

An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method

Ali A. Yassine

Product Development Research Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL. 61801
yassine@uiuc.edu

Brief description of the author

Ali Yassine is an assistant professor at the Department of General Engineering at the University of Illinois at Urbana-Champaign (UIUC). He is the director of the Product Development Research Laboratory and an affiliated professor with the Technology Entrepreneur Center (TEC). His research concentrates on analyzing a set of product development problems, including the management of iteration and overlapping decisions.

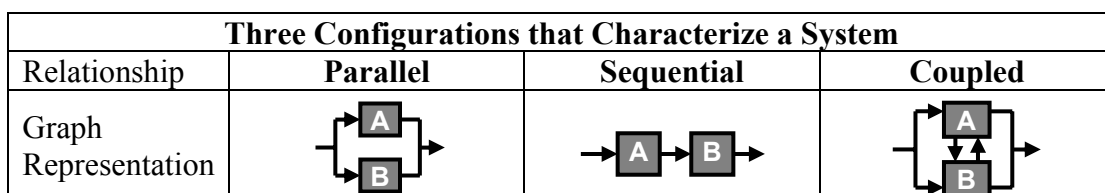
Abstract

The design and development of complex engineering products require the efforts and collaboration of hundreds of participants from diverse backgrounds resulting in complex relationships among both people and tasks. Many of the traditional project management tools (PERT, Gantt and CPM methods) do not address problems stemming from this complexity. While these tools allow the modeling of sequential and parallel processes, they fail to address interdependency (feedback and iteration), which is common in complex product development (PD) projects. To address this issue, a matrix-based tool called the Design Structure Matrix (DSM) has evolved. This method differs from traditional project-management tools because it focuses on representing information flows rather than work flows. The DSM method is an information exchange model that allows the representation of complex task (or team) relationships in order to determine a sensible sequence (or grouping) for the tasks (or teams) being modeled. This article will cover how the basic method works and how you can use the DSM to improve the planning, execution, and management of complex PD projects using different algorithms (i.e., partitioning, tearing, banding, clustering, simulation, and eigenvalue analysis).

Introduction: matrices and projects

Consider a system (or project) that is composed of two elements /sub-systems (or activities/phases): element "A" and element "B". A graph may be developed to represent this system pictorially. The graph is constructed by allowing a vertex/node on the graph to represent a system element and an edge joining two nodes to represent the relationship between two system elements. The directionality of influence from one element to another is captured by an arrow instead of a simple link. The resultant graph is called a directed graph or simply a digraph. There are three basic building blocks for describing the relationship amongst system elements: parallel (or concurrent), sequential (or dependent) and coupled (or interdependent) (fig. 1)

Fig.1



The matrix representation of a digraph is a binary (i.e. a matrix populated with only zeros and ones) square (i.e. a matrix with equal number of rows and columns) matrix with m rows and columns, and n non-zero elements, where m is the number of nodes and n is the number of edges in the digraph. The matrix layout is as follows: the system elements names are placed down the side of the matrix as row headings and across the top as column headings in the same order. If there exists an edge from node i to node j , then the value of element ij (column i , row j) is unity (or flagged with a mark such as “X” or “●”). Otherwise, the value of the element is zero (or left empty). In the binary matrix representation of a system, the diagonal elements of the matrix do not have any interpretation in describing the system, so they are usually either left empty or blacked out (fig. 2)

Fig. 2

Three Configurations that Characterize a System						
Relationship	Parallel			Sequential		
DSM Representation		A	B		A	B
	A	■	■	A	■	■
	B	■	■	B	X	■

Binary matrices for system modeling are useful in systems modeling because they can represent the presence or absence of a relationship between pairs of elements of a system. A major advantage of the matrix representation over the digraph is in its compactness and ability to provide a systematic mapping among system elements that is clear and easy to read regardless of size.

If the system to be represented is a project composed of a set of tasks to be performed, you can use the matrix approach with the Design Structure Matrix (DSM) methodology. Rows and columns are headed with the complete list of tasks to be performed in the project. Marks in the matrix explain if there are information-based relations among the tasks, and, if so, which kind (sequential, parallel, coupled). Marks in a single row (of the DSM) represent all of the tasks whose output is required to perform the task corresponding to that row. Similarly, reading down a specific column reveals which task receives information from the task corresponding to that column. If the order of elements in the matrix depict a time sequence, then marks below the diagonal represent forward information transfer to later (i.e. downstream) tasks. This kind of mark is called forward mark or forward information link. Marks above the diagonal depict information fed back to earlier listed tasks (i.e. feedback mark) and indicate that an upstream task is dependent on a downstream task.

In the parallel configuration, the system elements do not interact with each other. Understanding the behavior of the individual elements allow us to completely understand the behavior of the system. If the system is a project, then system elements would be project tasks to be performed. As such, activity B is said to be independent of activity A and no information exchange is required between the two activities. In the sequential configuration, one element influences the behavior or decision of another element in a uni-directional fashion. That is, the design parameters of system element B are selected based on the system element A design parameters. Again, in terms of project tasks, task A has to be performed first before task B can start. Finally, in the coupled system, the flow of influence or information is intertwined: element A influences B and element B influences A. This would occur if parameter A could not be determined (with certainty) without first knowing parameter B and B could not be determined without knowing A. This cyclic dependency is called "Circuits" or "Information Cycles".

Box 1: Four different types of data that can be represented in a DSM (Browning, 2001)

With a matrix you can represent information relations among components of a product, among teams concurrently working on a project, among activities (the structure of a project we are mainly focused on in this article), and among parameters.

DSM Data Types	Representation	Application	Analysis Method
Task-based	Task/Activity input/output relationships	Project scheduling, activity sequencing, cycle time reduction	Partitioning, Tearing, Banding, Simulation and Eigenvalue Analysis
Parameter-based	Parameter decision points and necessary precedents	Low level activity sequencing and process construction	Partitioning, Tearing, Banding, Simulation and Eigenvalue Analysis
Team-based	Multi-team interface characteristics	Organizational design, interface management, team integration	Clustering
Component-based	Multi-component relationships	System architecting, engineering and design	Clustering

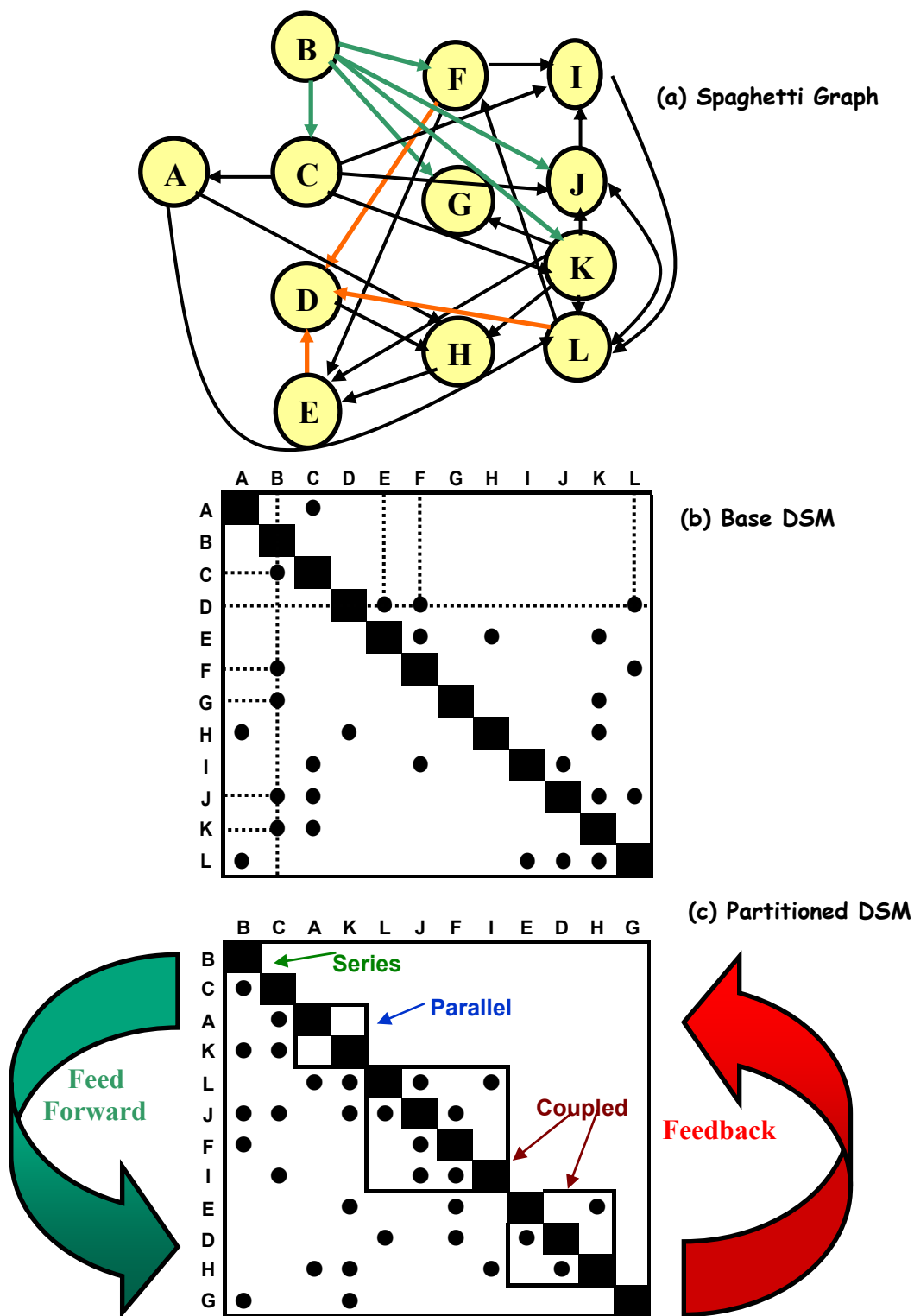
A DSM representation of a project – Task-based DSM and Parameter-based DSMs

A DSM is a compact, matrix representation of a project network. The matrix contains a list of all constituent activities and the corresponding information exchange patterns. That is, what information pieces (parameters) are required to start a certain activity and where does the information generated by that activity feed into (i.e. which other tasks within the matrix utilize the output information). The DSM provide insights about how to manage a complex project and highlights issues of information needs and requirements, task sequencing, and iterations. A sample DSM is shown below (fig. 3).

For example, in Figure 3, B feeds C, F, G, J and K, while D is fed by E, F and L. As mentioned earlier, all marks above the diagonal are feedback marks. Feedback marks correspond to required inputs that are not available at the time of executing a task. In this case, the execution of the dependent task will be based on assumptions regarding the status of the input tasks. As the project unfolds these assumptions are revised in light of new information, and the dependent task is re-executed if needed. It is worth noting how easy it is to determine feedback relationships in the DSM compared to the graph, which makes the DSM a powerful, but simple, graphic representation of a complex system or project.

The matrix can be manipulated in order to eliminate or reduce the feedback marks. This process is called partitioning and will be discussed next (Steward, 1981; Yassine et al., 1999). When this is done, a transparent structure for the network starts to emerge, which allows better planning of the PD project. We can see which tasks are sequential, which ones can be done in parallel, and which ones are coupled or iterative (see Figure 3c).

Fig. 3



Once the DSM is partitioned, tasks in series are identified and executed sequentially. Parallel tasks are also exposed and can be executed concurrently. For the coupled ones, upfront planning is necessary. For example, we would be able to develop an iteration plan by determining what tasks should start the iteration process based on an initial guess or estimate of a missing piece of information. In Figure 3c, block E-D-H can be executed as follows: task E starts with an initial guess on H's output, E's output is fed to task D, then D's output is fed to task H, and finally H output is fed to task E. At this point, task E compares H's output to the initial guess made and decides if an extra iteration is required or not depending on how far the initial estimate deviated from the latest information received from H. This iterative process proceeds until convergence occurs (modeling the convergence process is discussed later in the paper).

Consider the parameter-based DSM shown in Figure 4. This example is taken from a larger DSM of 105 elements (Smith and Eppinger, 1997). The elements in the DSM represent the original sequence of design parameters or decisions that were made. Notice the large number of feedback marks (over a dozen) and notice also that if rework is triggered by any feedback mark, this will impact the all the other tasks. The DSM was partitioned and the result is shown in Figure 4 (bottom). The partitioned DS_m shows an improvement in the flow of design decisions and a faster, less iterative, process. Notice how are the iteration loops are now confined to two smaller sets of tasks.

Fig. 4: A brake example: Original DSM (top) and Partitioned DSM (bottom)

		1	2	3	4	5	6	7	8	9	10	11	12	13
Customer_Requirements	1	1												
Wheel Torque	2		2		X									
Pedal Mech. Advantage	3	X		3	X	X			X		X			X
System_Level_Parameters	4	X			4									
Rotor Diameter	5	X	X	X	X	5		X	X		X	X		X
ABS Modular Display	6		X				6			X				
Front_Lining_Coeff._of_Friction	7			X	X	X		7	X		X			X
Piston-Rear Size	8		X		X				8		X			
Caliper Compliance	9			X	X					9	X			X
Piston- Front Size	10		X		X				X		10			
Rear Lining Coef of Friction	11			X	X	X			X		X	11		X
Booster - Max. Stroke	12												12	X
Booster Reaction Ratio	13		X	X	X	X		X	X	X	X	X	X	13

		1	4	2	10	8	3	11	7	13	5	12	9	6
Customer_Requirements	1	1												
System_Level_Parameters	4	X	4											
Wheel Torque	2		X	2										
Piston- Front Size	10		X	X	10	X								
Piston-Rear Size	8		X	X	X	8								
Pedal Mech. Advantage	3	X	X		X	X	3			X	X			
Rear Lining Coef of Friction	11		X		X	X	X	11		X	X			
Front_Lining_Coeff._of_Friction	7		X		X	X	X		7	X	X			
Booster Reaction Ratio	13		X	X	X	X	X	X	X	13	X			
Rotor Diameter	5	X	X	X	X	X	X	X	X	X	5			
Booster - Max. Stroke	12									X		12		
Caliper Compliance	9		X		X		X			X			9	
ABS Modular Display	6												X	6

Building / Creating the Design Structure Matrix

The success of the DSM method is determined by appropriate system decomposition and by the accuracy of the dependence relationships collected. Therefore, it is vital to carefully decompose the system under study into meaningful system elements (i.e. subsystems or modules). An appropriate decomposition can be established using two main approaches:

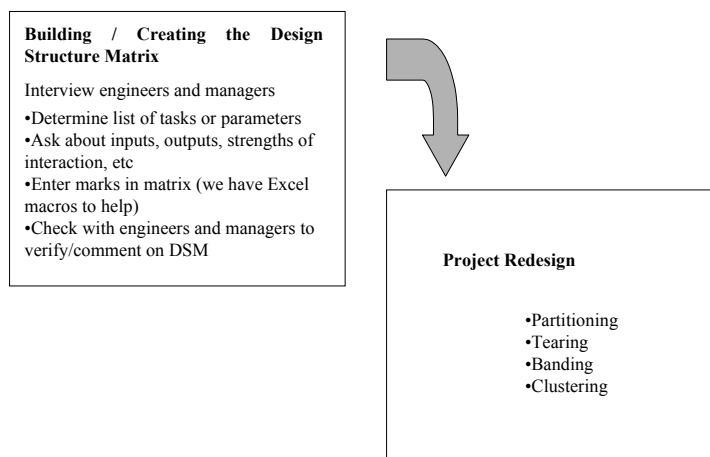
- Convert existing documentation: design manuals, process sheets, project schedules, IDEF models, ... etc.
- Structured expert interviews.

We recommend a hybrid approach where a starting DSM is built from existing documentation, and then a subsequent step of expert interviews is used to supplement and validate the initial DSM. In the second step, a group of managers/experts are gathered from different functional groups of an organization and asked to collectively list the different sub-systems that comprise the system as a whole. The decomposition can be hierarchical or non-hierarchical (sometimes called network decomposition). In the hierarchical decomposition, the system can be divided into sub-systems or modules and those modules are, in turn, divided into finer components. In the network decomposition, a system hierarchy is not evident.

Once the appropriate system elements or set of activities that comprise a project have been identified, they are listed in the DSM as row and column labels in the same order. The elements within the matrix are then identified by asking the appropriate manager/expert in the group for the minimum set of parameters (taken from the list) that influence their own sub-system and contribute to its behavior. In a task-based DSM, this can be the minimum set of activities that need to be performed before the activity under questioning can be started. In a parameter based DSM, the rows and columns are design parameters that drive the design or define the system and the managers/experts can then be asked to define the precedence relationships between the listed parameters. These tasks/parameters/elements are marked in the DSM by an 'X' or “●”.

After having represented the process, you can start redesigning it, using the different criteria described below. Fig. 5 shows the activities included in a DSM-based project redesigning process.

Fig. 5



DSM Partitioning

Partitioning is the process of manipulating (i.e. reordering) the DSM rows and columns such that the new DSM arrangement does not contain any feedback marks; thus, transforming the DSM into a

lower triangular form. For complex engineering systems, it is highly unlikely that simple row and column manipulation will result in a lower triangular form. Therefore, the analyst's objective changes from eliminating the feedback marks to moving them as close as possible to the diagonal (this form of the matrix is known as block triangular). In doing so, fewer system elements will be involved in the iteration cycle resulting in a faster development process.

There are several approaches used in DSM partitioning. However, they are all similar with a difference in how do they identify cycles (loops or circuits) of information. All partitioning algorithms proceed as follows:

1. Identify system elements (or tasks) that can be determined (or executed) without input from the rest of the elements in the matrix. Those elements can easily be identified by observing an empty row in the DSM. Place those elements in the top of the DSM. Once an element is rearranged, it is removed from the DSM (with all its corresponding marks) and step 1 is repeated on the remaining elements.
2. Identify system elements (or tasks) that deliver no information to other elements in the matrix. Those elements can easily be identified by observing an empty column in the DSM. Place those elements in the bottom of the DSM. Once an element is rearranged, it is removed from the DSM (with all its corresponding marks) and step 2 is repeated on the remaining elements.
3. If after steps 1 and 2 there are no remaining elements in the DSM, then the matrix is completely partitioned; otherwise, the remaining elements contain information circuits (at least one).
4. Determine the circuits by one of the following methods:
 - Path searching: information flow is traced either backwards or forwards until a task is encountered twice (Sargent and Westerberg, 1964; Steward, 1981). All tasks between the first and second occurrence of the task constitute a loop of information flow.
 - Powers of the Adjacency Matrix Method: Raising the DSM to the n -th power shows which element can be reached from itself in n steps by observing a non-zero entry for that task along the diagonal of the matrix (Warfield, 1973).
5. Collapse the elements involved in a single circuit into one representative element and go to step 1.

DSM Tearing

Tearing is the process of choosing the set of feedback marks that if removed from the matrix (and then the matrix is re-partitioned) will render the matrix lower triangular. The marks that we remove from the matrix are called "tears".

Identifying those "tears" that result in a lower triangular matrix means that we have identified the set of assumptions that need to be made in order to start design process iterations when coupled tasks are encountered in the process. Having made these assumptions, no additional estimates need to be made.

No optimal method exist for tearing, but we recommend the use of two criteria when making tearing decisions:

- Minimal number of tears: the motivation behind this criterion is that tears represent an approximation or an initial guess to be used; we would rather reduce the number of these guesses used.
- Confine tears to the smallest blocks along the diagonal: the motivation behind this criterion is that if there are to be iterations within iterations (i.e. blocks within blocks), these inner iterations are done more often. Therefore, it is desirable to confine the inner iterations to a small number of tasks.

DSM Banding

Banding is the addition of alternating light and dark bands to a DSM to show independent (i.e. parallel or concurrent) activities (or system elements) (Grose, 1994). The collection of bands or levels within a DSM constitute the critical path of the system/project. Furthermore, one element/activity within each band is the critical/bottleneck activity. Thus, fewer bands are preferred since they improve the concurrency of the system/project. For example, in the DSM shown below (fig. 6), tasks 4 and 5 do not depend on each other for information; therefore, they belong to the same band. Note that in the banding procedure, as described above, feedback marks are not considered (i.e. they are ignored in the process of determining the bands).

Fig. 6: Banded DSM

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1														
2	X								X					
3		X		X										
4	X		X											
5	X		X			X		X				X	X	
6	X				X									
7	X					X								
8						X						X		
9	X		X	X				X						
10				X		X	X	X			X			
11						X	X	X		X				
12	X					X	X			X	X			
13	X				X							X		
14	X	X	X	X	X	X	X	X	X	X	X	X	X	

Numerical DSM

In binary DSM notation (where the matrix is populated with "ones" & "zeros" or "X" marks & empty cells) a single attribute was used to convey relationships between different system elements; namely, the "existence" attribute which signifies the existence or absence of a dependency between the different elements.

Compared to binary DSMs, Numerical DSMs (NDSM) could contain a multitude of attributes that provide more detailed information on the relationships between the different system elements. An improved description/capture of these relationships provide a better understanding of the system and allows for the development of more complex and practical partitioning and tearing algorithms.

As an example, consider the case where task B depends on information from task A. However, if this information is predictable or have little impact on task B, then the information dependency could be eliminated. Binary DSMs lack the richness of such an argument.

What attributes/measures can be used?

- Levels Numbers: Steward (1981) suggested the use of level numbers instead of a simple "X" mark, for certain marks in the binary matrix. Level numbers reflect the order in which the

feedback marks should be torn. The mark with the highest level number will be torn first and the matrix is reordered (i.e. partitioned) again. This process is repeated until all feedback marks disappear. Level numbers range from 1 to 9 depending on the engineers judgment of where a good estimate, for a missing information piece, can be made.

- Importance Ratings: A simple verbal scale can be constructed to differentiate between different important levels of the "X" marks. As an example, we can define a 3-level: 1= high dependency, 2= medium dependency, and 3= low dependency. In the above scenario, we can proceed with tearing the low dependency marks first and then the medium and high in a process similar to the level numbers method, above.

Some other attributes depend on the type of DSM used in the representation and analysis of the problem. For example, in an Activity-based DSM, the following measures can be used:

- Dependency Strength: This can be a measure between 0 and 1, where 1 represents an extremely strong dependency. The matrix can, now, be partitioned by minimizing the sum of the dependency strengths above the diagonal.
- Volume of Information Transferred: An actual measure of the volume of the information exchanged (measured in bits) may be utilized in the DSM. Partitioning of such a DSM would require a minimization of the cumulative volume of the feedback information.
- Variability of Information Exchanged: A variability measure can be devised to reflect the uncertainty in the information exchanged between tasks. This measure can be the statistical variance of outputs for that task accumulated from previous executions of the task (or a similar one). However, if we lack such historical data, a subjective measure can be devised to reflect this variability in task outputs (Yassine et al., 1999).
- Probability of Repetition: This number reflects the probability of one activity causing rework in another. Upper-diagonal elements represent the probability of having to loop back (i.e. iteration) to earlier (upstream) activities after a downstream activity was performed (Smith et al., 1997). While lower-diagonal elements can represent the probability of a second-order rework following an iteration (Browning, 1998). Partitioning algorithms can be devised to order the tasks in this DSM such that the probability of iteration or the project duration is minimized. Browning (1998) devised a simulation algorithm to perform such a task.
- Impact strength: This can be visualized as the fraction of the original work that has to be repeated should iteration occur [Browning (1998) and Carrascosa et al. (1998)]. This measure is usually utilized in conjunction with the probability of repetition measure, above, to simulate the effect of iterations on project duration.

Task-Based DSM Simulation

We can use DSM models to obtain schedule and cost distributions for a specific task execution sequence (i.e. task order) (Browning and Eppinger, 2002). Cost and duration (and their variances) are largely a function of the number of iterations required in the process execution and the scope or impact of those iterations. Since iterations may or may not occur (depending on a variety of variables), this model treats iterations stochastically, with a probability of occurrence depending on the particular package of information triggering rework.

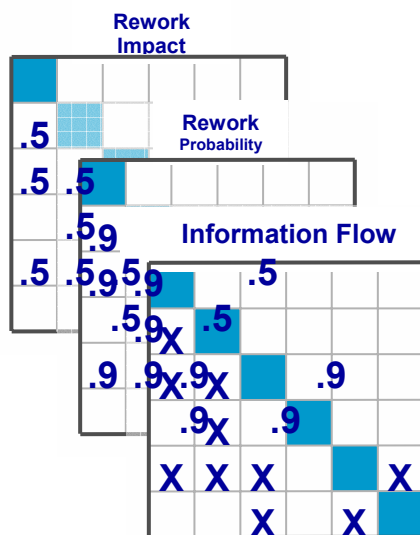
The DSM simulation model characterizes the design process as being composed of activities that depend on each other for information. Changes in that information cause rework. As discussed earlier, rework in one activity can cause a chain reaction through supposedly finished and in-progress activities. Activity rework is a function of rework risk, which combines the probability of a change in inputs and the impact this change might have on the dependent activities. The impact measure used in the simulation represents the fraction of the original work that needs to be repeated

or reworked. Both probabilities and impacts are numbers between 0 and 1 and they replace the marks in the binary DSM. Thus, the simulation model is represented by two DSMs, one containing probabilities of information change and the other containing the impacts of change. In the probability DSM, the superdiagonal marks are replaced by first-order rework probabilities, and the lower-diagonal marks are replaced by second-order rework probabilities. (fig 7).

The simulation also accounts for stochastic activity duration (and cost) by using three point estimates (optimistic (O), most likely (M), and pessimistic (P)) to form a triangular distribution for each activity's cost and duration. For each simulation run, the model randomly samples a single value for each activity's cost and duration from that activity's triangular distribution. Each activity also has an associated improvement curve, which represents learning, set up times, etc. The learning curve (LC) is given by a percentage, the percentage of the original duration required to regenerate the outputs (e.g., it takes 30% of the original duration to repeat the activity second and successive times.) (fig 7).

The model also uses three other vectors, each with a length equal to the number of activities. First, a sequencing vector specifies the order of the activities in the DSM—i.e., the process configuration. Changing the sequencing vector allows for the exploration of alternative process configurations. Second, a work vector, W , keeps track of the amount of work remaining to be done on each activity. Usually, each entry in this vector is set to 100% to begin each simulation run. Third, a “work now” vector of Boolean entries indicates if an activity is to do work at a given time during the simulation execution. This vector is discussed below.

Fig 7: Required input for simulation



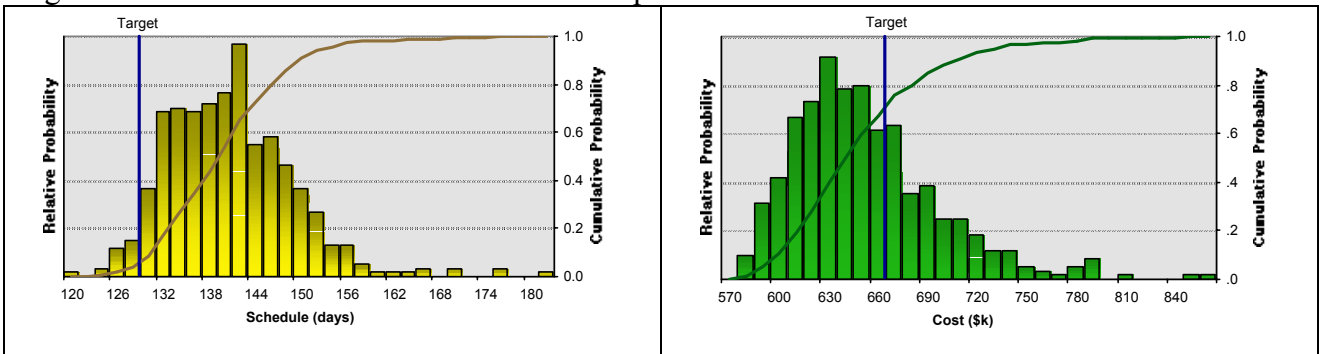
Activities	Durations (days)			Costs (\$k)			LC
Name	O	M	P	O	M	P	
Activity 1	1.9	2	3	8.6	9	13.5	35%
Activity 2	4.75	5	8.75	5.3	5.63	9.84	20%
.....
.....
Activity N	9	10	12.5	6.8	7.5	9.38	33%

The simulation uses a simple, time advancing approach. Each run consists of a series of equal time steps, Δt , the size of which is smaller than the duration of the shortest activity (e.g., if activities have durations ranging from five to 50 weeks, a reasonable Δt could be 0.5 weeks, as activity durations are rounded off to an integer number of time steps. Smaller time steps provide greater model resolution at the expense of greater simulation execution time.)

During each time step, the model checks for the most upstream activity requiring work and any activities that can be executed concurrently. Activities do not begin work until their needed inputs are available from completed, upstream activities. Work is done on available activities during the time step, and their work remaining is reduced by the fraction of their duration represented by the time step. The cumulative process cost is increased based on the cost of this work. Whenever an activity finishes, the model checks for potential iterations (rework for upstream activities) and second-order rework resulting from iterations using the probabilities in the DSM. If rework occurs, its amount—a percentage of the activity given in the DSM and modified by improvement curve effects—is added to the W vector. When all activities are complete, all W vector entries equal zero. The model converts the number of time steps required into appropriate units and outputs this as the process duration or schedule, S , for the run.

Running the simulation provides the sample output shown in figures 8 (Browning and Eppinger, 2002). Figure 8 shows probability mass functions (PMFs) and cumulative distribution functions (CDFs) for a large set of schedule and cost outcomes. The mean duration, $E[S]$, is 141 days with a standard deviation, SD_S , of 8 days. The mean cost outcome, $E[C]$, is \$647k with a standard deviation, SD_C , of \$45k.

Fig. 8: Schedule and Cost PMFs and CDFs Output from Model



Team-Based DSM and Component-Based DSM

Although this article is focused on activity-based DSM, some insights on team-based DSM could be interesting, in that it is a useful tool to analyze organizational structures in projects/product development processes. In fact, the information flow management among activities and among groups should be just one integrated design step.

Team-based DSM is used for organizational analysis and design based on information flow among various organizational entities. Individuals and groups participating in a project are the elements being analyzed (rows and columns in the matrix). A Team-based DSM is constructed by identifying the required communication flows and representing them as connections between organizational entities in the matrix. For the modeling exercise it is important to specify what is meant by information flow among teams. The table (fig 9) presents several possible ways information flow can be characterized.

Fig. 9: Characterization of Information Flows

Flow Type	Possible Metrics
Level of Detail	Sparse (documents, e-mail) to rich (models, face-to-face)
Frequency	Low (batch, on-time) to high (on-line, real)
Direction	one-way to two-way
Timing	Early (preliminary, incomplete, partial) to late (final)

The matrix can be manipulated in order to obtain clusters of highly interacting teams and individuals while attempting to minimize inter-cluster interactions. The obtained groupings represent a useful framework for organizational design by focusing on the predicted communication needs of different players. For an application example, refer to McCord and Eppinger (1993). They proposed a team-based DSM to analyze the organizational structure necessary for an improved automobile engine development process.

Similar to the modeling and analysis of organizations using a DSM approach, we can also model a system or product using component-based DSMs. These DSMs allow the modeling and analysis of system / product architectures by defining the interactions between the subsystems and components comprising the system / product. A qualification scheme for the interactions within a component-based DSM can assist in this analysis. Figure 10 represent the classification of interactions proposed by Pimmler and Eppinger [1994].

Fig. 10: Simple Taxonomy of System Element Interactions

Interaction	Description
Spatial (S)	Identifies needs for adjacency between two elements. Associations of physical space and alignment
Energy (E)	Identifies needs for energy exchange between two elements
Information (I)	Needs for data or signal exchange between two elements
Material (M)	Needs for material exchange between two elements

As an example, consider the following component-based DSM for an automobile Climate Control System (Pimmler and Eppinger, 1994) (fig. 11). The DSM was rearranged such that the following module structure was apparent (fig. 12). Clustering the marks along the diagonal of the DSM resulted in the creation of three "chunks" for the Climate Control System: Front End Air Chunk, Refrigerant Chunk, and Interior Air Chunk (Pimmler and Eppinger, 1994).

Fig. 11

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Radiator	A	2 0 0 2			2 -2 0 0											
Engine Fan	B	2 0 0 2			2 0 0 2								1 0 0 0			
Heater Core	C			1 0 0 0			2 0 0 0	-1 0 0 0								0 0 0 2
Heater Hoses	D		1 0 0 0						-1 0 0 0							
Condenser	E	2 -2 0 0	2 0 0 2			0 2 0 2		-2 2 0 2								
Compressor	F				0 2 0 2			0 2 0 2	1 0 0 2	0 0 2 0	0 0 2 0		1 0 0 0			
Evaporator Case	G		2 0 0 0					2 0 0 0						2 0 0 0	2 0 0 0	2 0 0 2
Evaporator Core	H		-1 0 0 0		-2 2 0 2	0 2 0 2	2 0 0 0		1 0 0 2							0 0 0 2
Accumulator	I			-1 0 0 0		1 0 0 2		1 0 0 2		1 0 0 0						
Refrigeration Controls	J					0 0 2 0			1 0 0 0		0 0 2 0		1 0 0 0			
EATC Controls	K					0 0 2 0				0 0 2 0		0 0 2 0	1 0 0 0	0 0 2 0	0 0 2 0	
Sensors	L										0 0 2 0		1 0 0 0			
Command Distribution	M		1 0 0 0			1 0 0 0				1 0 0 0	1 0 0 0	1 0 0 0		1 0 0 0	1 0 0 0	1 0 0 0
Actuators	N						2 0 0 0				0 0 2 0		1 0 0 0			
Blower Controller	O						2 0 0 0				0 0 2 0		1 0 0 0			2 0 0 2
Blower Motor	P		0 0 0 2				2 0 0 2	0 0 0 2					1 0 0 0		2 0 0 2	

Key:
S E
I M

Fig. 12

	K	J	L	D	M	A	B	E	F	I	H	C	P	O	G	N
Air Controls	K	0 0 2 0	0 0 2 0	0 0 2 0		1 0 0 0			0 0 2 0					0 0 2 0		0 0 2 0
Refrigeration Controls	J	0 0 2 0			1 0 0 0				0 0 2 0	1 0 0 0						
Sensors	L	0 0 2 0			1 0 0 0				2 0 0 0							
Heater Hoses	D									-1 0 0 0		1 0 0 0				
Command Distribution	M	1 0 0 0	1 0 0 0	1 0 0 0			1 0 0 0		1 0 0 0				1 0 0 0	1 0 0 0		1 0 0 0
Radiator	A						2 0 0 2	2 -2 0 0								
Engine Fan	B				1 0 0 0	2 0 0 2		2 0 0 2								
Condenser	E					2 -2 0 0	2 0 0 2		0 2 0 2	-2 2 0 2						
Compressor	F	0 0 2 0	0 0 2 0		1 0 0 0			0 2 0 2		1 0 0 2	0 2 0 2					
Accumulator	I		1 0 0 0		-1 0 0 0			1 0 0 2		1 0 0 2						
Evaporator Core	H							-2 2 0 2	0 2 0 2	2 1 0 2		-1 0 0 0	0 0 0 2		2 0 0 0	
Heater Core	C				1 0 0 0							-1 0 0 0	0 0 0 2		2 0 0 0	
Blower Motor	P				1 0 0 0							0 0 0 2	0 0 0 2	2 0 0 2	2 0 0 0	
Blower Controller	O	0 0 2 0			1 0 0 0							2 0 0 2		2 0 0 0		
Evaporator Case	G											2 0 0 0	2 0 0 0	2 0 0 0	2 0 0 0	
Actuators	N	0 0 2 0			1 0 0 0										2 0 0 0	

Clustering Overview

In partitioning, we have seen that the main objective was to move the feedback marks from the above the diagonal to below the diagonal, given that the DSM elements were tasks to be executed. However, when the DSM elements are people in charge of these tasks or are sub-systems and components of a larger system, then we have a different objective for arranging the DSM. The new goal becomes finding subsets of DSM elements (i.e., clusters or modules) that are mutually

exclusive or minimally interacting. This process is referred to as clustering. In other words, clusters contain most, if not all, of the interactions (i.e., DSM marks) internally and the interactions or links between separate clusters is eliminated or minimized (Fernandez, 1998; Sharman and Yassine, 2003; Yu et al., 2003). In which case, the blocks become analogous to team formations or independent modules of a system (i.e., product architecture). Furthermore, in this setting, marks below the diagonal are synonymous to marks above the diagonal and they represent interactions between the teams or interfaces between the modules.

As an example, consider the DSM in Figure 13. The entries in the matrix represent the frequency and/or intensity of communication (i.e., how frequently teams need to work with other teams: daily, weekly, or monthly) exchanged between the different development participants, represented by person A, person B, ...etc.

Fig. 13: Original DSM

	A	B	C	D	E	F	G
A		•			•	•	
B				•			•
C		•		•			•
D		•	•		•		•
E				•		•	
F	•				•		
G		•	•	•			

As can be seen in Figure 14a, the original DSM was rearranged to contain most of the interactions within two separate blocks: EF and EDBCG. However, three interactions are still outside any block (i.e., team) and constitute the points of interaction/collaboration between the two teams. An alternative team arrangement is suggested in Figure 14b. This arrangement suggests the forming of two overlapping teams (i.e., AFE and EDBCG) where one person (i.e., E) belongs to both teams and may act as a liaison person. The proper decision about how the elements of the product are assigned to chunks is affected by several technical and non-technical factors.

Fig. 14: Clustered DSM

	A	F	E	D	B	C	G
A		•	•				
F	•		•				
E		•		•			
D			•		•	•	•
B				•			•
C				•	•		•
G				•	•	•	

(a) Clustered DSM

	A	F	E	D	B	C	G
A		•	•				
F	•		•				
E		•		•			
D			•		•	•	•
B				•			•
C				•	•		•
G				•	•	•	

(b) Alternative Clustering

Several computational clustering techniques that search for optimal solutions based on tradeoffs between the importance of capturing intra block dependencies versus inter block dependencies may be able to arrive to optimal solutions under certain assumptions (Fernandez, 1998, Yu et al., 2003).

Summary and Concluding Remarks

The DSM method supports a major need in engineering design management: documenting information that is exchanged. The method provides visually powerful means for capturing, communicating, and organizing engineering design activities and architectural issues such as project team formation and product architecture.

In this article, I have provided an introduction to the DSM method as an alternative approach to classical project management techniques for managing complex PD projects. The method is useful by simply building and inspecting the DSM and even without further analysis, building a DSM model of a project/system, improves visibility and understanding of project/system complexity. With the help of a DSM model you can easily convey the process to others in a single snapshot.

We have demonstrated the power of DSMs using few real world applications, but the literature is full of successful DSM implementations from various fields including automotive (McCord and Eppinger, 1993; Yassine et al., 2000), aerospace (Grose, 1994; Browning and Eppinger, 2002), semiconductor (Osborne, 1993), construction (Austin et al., 2000), and telecom (Eppinger, 2002), to name a few.

Many of the DSM tools and algorithms presented in this article are implemented in several commercial and research software. A major resource is the DSM website (www.DSMweb.org) that includes lots of examples, cases, tutorial, references, and computer macros to perform partitioning, tearing, clustering, and simulation. It also contains links to other DSM researchers and research institutions, in addition to commercial DSM software.

References

- Austin, S., Baldwin, A., Li, B., and Waskett, P., "Application of the Analytical Design Planning Technique to Construction Project Management," *Project management Journal*, Vol. 31, pp. 48-59, 2000.
- Browning, T. & Eppinger, S., "Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development," *IEEE Trans. on Eng. Mangt*, 49(4): 428-442, 2002.
- Browning, T. Applying the Design Structure Matrix to System Decomposition and Integration problems: A Review and New Directions. *IEEE Transactions on Engineering management*, Vol. 48, No. 3, August 2001.
- Browning, Tyson R., "Use of Dependency Structure Matrices for Product Development Cycle Time Reduction", *Proceedings of the Fifth ISPE International Conference on Concurrent Engineering: Research and Applications*, Tokyo, Japan, July 15-17, 1998c, pp. 89-96
- Carrascosa, Maria, Eppinger, Steven D. and Whitney, Daniel E., "Using the Design Structure Matrix to Estimate Product Development Time", *Proceedings of the ASME Design Engineering Technical Conferences (Design Automation Conference)*, Atlanta, GA, Sept. 13-16, 1998.
- Eppinger, S. Innovation at the Speed of Information. *Harvard Business Review*, January 2001.
- Eppinger, S., "Innovation at the Speed of Innovation," *Harvard Business Review*, Vol. 79, pp. 149-158, 2001.
- Eppinger, S.D., Whitney, D.E., Smith, R.P., and Gebala, D.A. A Model-Based Method for Organizing Tasks in Product Development. *Research in Engineering Design*, Vol. 6, pp. 1-13, 1994.
- Fernandez, C.I.G. Integration Analysis of Product Architecture to Support Effective Team Co-Location. Master's Thesis (ME), Massachusetts Institute of Technology, Cambridge, MA, 1998.
- Grose, David L., "Reengineering the Aircraft Design Process", *Proceedings of the Fifth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, Sept. 7-9, 1994.
- McCord, K.R. and Eppinger, S.D. Managing the Integration Problem in Concurrent Engineering. M.I.T. Sloan School of Management, Cambridge, MA, Working Paper no.3594, 1993.
- McCord, Kent R. and Eppinger, Steven D., "Managing the Integration Problem in Concurrent Engineering", M.I.T. Sloan School of Management, Cambridge, MA, Working Paper no.3594,, 1993.
- Osborne, S., "Product Development Cysle Time Chracterization through Modeling of Process Iteration," MS Thesis, MIT, Cambridge, MA., 1993.
- Pimmler, T.U. and Eppinger, S.D. Integration Analysis of Product Decompositions. In *Proceedings of the ASME Sixth International Conference on Design Theory and Methodology*, Minneapolis, MN, Sept., 1994.
- Pimmler, Thomas U. and Eppinger, Steven D., "Integration Analysis of Product Decompositions", *Proceedings of the ASME Sixth International Conference on Design Theory and Methodology*, Minneapolis, MN, Sept., 1994. Also, M.I.T. Sloan School of Management, Cambridge, MA, Working Paper no. 3690-94-MS, May 1994.
- Sargent, R. W. and Westerberg, A. W., "Speed-up in Chemical Engineering Design," *Transactions of the Institute of Chemical Engineers*, Vol. 42, 1964.
- Sharman, D. and Yassine, A., "Characterizing Complex Product Architectures," *Systems Engineering*, Forthcoming, 2004.
- Smith, Robert and Eppinger, Steven, "Identifying Controlling Features of Engineering Design Iteration, *Management Science*, vol. 43, no. 3, pp. 276-293, March 1997.
- Steward, D.V. The Design Structure System: A Method for Managing the Design of Complex Systems. *IEEE Transactions on Engineering Management*, Vol. 28, pp. 71-74, 1981.
- Steward, Donald, "The Design Structure System: A Method for Managing the Design of Complex Systems" *IEEE Transactions on Engineering Management*, vol. 28, pp. 71-74, 1981.

- Strang, G., Linear Algebra and its Applications, Second Edition, Harcourt Brace Javonovich, New York, 1980.
- Ulrich, K.T. and Eppinger, Steven D., Product Design and Development, McGraw-Hill, New York, 1995 (1st edition) and 2000 (2nd edition).
- Warfield, John N., "Binary Matrices in System Modeling" IEEE Transactions on Systems, Man, and Cybernetics, vol. 3, pp. 441-449, 1973.
- Yassine, A., Falkenburg, D., and Chelst, K. Engineering Design Management: An Information Structure Approach. International Journal of Production Research, Vol. 37, no. 19, pp. 2957-2975, 1999.
- Yassine, Ali A. and Falkenburg, Donald R, A Framework for Design Process Specifications Management," Journal of Engineering Design, vol. 10, no. 3, September 1999.
- Yassine, A., Whitney, D., Lavine, J. and Zambito, T., "DO-IT-RIGHT-FIRST-TIME (DRFT) Approach to Design Structure Matrix (DSM) Restructuring", Proceedings of the 12th International Conference on Design Theory and Methodology (DTM 2000), September 10-13, 2000 Baltimore, Maryland, USA.
- Yu, T., Yassine, A., and Goldberg, D., "A Genetic Algorithm for Developing Modular Product Architectures," Proceedings of the ASME 2003 International Design Engineering Technical Conferences, 15th International Conference on Design Theory & Methodology, September 2-6, 2003. Chicago, Illinois.