

Engineering design management: an information structure approach

A. YASSINE^{†*}, D. FALKENBURG[‡] and K. CHELST[‡]

It is characteristic of engineering design that precedence relationships among the constituent design tasks contain information flow conflicts. The existence of these conflicts with the lack of formal methods to manage them render the development cycle time unpredictable. This paper discusses a qualitative approach to engineering design management from an information structure perspective. The objective is to model, analyse and manage the interactions manifested by the information exchanges within the design process. We introduce the notion of Structural Sensitivity Analysis (SSA), which is devised based on two measures of information dependency among design tasks: Sensitivity and Variability. Those two measures of dependency enhance the classical design structure matrix method and allow for more complex analysis to be performed.

1. Introduction

The difficulties in designing complex engineering products do not arise simply from their technical complexity. The managerial complexity, necessary to manage the interactions between the different engineering disciplines, imposes additional challenges on the design process. The engineering complexity of a design is simplified by decomposing the design process into smaller engineering tasks and the assignment of these tasks to individuals (Kusiak and Park 1990, Steward 1991). The managerial complexity of a design is contained by using project management tools that model the interfaces and dependencies among the decomposed design tasks. Managing the design process includes four major steps.

- Model the information and dependency structure of the design process.
- Provide a design plan showing the order of execution for the design tasks.
- Reduce the risk and magnitude of iteration between design tasks.
- Explore opportunities for reducing the project cycle time.

The first step in managing the design process is building a structural model that captures all the constituent design tasks and the dependency structure among them. This model will be a design plan showing the order in which the design tasks are performed (i.e. an execution sequence). A major problem in design process management is the existence of information cycles in the design plan.

Cycles represent conflicts in the flow of information within the design process. That is, there is no obvious order in which the design tasks can be performed. In

Revision received May 1998.

[†] Center for Technology, Policy and Industrial Development, Massachusetts Institute of Technology, Cambridge, MA 02139-4307, USA.

[‡] Industrial and Manufacturing Engineering Department, Wayne State University, Detroit, MI 48202, USA.

*To whom correspondence should be addressed.

order to overcome this difficulty, structural analysis is used to determine an execution order for the design tasks by making a decision on what tasks proceed first; thus, breaking the information cycle. This process is called tearing an information cycle. Tearing is analogous to making a guess on an unavailable, but required, design parameter. Several design iterations might be necessary in order to refine the guessed value of the design parameter. Consequently, the number and magnitude of such an iteration process is greatly affected by the choice of a structural model and the tearing criterion utilized.

A structural model is insufficient in some design situations, because it overlooks some important characteristics of the design process. Additional information might be required to provide better criteria for tearing. In this paper, we propose an enhanced structural model based on the Design Structure Matrix (DSM) model developed by Steward (1981, 1991). We utilize two measures to describe the strength of dependency between tasks: Sensitivity and Variability. This bi-variate structural model provides a better reflection of the design structure and allows us to develop an improved tearing procedure. For example, a simple structural model shows that task B depends on information from task A. However, if this information is predictable (i.e. variability measure) or has small influence on task B (i.e. sensitivity measure), then the information dependency could be eliminated, allowing for concurrent execution of both tasks.

The rest of the paper proceeds as follows. In the next section, we survey existing models in design project management; in particular, the Design Structure Matrix (DSM) model. Section 3 discusses in depth the existent structural analysis techniques of a design process based on the DSM representation; i.e. partitioning and tearing algorithms of a DSM. In section 4, we motivate the need for a more comprehensive analysis approach than the design structure system, and introduce the concept of Structural Sensitivity Analysis (SSA). Furthermore, we extend the classical DSM representation by the incorporation of SSA into the matrix, resulting in a Numerical Design Structure Matrix (NDSM). Then, we present a new tearing method based on the NDSM. In section 5, we discuss the managerial implications and ramifications of tearing on the management of the design process. These implications are demonstrated by a simple example in section 6. Finally, section 7 presents our concluding remarks and potential extensions to this research.

2. Design process modelling and management tools

Once decomposed, the design process can be described as an interconnected network of design tasks or a directed graph (Lawler 1976). A directed graph (or simply a digraph) is most popular for representing the precedence relationships among tasks of a project. It consists of a set of nodes, representing the design tasks, and a set of directed lines connecting these nodes. The directed lines or linkages reflect a dependency or a relationship between the connected tasks. This description of the design process is called the information structure, or simply structure, of the design process (Steward 1981).

The Project Evaluation and Review Technique (PERT) method (Spinner 1989) is a digraph representation of a project where the nodes are arranged along a time line. In the PERT method, three probabilistic time estimates are given to each task, reflecting the uncertainty in the duration of tasks. The critical path method (CPM) (Spinner 1989) is a variation of the PERT method. However, the time of any task could be compressed by expending more resources. Thus, CPM assumes a

time-cost tradeoff rather than probabilistic times used in PERT. The precedence diagramming method (PDM) (Spinner 1989) places the tasks on the nodes of the digraph (Activity-on-Node representation), rather than the arrows (Activity-on-Arrow representation). PDM networks allow the digraph to be drawn to scale, which permits the user to see visually the times when activities are scheduled to occur.

Although all of the above network techniques incorporate more information than the digraph on which they are based on, they require that there is only one-way progression along paths, ignoring iterations and feedback loops that are a characteristic of engineering designs. Furthermore, these techniques assume that all steps in a process (number of tasks and their duration) are predictable, which is not necessarily true for engineering design, especially in the conceptual stages. Finally, these methods improve the process flow only by crashing (using more resources) the critical activities, and they do not consider concurrency and overlapping. A more comprehensive technique is the Structured Analysis and Design Technique (SADT) which is a graph-based technique that captures more details than the other graph-based techniques by representing some of the intra-task complexity (Ross 1977). The US Air Force standard IDEF (ICAM DEEinition or Integration DEEinition) project definition method was developed from SADT to perform modelling activities in support of Computer Integrated Manufacturing (CIM) and Concurrent Engineering (CE) (Mayer *et al.* 1993). The IDEF3 is the process description capture method (Belhe and Kusiak 1995). It is used to describe the process flow of a system by defining a sequence of activities and the relationships between them. Just like all other graph-based representations, the IDEF3 charts suffer from practical size limitations. They tend to grow rapidly for a large number of tasks where visual inspection of the information structure becomes very complex and misleading.

A more compact representation of a project is the Design Structure Matrix (DSM) introduced by Steward (1981, 1991). The DSM associated with a digraph is a binary square 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. If there exists an edge from node i to node j , then the value of element ij (column i , row j) is unity (or marked with an X). Otherwise, the value of the element is zero (or left empty). The matrix representation of a digraph provides a systematic mapping among design tasks that is clear and easy to read regardless of size. It can be shown that an empty row represents a node without input, and that an empty column represents a node without output (Marimont 1959). Off-diagonal 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 one interprets the task ordering in the matrix as the execution 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 or information link) and indicate that an upstream task is dependent on a downstream task.

In this paper, we choose to use the DSM representation of the design process for three reasons. First, it overcomes the size and visual complexity of all graph-based techniques. Second, matrices are amenable to computer manipulation and storage. Finally, the DSM representation was used and proved, by several researchers

(Eppinger *et al.* 1990, Steward 1991, Kusiak and Wang 1993), to be useful in concurrent engineering management and implementation.

3. Information structure analysis

We have identified four types of design tasks in a design structure matrix: independent; dependent; interdependent; and input. Independent tasks are those that do not need any input or information from any other task. Dependent tasks are those that require input from other tasks, but not themselves. Interdependent tasks are those that need inputs from other tasks including themselves; i.e. a cycle in the information flow exists. Input tasks provide no information to any other tasks (i.e. have no output). Independent and input tasks can be easily identified in a DSM by the presence of empty rows or columns, respectively. If all the tasks in the DSM are of the dependent type, then the flow of information in the system is exclusively in the forward direction and the order of execution of the tasks is easily determined by simply manipulating the matrix into a lower triangular matrix. The existence of interdependent tasks complicates the design process; precisely, which task to do first. In this case, the DSM cannot be manipulated into a lower triangular form.

The structural analysis of the design process starts with building a structural model using a DSM. The DSM is then sorted and tasks are rearranged in an attempt to eliminate feedback marks. Then, the DSM is partitioned into blocks containing task subsets involved in a cyclic information flow. Finally, feedback marks are torn from the DSM to break the information cycles. A summary of these steps along with a brief description of goals and values is outlined in table 1.

3.1. Building the DSM

The first step in the structural analysis of a design process is building the DSM. An exhaustive list of all the tasks that collectively define the whole design process (for a specific product or project) is established by a group of experts or representatives from all functional areas of an organization. After the team agrees on this list,

Step	Process	Goal	Value
1	Build the DSM	To build a matrix representation of the design process	Identify/organize task sequences and realtionships in a compact form
2	Sort the DSM	To achieve a sequence for the tasks with no feedback information flows	Provide smoother information flow where all requisite information for a task is available before the task
3	Cycle detection	To identify the existence of cyclic information flows	Recognize the existence of iteration in the design process.
4	Partitioning of the DSM	To contain task subsets (that are involved in the same information cycle) in a block around the diagonal of the matrix	Identify iteration task subsets to focus on rather than the whole DSM
5	Tearing of marks in the DSM	To break cycles and achieve a relative task sequence with no cyclic information flows	Reduce the likelihood and magnitude of iteration within the design process and identify a starting point for iteration of tasks

Table 1. Structural analysis of the design process.

we make up the DSM by asking the engineer responsible for each task for the minimum set of other tasks (taken from the list) that need to be performed before his/her task can start (Steward 1991). These tasks are his/her predecessors and are marked in the DSM by an X mark.

3.2. *Sorting the DSM*

There are many procedures and algorithms to order a set of tasks. Steward (1981, 1991) suggests that after finding the source of the feedback problem (by reading the row and column index of the feedback element), swap the two columns and rows. This process is repeated until no more feedback elements exist. Horowitz and Sahni (1983) devised an algorithm known as the topological sorting algorithm, but it assumes that adjacency lists represent the network rather than a matrix form. Kingston (1990) presented another sorting algorithm that works by deleting, from the digraph, a node with no input and the corresponding edges leading out of it. Lawler (1976) developed a simple and efficient procedure for finding a topological order of the tasks using the matrix form when no cycles exist. Lawler's algorithm starts by finding the in-degree of task $i(I_i)$, which is the row sum of that task. Then, he ranks the task with zero row-sum, if it exists, to be the first task in the DSM. This task with all its corresponding marks is deleted from the DSM and the above process is repeated to find another task with zero row-sum. If there exist no tasks with zero row-sum and the DSM is not empty, then the design process contains cyclic flows of information and the procedure is terminated. A listing of Lawler's algorithm is shown in the Appendix as procedure 1.

3.3. *Partitioning the DSM*

The existence of interdependent tasks result in the termination of procedure 1 without finding a partial order for the design tasks. No partial order of the tasks will render the matrix lower triangular. Instead, we seek a block triangular form. A block is the largest subset of a digraph in which every node in this subset has a path to every other node in the subset. Partitioning analysis is used to identify the tasks in a loop and clusters them in a block along the diagonal of the DSM such that all the predecessors of a block appear somewhere before that block (Steward 1991).

All partitioning algorithms, cited in the literature, are similar. The major difference lies in how they identify the cycles. Once a cycle is identified, the group of tasks involved in the cycle is collapsed into one representative node and the search for the rest of the cycles continues in a similar fashion. A generic partitioning procedure is described in the Appendix as procedure 2.

Two popular methods exist to identify a cycle: path searching (Tiernan 1970, Weinblatt 1972, Steward 1981) and powers of the DSM (Kehat and Shacham 1973, Deo 1974). In path searching, the information flow is traced backward until a node is encountered twice. All the tasks visited within this journey constitute an information cycle. Using the powers method, the binary DSM is raised to the n -th power to indicate which tasks can be traced back to themselves in n steps; thus, constituting an information cycle.

3.4. *Tearing the DSM*

Tearing, the next step after partitioning, is concerned with the order of design tasks within each block. The goal is to re-sequence tasks within the blocks of coupled tasks to find an initial ordering to start the iteration process. In other words, tearing

is the process of eliminating one or more feedback marks within a block such that the rearrangement of the tasks within the block renders the block lower triangular. Note that a reordering of the tasks within a block will affect which marks occur above the diagonal, and consequently which marks should be torn. Tearing a mark corresponds to making a guess or assumption about the initial value of a variable, which must be corrected by iteration around the block.

No optimal method exists for tearing (Kehat and Shacham 1973), but researchers (Weinblatt 1972, Steward 1981, Rogers 1989, Kusiak and Wang 1993) have identified two important criteria for tearing procedures.

- 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 performed more often. Therefore, it is desirable to confine the inner iterations to a small number of tasks.

Steward (1981) maps the circuits of a DSM into shunt diagrams, which reveal the tears that are most effective in breaking circuits. The Steward method of tearing is not suitable for large systems, because the construction of shunt diagrams becomes complex and their analysis is even more difficult. Rogers (1989) used simple heuristics to choose a set of tears and he implemented them using a knowledge-based software package. Kusiak and Wang (1993) uses Weinblatt's (1972) algorithm to identify all edge disjoint cycles in a block, and then calculates the frequency of occurrence of each edge. The edge with the highest frequency is torn and the matrix is then repartitioned to check whether another cycle still exists. More recently, Eppinger *et al.* (1990) used a numerical DSM to incorporate the strength of dependency among tasks. The measures used are repeat probabilities reflecting the likelihood of repeating the task if it proceeds without a particular required input.

4. Structural sensitivity analysis

The inadequacy of purely structural models in containing important non-structural information about the design process motivated several researchers to pursue extensions of the DSM. A richer DSM allows for the development of more practical and efficient tearing algorithms.

Steward (1981) discussed the use of numbers instead of marks in the DSM to represent the difficulty level of using an estimate. Smith and Eppinger (1990) extended the basic representation of a DSM to accommodate numerical values that reflected the difficulty of performing tasks in the absence of predecessor information. Krishnan *et al.* (1991) introduced the notion of a quality loss function to capture the decrease in quality of task results due to constraints imposed by a certain sequence of tasks. Sobieszczanski-Sobieski (1988) devised an algorithm that calculates the system sensitivity derivatives from a set of equations derived from the implicit function theorem. In a more recent paper, Smith and Eppinger (1997) used methods of feedback control theory to analyse and identify controlling features of the iteration process. The method called for the determination of the eigenvalues and eigenvectors for the Work Transformation Matrix (WTM) which is a DSM containing the strength of dependency between the tasks.

All of the above research extended the classical representation of a DSM by replacing the X mark in the matrix by a single dependency measure. The major disadvantage of all these methods is that they consider only one dependency measure in the DSM. These representations ignore the complete/full dependency structure profile of the design process that can be better captured through the utilization of multiple attributes, as suggested by our proposed extension to the DSM. Utilizing a two-dimensional variable in the DSM is a new idea and it provides a better reflection of the dependency strength between design tasks.

4.1. *The sensitivity and variability attributes*

In this section, we define two basic measures to clarify the strength of coupling and dependency between two tasks; i.e. sensitivity and variability. The two measures will, then, be converted into a single dependency strength measure and used in the DSM analysis.

The 'sensitivity' of a task to one of its predecessors expresses how sensitive a task is to changes in the output of its predecessor. If a task is very sensitive to its predecessor information, then a small change in predecessor information has a huge impact on its final results. On the other hand, if a task is insensitive (or weakly sensitive) to predecessor design changes, then predecessor information has to change drastically before the dependent task is affected by the change.

'Variability' or 'uncertainty' is the possible deviation of an estimate, at the time of assessment, from the actual value. In this paper, we are concerned with the variability of the predecessor task and the impact of this variability on the successor task. A task exhibits a high variability if the team working on the successor task is incapable of guessing a value, or range of values, for the output of the predecessor task. On the other hand, a task is said to exhibit low variability if the team working on the successor task can always provide a good estimate on the output value of its predecessor.

It is difficult, if not impossible, to come up with a universal objective measurement scale for sensitivity and variability in a design context. We, therefore, construct a discrete, subjective measurement scale for these two attributes. For further details on constructed attributes see Keeney (1992). The sensitivity attribute is made up of verbal descriptions of four different levels of dependency. A numerical indicator is assigned to each dependency level. For example, a sensitivity value of 0 (i.e. weak) means that the information produced by the predecessor task is of no interest or importance to the successor task. If the sensitivity value is 3, this means that the predecessor information is very critical to the successful completion of the successor task, and that the successor task cannot proceed without final and correct predecessor information. Table 2 summarizes the constructed attribute for sensitivity. A

Attribute level	Description of attribute level
0	<i>Weak</i> : the information produced by the precedent task is practically irrelevant (trivial information).
1	<i>Not vital</i> : a major part of the task can be performed without information from this predecessor (verification information).
2	<i>Vital</i> : the task can be started without complete/finalized information from its predecessor, but preliminary/partial information is necessary.
3	<i>Extremely vital</i> : it is absolutely impossible for the task to proceed without complete/finalized predecessor information

Table 2. A constructed attribute for sensitivity.

Attribute level	Description of attribute level
0	<i>Definite</i> : a relatively certain outcome will result.
1	<i>Confident</i> : an outcome can be identified as highly probably (90% sure)
2	<i>Uncertain</i> : an interval of values can be identified, but there is now way to conclude which value within is more likely.
3	<i>Extremely uncertain</i> : it is not possible to identify any limits on the outcome.

Table 3. A constructed attribute for variability.

different set of attribute levels and descriptions can be tailored for a specific design situation.

Similarly, a scale for variability is devised to measure how good an estimate can be made. The constructed levels of this attribute are shown in table 3 along with their verbal descriptions. The table shows that if the actual value of a task output is available at the time its successor is executed, then variability does not exist and the attribute value is zero. Furthermore, if the outcome of a predecessor is highly predictable (even though it is not certain) then we say that the variability is small (i.e. confident outcome).

Once we have devised an instrument to measure sensitivity and variability, the next logical question becomes how do we assign sensitivity and variability values to information links. Experts, in a specific design situation, assign these values through a structured interview process. A discussion of the different subjective assessment techniques is outside the scope of this paper. However, the interested reader is referred to Merkhofer (1987), or Shephard and Kirkwood (1994) for an extensive discussion on subjective assessments of uncertain quantities utilizing a structured expert interview process.

4.2. *The sensitivity–variability matrix—NDSM*

The sensitivity–variability matrix is a two-dimensional Numerical DSM (NDSM), where the X mark in any cell is replaced by the values of sensitivity and variability for that link. The diagonal elements are always equal to zero. As mentioned earlier, the sensitivity and variability values for each mark in the matrix are created by expert assessments through a structured interview process.

The two-dimensional NDSM is converted into a single dimension NDSM in order to utilize the tearing method discussed in the next section. This single measure is referred to as the dependency strength of an information link or DSM mark. A multiplicative utility function (Keeney and Raiffa 1993) is utilized in the conversion. Multiplication of both attributes to create a single dependency measure is an appropriate method, because the two measures exhibit complementarity (Keeney and Raiffa 1993). That is, the overall dependency strength of an information link is neutralized if the link scores high on one scale and low on the other scale. For example, if a task is extremely sensitive to its predecessor information (i.e. a sensitivity value of 3) but the predecessor output is certain (i.e. a variability value of 0), then the overall dependency strength of the link is weak (i.e. an overall product value of 0). In this case, the dependent task can be started based on a good guess on the predecessor information. On the other hand, required information with a large impact (i.e. a sensitivity value of 3) and large variability (i.e. a variability value of

2 or 3) implies a strong link (i.e. an overall product value of 6 or 9). Therefore, the resultant matrix will be populated with elements between 0 and 9 (i.e. 0, 1, 2, 3, 4, 6 and 9), where 0 denotes a non-existent or trivial relationship between the two tasks considered, and 9 denotes an extremely important or binding relationship. The values between 0 and 9 describe a whole range of different dependency strength levels.

4.3. Tearing the NDSM

Beginning with a partitioned DSM as our base model, we start substituting each X mark in the DSM with the associated sensitivity and variability attribute values. These values are subjectively assessed by expert interviews. Then, the two values are converted into a single dependency number by multiplying them together resulting in a one-dimensional NDSM.

The main goal of tearing is to break the information cycle in each block (i.e. task subsets involved in a cycle) and establish a feasible starting execution sequence for this subset of tasks. The tearing criterion used in the proposed algorithm is simple. Tasks that are least dependent on, but deliver maximum input to the rest of the tasks within the cycle are scheduled first in the block (Kehat and Shacham 1973). In order to schedule the tasks based on the above criterion, an index P_i is devised that captures the input to output ratio for each task. The relative position of a task within a block is determined by its P_i value.

We now look at tearing each block separately. For each block in the partitioned NDSM, calculate the block in-degrees (BI_i) and the block out-degrees (BO_i) for all the tasks within that block. Note that BI_i and BO_i are the row and column sums of task i , respectively; however, considering only the subset of tasks and marks contained within the block. Next, we calculate the ratio $P_i = BI_i/BO_i$, which is a relative importance index. The task with the minimum P_i value is scheduled first within the block, since it requires minimum input and delivers maximum output. This is equivalent to tearing all the links, within the block, into that task. The scheduled task and all its corresponding marks are removed from the block. Next, we check if the cycle is broken, due to the removal of the scheduled task, using procedure 1. If an information cycle is encountered again within the block, the process of finding new P_i values is repeated. After ranking all the tasks within a block, we tear all the feedback marks in the block. This procedure is listed in the Appendix as procedure 3.

As an example, consider the block, in figure 4, consisting of tasks 3, 4 and 5. The associated BI_i , BO_i and P_i values are shown in table 4. The table shows that task 3 has the minimum P_i value; therefore, it was scheduled first within the block (i.e. rank = 1). Once task 3 is ranked, we remove it from the block along with its corresponding marks. The removal of task 3 results in breaking the cycle within the block. This was evident by finding a zero block in-degree for task 4; thus, scheduling it second within the block. Finally, task 5 is scheduled last in the block.

Task	BI_i	BO_i	P_i	Rank
3	3	9	0.33	1
4	9	9	1	-
5	9	3	3	-

Table 4. Sample calculations for tearing.

5. Managerial implications of the tearing process

When a mark is torn from the DSM, a decision was made on what tasks to work on first without requisite information. This decision was necessary to establish a practical design plan (i.e. task execution sequence), but it did not change the reality about the information dependency structure between the tasks. The penalty associated with such a decision is reflected in the risk of subsequent task iterations. Consequently, managers of design projects are faced with the extra challenge of reducing or managing the risk of design iterations after establishing a design plan. This added step is necessary because it transforms the tearing algorithm from a mechanical process into a practical managerial tool that helps managers better understand and manage the risk associated with tearing a feedback mark. In addition, this tool establishes useful guidelines that help in the creation of a structured and disciplined transformation from a traditional, sequential design into a concurrent one.

Managerial actions are taken based on the dependency strength of a torn feedback mark. As this value increases from 0 to 9, the risk level increases accordingly and a suitable risk management strategy is utilized to cope with that risk level. For example, if the dependency strength is equal to 0, 1 or 2, then the feedback mark is torn without any further action. This is referred to as artificial de-coupling (Eppinger *et al.* 1990). However, if the dependency strength is 3 or 4, then overlapping (Krishnan *et al.* 1995, 1997) and task redefinition (Eppinger *et al.* 1990) are recommended as risk management strategies. Finally, multi-functional or concurrent engineering teaming strategies (Bursic 1992) are most convenient when the dependency strength is 6 or 9. These different risk management strategies are summarized in table 5.

5.1. Artificial de-coupling

Analysis of the NDSM may show that the dependency strength of a feedback link between two interdependent tasks is weak because of minimal variability in one task or weak sensitivity of one task on the other. In this case, the weak link may be removed from the process structure to break the interdependency between the two tasks. The tearing of an information link may result in future repetition (i.e. redesign or subsequent iterations) of one or both tasks because of a false assumption made by one task on missing predecessor information (Smith and Eppinger 1990). If either of the attribute values is zero, there is no risk associated with artificial de-coupling. However, if the dependency strength is strictly less than 3, then the risk of repetition is minimal and the feedback mark may also be torn without any further managerial actions.

Product value	Managerial actions
0, 1 or 2	Artificial de-coupling
3 or 4	Overlapping and/or task redefinition
6 or 9	Task representation

Table 5. Managerial actions.

5.2. Overlapping

Feedback marks that have a dependency strength of 3 or 4 cannot be removed from the NDSM without incurring significant risk of subsequent design iterations. One way of reducing such a risk is through overlapping. Overlapping requires both tasks to share and communicate information more effectively and frequently. A better communication pattern between the two interdependent tasks allows for early detection of design conflicts. The earlier the design team can identify conflicts, the lower the chance of iterations and the smaller the magnitude of subsequent design iterations as compared to sequential execution (Clark and Fujimoto 1991). The sequential and overlapping execution strategies for two interdependent tasks are shown in figures 1 and 2, where A_1 to A_n and B_1 to B_n represent n iterations of tasks A and B, respectively.

In overlapping, we determine what fraction of the predecessor task must be completed before the follower task can begin. For overlapping to be effective, upstream (predecessor) information availability and downstream (successor) information needs must be understood (Krishnan *et al.* 1997). In many design situations, the official release time of design information does not coincide with the time this information is really available (Whitney 1992). Finding when this information is really available and whether it can be released right away to subsequent tasks is critical for overlapping (Krishnan *et al.* 1995). The critical step in overlapping is identifying a point within the predecessor task duration where preliminary (i.e. partial) design information is sufficiently evolved to be utilized by the successor task. If it is not possible to find such a point, then the opportunity for overlapping is lost.

For the first overlapping case (where sensitivity = 3 and variability = 1), we can utilize the preemptive overlapping strategy as described by Krishnan *et al.* (1997). In this strategy, predecessor information can be frozen earlier than its normal (i.e. scheduled) freeze time since its variability is low. This will allow the successor task to start earlier, but with final predecessor information.

For the second overlapping case (where the sensitivity = 1 and variability = 3), we can utilize the iterative overlapping strategy as described by Krishnan *et al.* (1997). Since the variability of predecessor information is high, we cannot freeze it early. However, we can utilize the predecessor information in its preliminary form

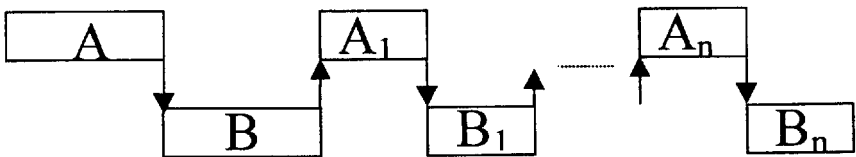


Figure 1. Sequential execution of interdependent tasks.

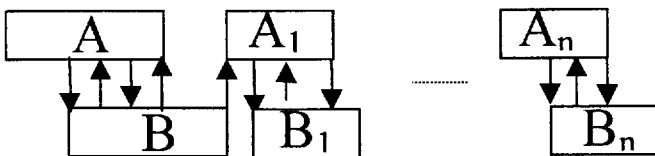


Figure 2. Overlapped execution of interdependent tasks.

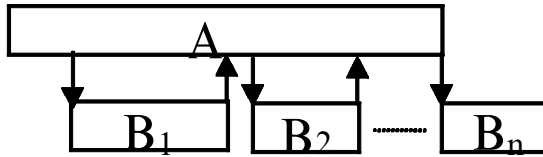


Figure 3. Task redefinition.

without a significant risk of downstream rework, because the downstream sensitivity is weak. In this strategy, the downstream task starts early based on preliminary upstream information, and upstream design changes are incorporated in subsequent downstream iterations.

5.3. Task redefinition

An improved communication policy, as in overlapping, may prove insufficient for reducing the risk level of iterations when the feedback dependency strength is 3 or 4. A more structured approach would be a redefinition of one of the tasks involved to investigate if the redefined task (or tasks) has lower sensitivity and variability values compared to the original task.

Task redefinition strategy, an extension to overlapping, calls for the disaggregation of predecessor design information into two or more information fragments that can be released (as incomplete or partial design information) earlier than the planned one-shot release. Consequently, the follower (i.e. downstream) task is broken into n sub-tasks, where each sub-task utilizes one (upstream) information fragment, as described in figure 3.

There is no general rule for the optimal number of upstream partial information pieces required or the number of downstream task breakdowns. This depends on the specific design situation being analysed and requires a thorough understanding of the design process details. In general, if it were not possible to disaggregate upstream design information, or possible to redefine the downstream task into a subset of smaller tasks, then we would lose the opportunity of improving the design process through task redefinition.

5.4. Multifunctional teams

In design situations where it is too risky to tear a feedback mark due to its high dependency strength (i.e. 6 or 9), we recommend weakening the dependency strength of this mark by representing it upstream through a concurrent engineering or multifunctional team. The basic goal of the teaming strategy is to guarantee that downstream concerns are considered upstream and achieve an instantaneous feedback policy. This will weaken the downstream feedback link by reducing the probability and severity of its occurrence (Bursic 1992).

A successful multifunctional team consists of representatives from all the tasks that are involved within the same information cycle. The result is two-fold: decreased variability of a predecessor due to the fact that upstream task engineer(s) are working closely with downstream task engineer(s); and lower sensitivity value of the successor due to the instantaneous feedback accomplished by the multifunctional team. Once the value of the feedback dependency strength is reduced to a value less

than 6, we can investigate further risk management strategies that are utilized when the product value is between 0 and 4.

6. Example: cylinder block development process

In this section, we describe a simplified, purely sequential process for the design and development of an automotive cylinder block. Furthermore, this example will only consider the activities and interactions within engineering design and casting operations, ignoring all other activities. Consequently, the purpose of the example is not to discuss in detail the development of a cylinder block; however, it is to demonstrate the concepts described earlier in this paper.

Engine block designers receive a 'high-level' description of the engine specifications (i.e. number of cylinders, power and torque requirements, and space available) which they convert into a detailed cylinder block design. Once the block design is detailed and documented, it is passed to the casting department to evaluate its casting feasibility and to extract from it the design of the sand cores. Finally, the designs of the sand cores are passed to a supplier to build the soft casting tools necessary. The first set of soft tools is used to produce tryout castings. After layout and inspection of the prototypes, design changes to the block might occur to rectify any structural or mechanical defects. This task is usually referred to as casting development.

6.1. DSM analysis

The cylinder block development process is depicted by the DSM in figure 4. We could not eliminate the feedback marks from the DSM using simple row and column manipulations, as described in procedure 1. The matrix is naturally divided into three blocks.

- Block 1: one big block consisting of tasks 3–9.
- Block 2: a smaller block, within block 1, consisting of tasks 3–5.

	Activities	1	2	3	4	5	6	7	8	9
1	Engineering Concept	*								
2	Design Concept	3,3	*							
3	Product Design		3,3	*		1,3	1,3			
4	Design Documentation			3,3	*					
5	Design Checking				3,3	*				
6	Core Design					3,3	*			2,2
7	Soft Tooling						3,3	*		
8	Tryout Castings							3,2	*	
9	Casting Development								3,2	*

Figure 4. (Sensitivity, Variability) DSM of the block design process.

- Block 3: a smaller block, within block 1, consisting of tasks 6–9.

The DSM shows that the development process is composed of two main stages: engineering design (block 2) and casting operations (block 3). Blocks 2 and 3 represent internal iterations within each stage. External iterations between engineering design and casting operations are represented by the existence of block 1 which couples blocks 2 and 3 by two information links. The first is the forward link (6, 5), the mark in row 6 and column 5, and the other is the feedback link (3, 6), the mark in row 3 and column 6.

6.2. *NDSM analysis*

Figure 4 presents the NDSM where each mark is labelled by both attributes of sensitivity and variability to assess the strength of the links. These attributes were assessed subjectively by interviewing each design participant responsible for that particular task. The sensitivity DSM shows that all the forward marks (i.e. those below the diagonal) have high values (i.e. 3) for both attributes, suggesting a strong sequential process. All the feedback marks (except one) have a low (i.e. value of 1) sensitivity attribute value, because they merely represent a checking and verification feedback for the task and not a required input into it. On the other hand, their variability attribute values are high (i.e. a value of 3). Multiplying both attributes together resulted in dependency strengths greater than or equal to 3. Therefore, artificial decoupling cannot be used for eliminating these feedback links.

The relative order of the subtasks within blocks 1, 2 and 3 is in agreement with tearing procedure 3. That is, task subsets within each block are listed in ascending P_i values. As a result, procedure 3 deletes marks (3, 6), (6, 9) and (3, 5). Now, the resultant matrix is lower triangular. In the next sub-sections, we investigate the risk management strategies utilized when tearing these marks.

6.3. *Tearing mark (3, 6)*

Since the dependency strength for this link is equal to 3, we can utilize overlapping between the two blocks (i.e. between engineering design and casting operations). Overlapping requires incomplete, partial predecessor information to be used by the successor task. Therefore, the forward link (6, 5) must be analysed.

Link (6, 5) indicates that the transfer of information, from design to casting, occurs in one shot and only when the engineering design is completely finalized. According to section 5, an improvement opportunity over the current information structure is to disaggregate this forward link into several information fragments and examine if parts of the cylinder block design information can be released earlier than the planned one-shot release of complete design information. To do so, more detailed information about the nature of this design process will be required.

Interviews with professionals in casting operations revealed that casting design for the cylinder block is performed in an inside-out manner. That is, the internal cores are designed first and then the exterior cores. This led us to break down the casting design task into three subtasks: inner cores design, side cores design and end cores design, as shown in figure 5.

The reason for designing interior cores first is that they do not interact physically with other engine components, e.g. the cylinder head, which dictates the number, size and location of head bolt bosses. For internal core design, it is enough to know the basic block specifications (i.e. bore size, the amount of cooling water and oil lubrication).

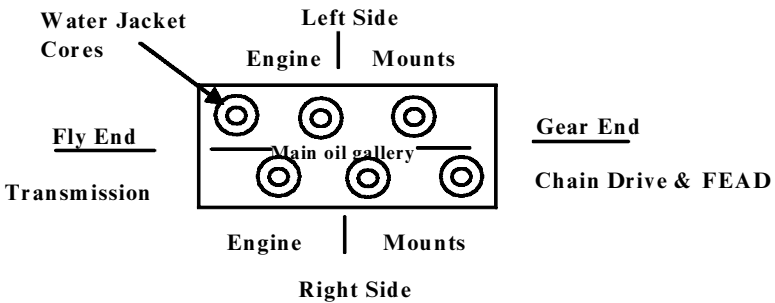


Figure 5. The automotive cylinder block.

tion required), which is preliminary design information that can be supplied to casting operations early in the development process. This will constitute the first batch of partial information transfer between engineering design and casting operations.

The next batch of information is required for the design of oil drain and water jacket cores. For water and oil flows, a high degree of accuracy for their physical size or volumes is not required. However, for the oil drains to be correctly located, they have to match with the oil holes in the cylinder head. Therefore, design information concerning the cylinder head oil holes location is required at this point. This information is usually available almost halfway into the development process and will be communicated to casting operations as the second batch of partial design information transfer. It is noteworthy that the side cores, creating the block left and right side exterior surfaces, can now be designed as an offset to the inner and oil drain cores.

Finally, the end cores, fly and gear ends, have to wait for enough design information to be evolved on the transmission, chain drive and Front End Accessory Drives (FEAD). Such detailed information is usually available at an advanced stage (i.e. towards the end) in the development process, and will constitute the third and last partial information transfer to casting operations.

To summarize the above discussion, casting operations require three types of information from engineering design. These different types of design information are usually available at different times during the development process. Therefore, engineering design information can be fragmented and released, on a timely basis, to casting operations in three partial information pieces.

- (1) Basic block parameters: for inner core design.
- (2) Cylinder head oil holes location: for oil drain cores and side cores design.
- (3) Transmission geometry along with the chain drive design and FEAD requirements: for end cores design.

Finalized engineering design information will also be transmitted to casting at the usual scheduled time. Casting will use the finalized design information to check and verify their initial assumptions made earlier and update the core designs.

6.4. *Tearing mark* (6, 9)

In block 3, the existence of the strong (sensitivity = 2, variability = 2) feedback mark (6, 9), from casting development to core design, suggests that task redefinition

	Activities	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Eng'g Concept	*														
2	Design Concept	x	*													
3	Product Design		x	*		x										
4	Document'n			x	*											
5	Checking				x	*										
6	Inner Cores		x	x			*	x								
7	Side Cores		x	x			x	*	x							
8	End Cores		x	x				x	*							
9	Update Cores					x	x	x	x	*						
10	Soft Tools									x	*					
11	Tryouts										x	*				
12	Casting Develop't											x	*			
13	Fix Tools												x	*		
14	Prototypes													x	*	
15	Verification														x	*

Figure 6. Modified DSM of the block design process.

within casting operations can be helpful in reducing the magnitude of this mark. The tasks were redefined such that we are able to incorporate the effect of the casting development task before the second iteration of the tooling construction. This resulted in the creation of two new tasks: fix tools and tryout castings, as shown in the modified DSM of figure 6.

6.5. *Tearing mark (3, 5)*

In block 2, link (3, 5) has an overall score of 3, which means that overlapping is a suitable strategy for design process improvement. ‘Design checking’ should not be performed in one shot; however, the design checker should work closely with the product design team and share product information on a timely basis. This will require an analysis of the forward links (4, 3) and (5, 4) to investigate the utilization of partial information by the design checker, similar to the situation discussed in section 6.3. In this paper, we will not perform such an analysis for these links.

7. **Conclusion**

The DSM is a compact representation of the information structure of a design process. It is a design plan showing the order in which the design tasks are performed, where assumptions should be made, and what tasks need to be verified. Existing partitioning and tearing algorithms are helpful in organizing the design activities within the DSM to increase the efficiency of the design process and reduce the product development time. However, improved DSM representations are necessary to develop more efficient and practical tearing algorithms.

In this paper, we enriched the classical DSM representation by including measures of sensitivity and variability into the matrix resulting in a NDSM. The NDSM allowed us to develop an improved tearing algorithm. Analysis of the NDSM pro-

vided a structured transformation approach from a sequential design process to a concurrent engineering practice. Our contributions can be summarized as follows.

- Develop a two-dimensional extension to the classical DSM model to describe the strength of dependency between tasks.
- Demonstrate a strategy and justification of a multiplicative single measure of dependency.
- Develop a new tearing algorithm utilizing the new measure of dependency.
- Emphasize the role of managers in managing the practical risks associated with tearing.

Acknowledgments

The authors would like to thank the editor of the *International Journal of Production Research*, and the two anonymous referees for their invaluable comments and feedback on previous revisions of this manuscript.

Appendix

Procedure 1: Sorting

Step 1. (start: Find row-sum of tasks)

Set $I_i = \sum_{j=1}^n a_{ij}$, $i = 1, \dots, n$.

Set $N = \{1, 2, \dots, n\}$

Set $m = 1$

Step 2. (detection of node with 0 in-degree)

Find $k \in N$ such that $I_k = 0$. If there is no such k , stop; the digraph contains cycles.

Set Rank(k) = m

Set $m = m + 1$

Set $N = N - k$

If $N = \emptyset$, stop; the computation is completed.

Step 3. (Revision of in-degrees)

Set $I_i = I_i - a_{ik}$ for all $i \in N$

Return to step 2.

Procedure 2: Partitioning

Step 1. Determine all the circuits that exist in the DSM using either path searching or powers of the matrix.

Step 2. Collapse all tasks within the same circuit into a single representative task.

Step 3. Order the remaining tasks using procedure 1. If a cycle is detected, then go to step 1; otherwise, the procedure is complete.

Procedure 3: Tearing

Note that the subset of tasks within a specific block is referred to set B .

Step 0. Set $n = 1$. (n is the rank of a task within a block.)

Step 1. Calculate $P_i = BI_i / BO_i$ for every task $i \in B$.

Step 2. Find k , element of B , such that P_k is minimum. If more than one task exists, select the task with the minimum number of block in-degrees, then the one with the maximum number of block out-degrees. If a tie still exists, the selection is immaterial.

Step 3. Assign rank n to task k

Set $B = B - k$. If $B = \phi$ go to step 7.

Set $BI_i = BI_i - a_{ik}$ for all $i \in B$.

Set $BO_j = BO_j - a_{kj}$ for all $j \in B$.

Set $n = n + 1$

Step 4. Check if there exists a $k \in B$ such that $BI_k = 0$. If yes, use procedure 1 to find a new sequence. Otherwise, go to step 1.

Step 5. The procedure is terminated. Tear all marks that fall above the diagonal of the block.

References

- BELHE, U. and KUSIAK, A., 1995, Resource constrained scheduling of hierarchically structured design activity networks. *IEEE Transactions on Engineering Management*, **42**, 150–158.
- BURSIC, K., 1992, Strategies and benefits of the successful use of teams in manufacturing organizations. *IEEE Transactions on Engineering Management*, **39**, 277–289.
- CLARK, K. and FUJIMOTO, T., 1991, *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry* (Boston, USA: Harvard Business School Press).
- DEO, N., 1974, *Graph Theory with Applications to Engineering and Computer Science* (Englewood Cliffs, NJ: Prentice Hall).
- EPPINGER, S. D., WHITNEY, D. E., SMITH, R. P. and GEBALA, D. A., 1990, Organizing the tasks in complex design projects. In *Proceedings of the 1990 ASME Conference – Design Theory and Methodology* (New York: ASME), pp. 39–46.
- HOROWITZ, E. and SAHNI, S., 1983, *Fundamentals of Data Structures* (Rockville, MD: Computer Science Press).
- KEENEY, R. and RAIFFA, H., 1993, *Decisions with Multiple Objectives* (Cambridge, MA: Cambridge University Press).
- KEENEY, R., 1992, *Value Focused Thinking* (Cambridge, MA: Harvard University Press).
- KEHAT, E. and SHACHAM, M., 1973, Chemical process simulation programs—2. *Process Technology International*, **18**, 115–118.
- KINGSTON, J., 1990, *Algorithms and Data Structures* (New York: Addison-Wesley).
- KRISHNAN, V., EPPINGER, S. and WHITNEY, D., 1991, Towards a cooperative design methodology: analysis of sequential design strategies. In *Proceedings of the 1991 ASME Conference—Design Theory and Methodology* (New York: ASME).
- KRISHNAN, V., EPPINGER, S. and WHITNEY, D., 1995, Accelerating product development by the exchange of preliminary design information. *Journal of Mechanical Design, Transactions of the ASME*, **117**, 491–498.
- KRISHNAN, V., EPPINGER, S. and WHITNEY, D., 1997, A model-based framework to overlap product development activities. *Management Science*, **43**, 437–451.
- KUSIAK, A. and PARK, K., 1990, Concurrent engineering: decomposition and scheduling of design activities. *International Journal of Production Research*, **28**, 1883–1900.
- KUSIAK, A. and WANG, J., 1993, Efficient organizing of design activities. *International Journal of Production Research*, **31**, 753–769.
- LAWLER, E., 1976, *Combinatorial Optimization: Networks and Matroids* (New York: Holt, Rinehart & Winston).
- MARIMONT, R., 1959, *Journal of Math. Phys.*, **38**, 164.
- MAYER, R., PAINTER, M. and DEWITTE, P., 1993, *IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications* (College Station, TX: Knowledge Based Systems).
- MERKHOFFER, M., 1987, Quantifying judgmental uncertainty: methodology, experiences, and insights. *IEEE Trans. Sys., Man, and Cybern.*, **SMC-17**, 741–752.
- ROGERS, J., 1989, A knowledge-based tool for multilevel decomposition of complex design problem. NASA TP 2903, May.
- ROSS, D., 1977, Structured Analysis (SA): a language for communicating ideas. *IEEE Transactions on Software Engineering*, **SE-3**, 16–34.

- SHEPHARD, G. and KIRKWOOD, C., 1994, Managing the judgmental probability elicitation process: a case study of analyst/manager interaction. *IEEE Trans. on Engineering Management*, **41**, 414-425.
- SMITH, R. and EPPINGER, S., 1990, Modeling design iterations. Working paper no. 3160-90-MS, Sloan School of Management, MIT.
- SMITH, R. and EPPINGER, S., 1997, Identifying controlling features of engineering design iteration. *Management Science*, **43**, 276-293.
- SOBIESZCZANSKI-SOBIESKI, J., 1988, On the sensitivity of complex, internally coupled systems. NASA TM 100573, Langley Research Center, Hampton Virginia.
- SPINNER, M., 1989, *Improving Project Management Skills and Techniques* (Englewood Cliffs, NJ: Prentice Hall).
- STEWARD, D., 1981, *System Analysis and Management: Structure, Strategy and Design* (New York: Petrocelli Books).
- STEWARD, D., 1991, Planning and managing the design of systems. *Proceedings of Portland International Conference on Management of Engineering and Technology*, Portland, Oregon, USA, 27-31 October.
- TIERNAN, J., 1970, An efficient search algorithm to find the elementary circuits of a graph. *Communications of the ACM*, **13**, 722-726.
- WEINBLATT, H., 1972, A new search algorithm for finding the simple cycles of finite directed graphs. *Journal of ACM*, **19**, 43-56.
- WHITNEY, D. E., 1992, Electro-mechanical design in Europe: University Research and Industrial Practice. Technical Report No. CSDL-P-3226, Draper Laboratory, 10 December.