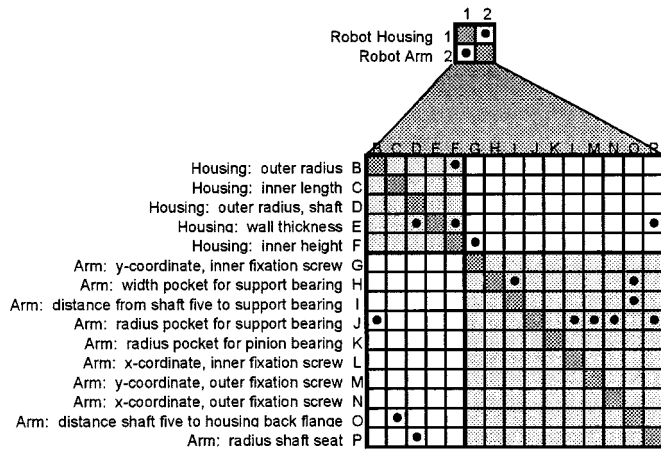
Fig. 11.   Parameter-based DSM for robot arm design activities (adapted from [86]).
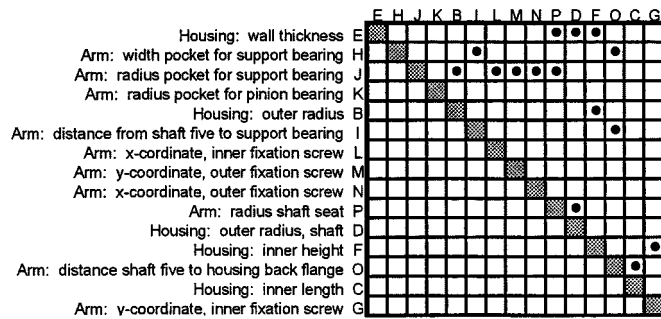

Fig. 12.   Resequenced DSM (adapted from [86]).

DSM after applying a block diagonalization algorithm to minimize feedbacks and isolate two coupled blocks of activities. Supposing it was reasonable to tear all of the subdiagonal marks from each block, we would be left with an executable process model that could be fed into a scheduling software tool such as Microsoft Project for further optimization.

D. Insights

The activity-based DSM has many advantages and can provide insight in several areas. First, it provides process visibility. Glancing down a column reveals where an activity gets its information. A quick examination of a row shows an activity's customers, and, thus, how changing an activity's outputs may affect other activities. Having this information at one's fingertips helps track the ramifications of changes, aiding in configuration control and change notification. Hence, the DSM facilitates interface management. Moreover, the DSM highlights feedbacks and the potential iterations they can cause. The DSM can also be used to determine which activities can be accomplished in parallel without causing additional iteration. Sometimes, project planners, attempting to decrease lead time, overlap activities without first considering their information dependencies, which can result in additional iteration and *increased* lead time.

Activity-based DSMs can be used to document value streams and can reveal nonvalue-added activities [18]. Some projects base work plans on traditional sets of activities, never formally considering the outputs those activities generate. In fact, many

activities produce superfluous outputs. Examining the information actually required to arrive at a final output in an activity-based DSM can help highlight nonessential activities or aspects of activities. Designing a design process around the flow of necessary data rather than around the interactions between traditional activities can eliminate nonvalue-added activities and portions thereof, supporting making processes "lean."

Furthermore, the activity-based DSM model supports process failure modes and effects analysis (FMEA). Product design engineers have long recognized the importance of investigating the ways in which a product can fail and the ramifications of those failures. In this context, FMEA leads to incorporating failure prevention, mitigation, and recovery into the product. FMEA is important to processes as well, serving to help them become more stable, robust, and predictable. By having the capability to show potential process failures in terms of feedbacks to rework certain activities, the DSM is an important tool for process FMEA.

Activity-based DSMs currently have some limitations in process representation that preclude them from entirely replacing other tools. First, a single DSM shows only a single process flow; it does not show all possible flow paths [64]. While the DSM can be augmented with additional symbols to represent contingent information flow (e.g., "♦" instead of "•" in Fig. 7), analysis and improvement of a large process with many possible flow paths becomes intractable without simulation. Second, the DSM does not explicitly show overlapping activities. A Gantt chart remains one of the best representations of activity concurrency. Fortunately, the same activity network data represented by a DSM can be used to generate a Gantt chart.[7] Some additional, general limitations of DSMs are discussed below.

An activity-based DSM provides a systematic method for designing a data-driven project schedule such that information transfer is timely and the design more rapidly converges to the desired performance specifications along multiple dimensions. By highlighting dependencies, feedback, and iteration, an activity-based DSM provides planners with a powerful capability for managing complex projects. It enables improved process understanding, which, in turn, leads to process innovation and improvement.

V. INTEGRATION ANALYSIS OF ACTIVITIES THAT DETERMINE LOW-LEVEL DESIGN PARAMETERS WITH THE PARAMETER-BASED DSM

A. Motivation

Most design process modeling takes place in a top–down fashion, through decomposition. If they begin at "the top," such models rarely reach the lowest levels of design activity, where individual design parameters are determined based on other parameters. Determining these parameters constitutes the lowest-level design activities, and a bottom-up, integrative analysis of these low-level activities can provide process structure insights. Moreover, in some cases, several low-level activities may be assimilated into a single activity by automating the exchange of

---

[7]A DSM can be "stretched" horizontally to obtain a notional Gantt chart (e.g., [10], [54]).

TABLE III
SUMMARY OF DSM TYPE CHARACTERISTICS (ADAPTED FROM [20])

	DSM Type	Representation	Applications	Integration Analysis via
Static	*Component-Based* or *Architecture DSM*	Components in a product architecture and their relationships	System architecting, engineering, design, etc.	Clustering
	Team-Based or *Organization DSM*	Individuals, groups, or teams in an organization and their relationships	Organization design, interface management, application of appropriate integrative mechanisms	
Time-Based	*Activity-Based* or *Schedule DSM*	Activities in a process and their inputs and outputs	Project scheduling, activity sequencing, cycle time reduction, risk reduction, etc.	Sequencing
	Parameter-Based DSM	Parameters to determine a design and their relationships	Low-level process sequencing and integration	

information between the design tools performing the low-level work. Thus, a difference between activity- and parameter-based DSMs is the level of analysis.

Activity- and parameter-based DSMs also differ in the scope of their representations. While an activity-based DSM models a design process, a parameter-based DSM merely documents the physical relationships between the parameters that determine a design. Thus, an activity-based DSM may include reviews, tests, and coordination links that would not typically appear in a parameter-based DSM.

B. Method

The methods of building and analyzing a parameter-based DSM are very similar to those used for an activity-based DSM. Reduced process duration may be the objective of analysis. Another objective can be to minimize what Krishnan *et al.* [58] have called design "quality loss"—the overconstraining of downstream options by upstream decisions. In many cases, parameter-based DSMs are not formally optimized. Instead, they may be used to provide a visual and concise description of interactions between low-level design activities or tools; this description can be used to highlight interface improvement opportunities and to structure integrated "meta-activities."

C. Example Applications and Insights

Rask and Sunnersjö [86] used a parameter-based DSM to describe the relationships between design variables of a robot arm and its housing (Fig. 11). Previously, design parameters were separated into two, coupled "meta-activities": "design arm housing" and "design arm." The DSM model was used to sequence and integrate the low-level activities in a new way (Fig. 12). Now, the two meta-activities have been combined into a single meta-activity, and this allows all of the low-level parameters to be determined sequentially, without iteration. That is, decomposition and integration analysis provided the

means to discover an innovative way to improve the process. Amen *et al.* provide another example in [6].

Black *et al.* [13] applied a parameter-based DSM to automobile brake system design. This work enabled designers to investigate the best initial points for iterated design[8] and helped the company develop a systematic approach to low-level design process planning. Cesiel [28] provides a similar application where designed experiments were used to study the parameter-based design options illuminated by the DSM.

Clearly, parameter-based DSMs have integrative applications. An aerodynamicist at Boeing Commercial Airplane Group used a parameter-based DSM to plan the development of a multivariable wing analysis software tool. Listing subroutines of the new tool (which were once individual analysis tools or algorithms themselves) as rows in the DSM and noting their necessary data exchanges as interfaces, the designer prescribed an efficient process for running, analyzing, and iterating through wing design procedures. Rogers *et al.* [95] demonstrate this application and use a DSM not only to integrate tools but also to represent the status of their execution in real time. The explicit representation of the relationships between design parameters and the capability to sequence these parameters for rapid convergence makes the DSM attractive to the field of multidisciplinary design optimization (MDO—[68], [94], [105]).[9]

VI. DISCUSSION

Table III contrasts the four DSM applications in a succinct format. The rest of this section explores barriers to the practical use of DSMs and the relationships between the four types of DSMs reviewed. These relationships point to interesting avenues for future research.

[8]Better starting points will lead to fewer necessary iterations [85].

[9]The parameter-based DSM can also be applied to many network problems currently treated using graph theoretic techniques, including: sequential, parallel, and distributed computing; chemical reactivity; communication networks; circuit design; and image processing.

A. Barriers To Use of DSMs in Industrial Practice

In practice, DSM-based approaches may have to overcome barriers resulting from organizational inertia, skepticism, "not invented here" syndrome, ignorance, etc. Typically, such attitudes stem from more fundamental problems, such as a lack of system thinking and closed-mindedness. In this author's experience, with a short time of learning and application (say, a few hours), most people are able to understand and develop some intuition for the DSM. However, DSMs also face some specific barriers to implementation and successful use. These barriers pertain mainly to the data requirements of the models, in terms of solicitation, magnitude, and consistency.

DSM models represent extensive system knowledge. Hence, DSMs can be difficult to build, especially initially, as they depict data that are not always at hand, easily gathered, or quickly assimilated. People often find it challenging to provide accurate responses to data collection efforts. People tend to respond with the way they wish organization or process elements would relate, the way they are supposed to relate, or the way they used to relate. People associated with specific elements of a process tend to be more aware of their required inputs (what they need) than of the destination and real use of their outputs. These obstacles to model building reflect general deficiencies stemming from a lack of systems perspective in organizations rather than specific shortcomings of the DSM.

In addition to being challenging to extract, the data required to build a DSM model can be vast in number and problematic to assimilate and verify. Many utilize a centralized approach to model building, where an individual or small team attempts to gather and verify all the required data. However, a more distributed approach makes sense for building many complex system models. A distributed approach builds a DSM hierarchically, with different modelers contributing at each level. Any single modeler works with about ten elements, delegating the modeling of each and then integrating the resulting submodels.

While a somewhat decentralized approach can ameliorate the "amount of data" problem, the submodel assimilation and verification challenges remain. Often, the submodels' external interfaces will not map correctly at higher levels, necessitating issue resolution. Sabbaghian *et al.* [97] developed a web-based tool to facilitate a distributed approach to building and verifying large-scale DSMs.

Building and using DSMs can encourage an organization to address important issues and collect appropriate data that might otherwise be neglected. Once an initial DSM model is built, it can serve as a knowledge base or platform for continued learning, improvement, and innovation. Hence, the data limitation barrier is not unique to DSMs. Rather, building a DSM model can expose a lack of appropriate and efficient data collection and integration in an organization.

While there is no absolute limit to DSM size, practical use provides restrictions. DSMs with fewer than ten elements can often be analyzed via visual inspection and manual manipulation. 50–100–element DSMs are legible on a standard page. Larger DSMs (up to 500 elements) have been built, but they are often "rolled up" or "dithered" (shown in lower resolution) and represented as smaller matrices. Of course, aggregation requires
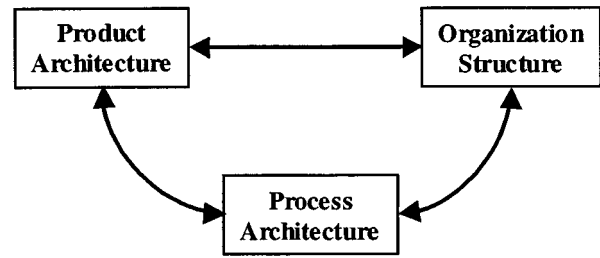

Fig. 13.   Relationships between product, process, and organization structures.

choices about how to integrate elements to represent them as a single element with minimal loss of information in the model. Individuals may have difficulty building DSMs with more than ten elements. Therefore, DSMs with more than ten elements may best be built by integrating smaller DSMs [22].

B. Relationships Between DSM Types

The DSMs discussed in this paper represent three types of systems: products, processes, and organizations. These systems relate to each other as shown in Fig. 13. First, the product architecture has a large influence on the appropriate structure of the product development organization [42], [49], [99], [106], since organizational elements are typically assigned to develop various product components. The ease of organizational partitioning and integration is tied to the nature of the product decomposition. Conversely, an established organization structure can constrain the consideration of alternative product architectures. The product architecture and organization structure relationship can affect an enterprise in several dimensions, including architectural innovation [52], [98]. Better understanding the relationship between product architectures and organization structures is a promising area for further research. DSMs will prove helpful in comparing and contrasting alternative product and organizational configurations.

Second, product architecture is related to the process that will develop it. The structure of a product—including functions, components, interfaces, modularization, etc.—affects how a development process can and should be configured. That is, the product design structure determines the process (activity) design structure [11], [79]. If separate design activities develop separate but coupled modules, then the need for these activities to exchange information should be noted when designing the design process. Conversely, a product development enterprise may find that its legacy development process overly constrains the design of unprecedented products. In an age of emphasis on process "capability and maturity," it would be interesting to see how "mature" processes deal with novel product development. Again, the DSM can be a useful tool in such research. Comparing a component-based DSM to an activity- or parameter-based DSM would inform decisions about the benefits to the design process of architectural modularization.

Third, an interesting relationship exists between the architecture of the product development process and the structure of the product development organization. Using DSMs, Morelli *et al.* [77] show how the interactions between organizational entities,

TABLE IV
DSM TYPES AND ALGORITHMS COMPARED

	Component-Based	Team-Based	Activity-Based	Parameter-Based
Clustering	●	●	3	4
Sequencing	1	2	●	●

such as product teams, can be anticipated based on the structure of predefined development activities. When a process contains coupled activities, the organizational elements with responsibility for executing those activities require integration [14], [76], [88]. Cross-functional teams are one mechanism for addressing these situations (e.g., [75], [84]). An important research question in the field of organization design is how to constitute cross-functional teams. Activity- and team-based DSMs, analyzed in tandem, could lead to a more systematic approach to this issue. Conversely, organizational design constraints may impact the design process. Such constraints might not appear in an activity-based DSM process model, but dual use with the team-based DSM could augment the analysis. Congruence between organization, work, and product architectures has been isolated as a source of competitive advantage [78]. The link between process and organization structures, and the corresponding relationship between the activity- and team-based DSMs, is a promising area for further research.

Although they address similar issues at different levels in the design process, the relationships between activity- and parameter-based DSMs should also be explored. Processes can be described at a low level—"close" to the product architecture—by a parameter-based DSM. Higher-level process representations include analyses, tests, reviews, etc., and are described by the activity-based DSM. Activity-based DSM models are typically built through process decomposition; parameter-based DSMs are often built though low-level activity integration. Theoretically, if modelers begin at both ends, these two approaches can and should meet. Further exploration of hierarchical DSM issues will lead to insights regarding activity and interface decomposition and integration, and regarding transformation from the information flow domain to the process domain. These insights could lead to significant process improvements. For example, some activity couplings can be removed by decomposing the coupled activities, analyzing their low-level steps, and reconfiguring the activity breakdown (e.g., [86]). Furthermore, high-level, legacy processes sometimes contain unnecessary activities, and some high-level activities produce superfluous outputs. Building a process up through the integration of low-level activities and deliverables—which are known to be required because of their closeness to the product design at a parameter level—can help identify extraneous activities and outputs at higher levels.

Finally, the cross-application of clustering and sequencing algorithms illuminates additional opportunities. Component- and team-based DSMs typically contain "symmetric" data, i.e., they are built by asking non-time-based questions such as "Do these components interact? How?," or "Which other teams do

you exchange technical information with?"[10] Activity- and parameter-based DSMs are built by asking asymmetric, precedence questions such as "What information do you need to begin your activity?" and "Where does it come from?" Mixing symmetric and asymmetric models leads to new DSM applications. Table IV indicates the uses of clustering (symmetry) and sequencing (asymmetry) discussed in this paper with ● marks and reveals opportunities for four new DSM applications:

1) Based on component interfaces, one could use a DSM to plan an assembly sequence. One could also study the propagation of error in an assembly and plan tolerance washout zones. Building these models would require asking an asymmetric question such as, "Which components must be assembled before this component can be added?"

2) Based on dynamic organization interfaces, one could use a DSM to study, analyze, and plan the phasing in and phasing out of various organization elements over the span of a project.

3) As discussed above, activity- and team-based DSMs can be used in tandem. Analysis of a hybrid model may require alternating clustering and sequencing. Also, activities often need to be grouped for planning and management purposes, such as for assignment to various groups for execution.

4) Appropriately clustering interdependent design parameters can reveal a preferred integration of low-level activities into higher-level ones. Indeed, clustering may be a key to tying top-down, activity-based DSMs together with bottom-up, parameter-based DSMs.

Each new DSM application enables fresh approaches to address research issues in the respective areas.

VII. CONCLUSION

All four DSM applications reviewed in this paper demonstrate the chief strength of matrix-based approaches: concise, visual representation of complex systems. This paper has emphasized how DSMs facilitate intelligent system decomposition and integration analysis—whether the system is a product, a process, or an organization. The system is analyzed and structured by rearranging the DSM, either by clustering or by sequencing. In many cases, merely building a DSM model provides a useful approach to organizing and visualizing system information. The representation and analysis capabilities of DSMs

---

[10]Some team-based DSM applications, such as Fig. 5, indicate the direction of information flow using the upper- and lower-triangular portions of the matrix. Thus, they may be asymmetric. However, time-sequencing is not an issue since all PDTs exist simultaneously.

contribute to improved system understanding and innovation. Enterprises that recognize, understand, and exploit the relationships between product architecture, organization structure, and process configuration should benefit from significant improvements.

REFERENCES

[1] R. H. Ahmadi, T. A. Roemer, and R. H. Wang, "Structuring product development processes," *Eur. J. Oper. Res.*, vol. 130, pp. 539–558, 2001.

[2] R. H. Ahmadi and H. Wang, "Rationalizing product design development processes," UCLA Anderson Graduate School of Management, Los Angeles, CA, 1994.

[3] C. Alexander, *Notes on the Synthesis of Form*. Cambridge, MA: Harvard Univ. Press, 1964.

[4] S. S. Altus, I. M. Kroo, and P. J. Gage, "A genetic algorithm for scheduling and decomposition of multidisciplinary design problems," in *Proc. 21st ASME Design Automation Conf.*, Boston, MA, 1995.

[5] ——, "A genetic algorithm for scheduling and decomposition of multidisciplinary design problems," *Trans. ASME*, vol. 118, pp. 486–489, 1996.

[6] R. Amen, I. Rask, and S. Sunnersjö, "Matching design tasks to knowledge-based software tools—When intuition does not suffice," in *Proc. ASME Design Engineering Technical Conf. (DETC)*, Las Vegas, NV, 1999.

[7] J. Andersson, "On engineering systems design: A simulation and optimization approach," M. E. thesis, Linköpings Universitet, Linköping, Sweden, 1999.

[8] S. Austin, A. Baldwin, B. Li, and P. Waskett, "Development of the ADePT methodology: An interim report on the link IDAC 100 project," Loughborough University, Dept. of Civil and Building Engineering, Loughborough, U. K., 1998.

[9] ——, "Application of the analytical design planning technique to construction project management," *Project Manage. J.*, vol. 31, pp. 48–59, 2000.

[10] S. Austin, A. Baldwin, and A. Newton, "A data flow model to plan and manage the building design process," *J. Eng. Des.*, vol. 7, pp. 3–25, 1996.

[11] C. Y. Baldwin and K. B. Clark, *Design Rules: The Power of Modularity*. Cambridge, MA: MIT Press, 2000, vol. 1.

[12] O. Becker, J. Ben-Asher, and I. Ackerman, "A method for system interface reduction using N^2 charts," *Syst. Eng.*, vol. 3, pp. 27–37, 2000.

[13] T. A. Black, C. F. Fine, and E. M. Sachs, "A method for systems design using precedence relationships: An Application to automotive brake systems," MIT Sloan School of Management, Cambridge, MA, 3208, 1990.

[14] B. Bolton, "Polyhedral dynamics applied to design and project management," *IEE Proc.*, vol. 135A, pp. 241–244, 1988.

[15] T. R. Browning, "Systematic IPT integration in lean development programs," M.S. thesis , MIT, Cambridge, MA, 1996.

[16] ——, "An introduction to the use of design structure matrices for systems engineering, project management, and organization planning," MIT Lean Aerospace Initiative, Cambridge, MA, WP97-001-18, 1997.

[17] ——, "Integrative mechanisms for multiteam integration: Findings from five case studies," *Syst. Eng.*, vol. 1, pp. 95–112, 1998.

[18] ——, "Modeling and analyzing cost, schedule, and performance in complex system product development," Ph.D. dissertation, MIT, Cambridge, MA, 1998.

[19] ——, "Use of dependency structure matrices for product development cycle time reduction," in *Proc. 5th ISPE Int. Conf. on Concurrent Engineering: Research and Applications*, Tokyo, Japan, 1998, pp. 89–96.

[20] ——, "The design structure matrix," in *Technology Management Handbook*, R. C. Dorf, Ed. Boca Raton, FL: Chapman & Hall/CRCnetBASE, 1999, pp. 103–111.

[21] ——, "Designing system development projects for organizational integration," *Syst. Eng.*, vol. 2, pp. 217–225, 1999.

[22] ——, "From process aggregation to process integration," in *11th Annu. Int. Symp. of INCOSE*, Melbourne, Australia, 2001.

[23] ——, "Modeling the customer value of product development processes," in *11th Annu. Int. Symp. of INCOSE*, Melbourne, Australia, 2001.

[24] 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," in *10th Annu. Int. Symp. INCOSE*, Minneapolis, MN, 2000, pp. 581–589.

[25] T. R. Browning and S. D. Eppinger, "Modeling impacts of process architecture on cost and schedule risk in product development," Lockheed Martin Aeronautics, Fort Worth, TX, Working Paper, 2001.

[26] T. Burns and G. M. Stalker, *The Management of Innovation*. London, U.K.: Tavistock, 1961.

[27] M. Carrascosa, S. D. Eppinger, and D. E. Whitney, "Using the design structure matrix to estimate product development time," in *Proc. ASME Des. Eng. Tech. Conf. (Design Automation Conf.)*, Atlanta, GA, 1998.

[28] D. S. Cesiel, "A structured approach to calibration development for automotive diagnostic systems," Mgmt./E.E. thesis, MIT, Cambridge, MA, 1993.

[29] L. C. Cheung, R. P. Smith, and Z. B. Zabinsky, "Optimal scheduling in the engineering design process," University of Washington, Seattle, WA, 1998.

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

[31] P. J. Clarkson and J. R. Hamilton, "'Signposting': A parameter-driven task-based model of the design process," *Res. Eng. Des.*, vol. 12, pp. 18–38, 2000.

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

[33] M. L. Danilovic, "Leadership and organization of integration in product development," Ph.D. dissertation, Linköpings Universitet, Linköping, Sweden, 1999.

[34] A. Deaton and J. Muellbauer, *Economics and Consumer Behavior*. New York: Cambridge Univ. Press, 1980.

[35] S. Denker and D. Steward, "Using the design structure method," Problematics, Brookline, MA, 1995.

[36] ——, "Using information exchange structure to design for concurrent design," Problematics, Brookline, MA, 1996.

[37] S. Denker, D. Steward, and T. Browning, "Planning Concurrency and managing iteration in projects," *CQM J.*, vol. 8, pp. 55–62, 1999.

[38] S. Denker and D. V. Steward, "Using concurrent engineering for competitive advantage," Problematics, Brookline, MA, 1996.

[39] *DoD Integrated Product and Process Development Handbook*, Office of the Under Secretary of Defense (Acquisition and Technology), Washington, DC, 1998.

[40] *Systems Engineering Management Guide*, Defense Systems Management College, Fort Belvoir, VA, 1990.

[41] S. D. Eppinger, "Model-based approaches to managing concurrent engineering," *J. Eng. Des.*, vol. 2, pp. 283–290, 1991.

[42] ——, "A planning method for integration of large-scale engineering systems," in *Inte. Conf. on Engineering Design ICED-97*, Tampere, Finland, 1997, pp. 199–204.

[43] ——, "Innovation at the speed of information," in *Harvard Bus. Rev.*, 2001, vol. 79, pp. 149–158.

[44] S. D. Eppinger, D. E. Whitney, R. P. Smith, and D. A. Gebala, "A model-based method for organizing tasks in product development," *Res. Eng. Des.*, vol. 6, pp. 1–13, 1994.

[45] C. I. G. Fernandez, "Integration analysis of product architecture to support effective team co-location," ME thesis, MIT, Cambridge, MA, 1998.

[46] E. P. C. Fernando, "Use of interdependency matrix for expediting implementation of an integrated development programme in a developing country," in *2nd Int. Congr. Project Planning by Network Analysis*, Amsterdam, The Netherlands, 1969, pp. 76–85.

[47] D. A. Gebala and S. D. Eppinger, "Methods for analyzing design procedures," in *Proc. ASME 3rd Int. Conf. on Design Theory and Methodology*, 1991, pp. 227–233.

[48] D. L. Grose, "Reengineering the aircraft design process," in 5th AIAA/USAF/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization, Panama City Beach, FL, 1994.

[49] R. K. Gulati and S. D. Eppinger, "The coupling of product architecture and organizational structure decisions," MIT, Cambridge, MA, Working Paper No. 151, 1996.

[50] A.-P. Hameri, J. Nihtilä, and J. Rehn, "Document viewpoint on one-of-a-kind delivery process," *Int. J. Prod. Res.*, vol. 37, pp. 1319–1336, 1999.

[51] M. Hayes, "The role of activity precedence relationships in node-oriented networks," in *2nd Int. Congr. for Project Planning by Network Analysis*, Amsterdam, The Netherlands, 1969, pp. 128–146.

[52] R. M. Henderson and K. B. Clark, "Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms," *Administ. Sci. Quart.*, vol. 35, pp. 9–30, 1990.

[53] P. Huovila, L. Koskela, L. Pietiläinen, and V.-P. Tanhuanpää, "Use of the design structure matrix in construction," in 3rd Int. Workshop on Lean Construction, Albuquerque, NM, 1995.

[54] K. Kähkönen, V.-P. Tanhuanpää, and S. Leino, "Design process analysis, optimization and management—A practical tool for the construction and engineering projects," VTT Building Technology, Finland, 1997.

[55] E. Kehat and M. Shacham, "Chemical process simulation programs-2: Partitioning and tearing of system flowsheets," *Process Technol. Int.*, vol. 18, pp. 115–118, 1973.

[56] L. Koskela, G. Ballard, and V.-P. Tanhuanpää, "Toward lean design management," in 5th Annu. Conf. of the International Group for Lean Construction (IGLC-5), 1997.

[57] V. Krishnan, S. D. Eppinger, and D. E. Whitney, "A model-based framework to overlap product development activities," *Manage. Sci.*, vol. 43, pp. 437–451, 1997.

[58] ——, "Simplifying iterations in cross-functional design decision making," *J. Mech. Des.*, vol. 119, pp. 485–493, 1997.

[59] A. Kusiak, *Engineering Design: Products, Processes, and Systems*. San Diego, CA: Academic, 1999.

[60] A. Kusiak and J. Wang, "Decomposition of the design process," *J. Mech. Des.*, vol. 115, pp. 687–695, 1993.

[61] ——, "Efficient organizing of design activities," *Int. J. Prod. Res.*, vol. 31, pp. 753–769, 1993.

[62] T. Lanner and G. Zeichen, "Dynamic product development," in *Proc. SPIE Conf. on Intelligent Systems in Design and Manufacturing*, Boston, MA, 1998, pp. 64–69.

[63] R. J. Lano, *A Technique for Software and Systems Design*. New York: North-Holland, 1979.

[64] N. Larson and A. Kusiak, "Managing design processes: A risk assessment approach," *IEEE Trans. Syst., Man, Cybern.*, vol. 26, pp. 749–759, 1996.

[65] W. P. Lewis and L. Cangshan, "The timely allocation of resources in the concurrent design of new products," *J. Eng. Des.*, vol. 8, pp. 3–17, 1997.

[66] C. H. Loch and C. Terwiesch, "Communication and uncertainty in concurrent engineering," *Manage. Sci.*, vol. 44, pp. 1032–1048, 1998.

[67] J. W. Lorsch and P. R. Lawrence, "Managing group and intergroup relations," in *Irwin Series in Management and the Behavioral Sciences*, L. L. Cummings, E. K. Warren, and J. F. Mee, Eds. Homewood, IL: Richard D. Irwin, 1972.

[68] B. J. Makins and D. W. Miller, "Web-based aerospace system evaluation software: The development and assessment of conceptual space missions," in *Proc. 10th Annu. Int. Symp. of INCOSE*, Minneapolis, MN, 2000, pp. 167–174.

[69] J. Malmström, "Matrix Based methods for product and process structuring," Licentiate of Eng. thesis, Chalmers University of Technology, Göteborg, Sweden, 1998.

[70] J. Malmström and J. Malmqvist, "Trade off analysis in product structures: A case study at Celsius Aerotech," Machine and Vehicle Design, Chalmers University of Technology, Göteborg, Sweden, 1998.

[71] J. Malmström, P. Pikosz, and J. Malmqvist, "The complementary roles of IDEF0 and DSM for the modeling of information management processes," *Concurrent Eng.: Res. Applicat.*, vol. 7, pp. 95–103, 1999.

[72] K. R. McCord and S. D. Eppinger, "Managing the integration problem in concurrent engineering," MIT Sloan School of Management, Cambridge, MA, Working Paper No. 3594, 1993.

[73] C. McCulley and C. L. Bloebaum, "A genetic tool for optimal design sequencing in complex engineering systems," *Structural Optimization*, vol. 12, pp. 186–201, 1996.

[74] N. F. Michelena and P. Y. Papalambros, "Optimal model-based decomposition of powertrain system design," in ASME Adv. Des. Autom., 1995.

[75] S. A. Mohrman, S. G. Cohen, J. Allan, and M. Mohrman, *Designing Team-Based Organizations*. San Francisco, CA: Jossey-Bass, 1995.

[76] J. Morabito, I. Sack, and A. Bhate, *Organization Modeling: Innovative Architectures for the 21st Century*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[77] M. D. Morelli, S. D. Eppinger, and R. K. Gulati, "Predicting technical communication in product development organizations," *IEEE Trans. Eng. Manage.*, vol. 42, pp. 215–222, 1995.

[78] D. A. Nadler and M. L. Tushman, *Competing by Design: The Power of Organizational Architecture*. New York: Oxford Univ. Press, 1997.

[79] P. Nightingale, "The product-process-organization relationship in complex development projects," *Res. Policy*, vol. 29, pp. 913–930, 2000.

[80] M. Nour and J. Scanlan, "Modeling and simulating product development process," in *Proc. 6th Int. Conf. on Concurrent Enterprising*, Toulouse, France, 2000, pp. 111–118.

[81] S. M. Osborne, "Product development cycle time characterization through modeling of process iteration," M.S. thesis, MIT, Cambridge, MA, 1993.

[82] T. U. Pimmler and S. D. Eppinger, "Integration Analysis of Product Decompositions," in *Proc. ASME 6th Int. Conf. on Design Theory and Methodology*, Minneapolis, MN, 1994.

[83] R. D. Pinkett, "Product development process modeling and analysis digital wireless telephones," S.M. thesis, MIT, Cambridge, MA, 1998.

[84] R. A. Pomponi, "The organization of integrated product teams (literature review)," MIT Lean Aerospace Initiative, Cambridge, MA, Working Paper No. WP97-002-34, 1997.

[85] N. Ramachandran, N. A. Langrana, L. I. Steinberg, and V. R. Jamalabad, "Initial design strategies for iterative design," *Res. Eng. Des.*, vol. 4, pp. 159–169, 1992.

[86] I. Rask and S. Sunnersjö, "Design structure matrices for the planning of rule-based engineering systems," in *Proc. Eur. Conf. on Integration in Manufacturing*, Göteborg, Sweden, 1998.

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

[88] H. R. Riahi and K. S. Pawar, "Modeling the dependency structure and optimization of concurrency in CE," in *5th Int. Conf. on Concurrent Enterprising*, The Hague, 1999.

[89] J. L. Rogers, "A knowledge-based tool for multilevel decomposition of a complex design problem," NASA, Hampton, VA, TP-2903, 1989.

[90] ——, "DeMAID/GA user's guide-design manager's aid for intelligent decomposition with a genetic algorithm," NASA, Hampton, VA, TM-110 241, 1996.

[91] ——, "Integrating a genetic algorithm into a knowledge-based system for ordering complex design processes," NASA, Hampton, VA, TM-110 247, 1996.

[92] ——, "Reducing design cycle time and cost through process resequencing," in *Proc. Int. Conf. on Engineering Design*, Tampere, Finland, 1997.

[93] ——, "Tools and techniques for decomposing and managing complex design projects," *J. Aircraft*, vol. 36, pp. 266–274, 1999.

[94] J. L. Rogers and C. L. Bloebaum, "Reducing time and cost in the design process," in *Proc. AIAA Professional Development Short Course*, Bellevue, WA, 1996.

[95] J. L. Rogers, A. O. Salas, and R. P. Weston, "A web-based monitoring system for multidisciplinary design projects," in *7th AIAA/USAF/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization*, St. Louis, MO, 1998.

[96] G. J. Rushton and A. Zakarian, "Modular vehicle architectures: A systems approach," in *10th Annu. Int. Symp. of INCOSE*, Minneapolis, MN, 2000, pp. 29–35.

[97] N. Sabbaghian, S. Eppinger, and E. Murman, "Product development process capture & display using web-based technologies," in *IEEE Int. Conf. on Syst., Man, Cybern.*, San Diego, CA, 1998, pp. 2664–2669.

[98] R. Sanchez and J. T. Mahoney, "Modularity, flexibility, and knowledge management in product and organization design," *IEEE Eng. Manage. Rev.*, pp. 50–61, 1997.

[99] G. Seliger, H. Karl, and H. Weber, "Cooperative design, manufacturing and assembly of complex products," *Annals CIRP*, vol. 46, pp. 67–70, 1997.

[100] M. W. Sequeira, "Use of the design structure matrix in the improvement of an automobile development process," M.S. thesis, MIT, Cambridge, MA, 1991.

[101] K. J. Singh, J. W. Erkes, J. Czechowski, J. W. Lewis, and M. G. Issac, "DICE approach for reducing product development cycle," in *Proc. Worldwide Passenger Car Conf. and Expo*, Dearborn, MI, 1992, pp. 141–150.

[102] R. P. Smith and S. D. Eppinger, "Identifying controlling features of engineering design iteration," *Manage. Sci.*, vol. 43, pp. 276–293, 1997.

[103] ——, "A predictive model of sequential iteration in engineering design," *Manage. Sci.*, vol. 43, pp. 1104–1120, 1997.

[104] ——, "Deciding between sequential and parallel tasks in engineering design," *Concurrent Eng.: Res. Applicat.*, vol. 6, pp. 15–25, 1998.

[105] J. Sobieszczanski-Sobieski, "Multidisciplinary optimization for engineering systems: Achievements and potential," NASA Langley Research Center, Hampton, VA, TM-101 566, 1989.

[106] M. E. Sosa, "Analyzing the effects of product architecture on technical communication in product development organizations," Ph.D. dissertation, MIT, Cambridge, MA, 2000.

[107] D. V. Steward, "The Design structure system: A method for managing the design of complex systems," *IEEE Trans. Eng. Manage.*, vol. 28, pp. 71–74, 1981.

[108] ——, *Systems Analysis and Management: Structure, Strategy, and Design*.   New York: PBI, 1981.

[109] D. A. Tacconi and F. L. Lewis, "A new matrix model for discrete event systems: application to simulation," *IEEE Control Syst. Mag.*, vol. 17, pp. 62–71, 1997.

[110] D. Tang, L. Zheng, Z. Li, D. Li, and S. Zhang, "Re-engineering of the design process for concurrent engineering," *Comput. Ind. Eng.*, vol. 38, pp. 479–491, 2000.

[111] R. J. Thomas and N. Worren, "Applying the dependency structure matrix to high-level organizational architectures," in *2nd MIT Design Structure Matrix Workshop*, Cambridge, MA, 2000.

[112] K. T. Ulrich and S. D. Eppinger, *Product Design and Development*, Second ed.   New York: McGraw-Hill, 2000.

[113] E. von Hippel, "Task partitioning: An innovation process variable," *Res. Policy*, vol. 19, pp. 407–418, 1990.

[114] J. N. Warfield, "Binary matrices in system modeling," *IEEE Trans. Syst., Man, Cybern.*, vol. 3, pp. 441–449, 1973.

[115] ——, *Societal Systems: Planning, Policy, and Complexity*.   New York: Wiley, 1976.

[116] R. L. Weil and P. C. Kettler, "Rearranging matrices to block-angular form for decomposition (and other) algorithms," *Manage. Sci.*, vol. 18, pp. 98–108, 1971.

[117] D. E. Whitney, "Designing the design process," *Res. Eng. Des.*, vol. 2, pp. 3–13, 1990.

[118] D. E. Whitney, Q. Dong, J Judson, and G. Mascoli, "Introducing knowledge-based engineering into an interconnected product development process (DETC99/DTM-8741)," in *Proc. ASME Design Engineering Technical Conf.*, Las Vegas, NV, 1999.

[119] J. P. Womack and D. T. Jones, *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*.   New York: Simon & Schuster, 1996.

[120] R. R. Yager, "Intelligent control of the hierarchical agglomerative clustering process," *IEEE Trans. Syst., Man, Cybern., B*, vol. 30, pp. 835–845, 2000.

[121] A. Yassine, D. Falkenburg, and K. Chelst, "Engineering design management: An information structure approach," *Int. J. Prod. Res.*, vol. 37, 1999.

[122] A. A. Yassine, D. E. Whitney, J. Lavine, and T. Zambito, "Do-it-right-first-time (DRFT) approach to DSM restructuring," in *Proc. ASME 2000 Int. Design Engineering Technical Conf. (DTM)*, Baltimore, MD, 2000.

Tyson R. Browning received the B.S. degree in engineering physics from Abilene Christian University, Abilene, TX, and two M.S. degrees and an interdisciplinary Ph.D. degree in technology, management and policy from the Massachusetts Institute of Technology, Cambridge.

He is currently a Senior Project Manager in the Enterprise Productivity Strategy Group, Lockheed Martin Aeronautics Company, Fort Worth, TX. In this position, he applies Lean thinking to help determine continuous improvement strategy, especially for processes such as product development, and also serves as a point of contact with university and government research on Lean. He also provides internal consulting and conducts applied research on engineering and enterprise process development. Previously, he worked with the Lean Aerospace Initiative at MIT, conducting research at Lockheed Martin, General Electric, Boeing, Raytheon, Sundstrand, and Daimler Chrysler. Several of his papers on organizational integration, risk management, the design structure matrix, and process modeling have been published and are forthcoming.