

# The Design Structure Matrix (DSM)

## Introduction to DSM

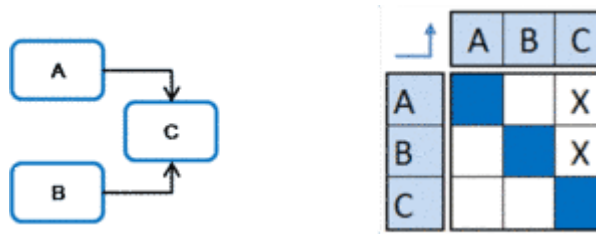
The Design Structure Matrix (DSM – also known as the dependency structure matrix, dependency source matrix, and dependency structure method) is a general method for representing and analyzing system models in a variety of application areas. A DSM is a **square matrix** (i.e., it has an equal number of rows and columns) that shows **relationships between elements in a system**. Since the behavior and value of many systems is largely determined by interactions between its constituent elements, DSMs have become increasingly useful and important in recent years. Relative to other system modeling methods, a DSM has two main advantages:

- It provides a simple and concise way to **represent** a complex system.
- It is amenable to powerful **analyses**, such as clustering (to facilitate modularity) and sequencing (to minimize cost and schedule risk in processes).

The DSM is related to other square-matrix-based methods such as a dependency map, a precedence matrix, a contribution matrix, an adjacency matrix, a reachability matrix, and an N-square diagram, and also related to non-matrix-based methods such as directed graphs, systems of equations, and architecture diagrams and other dependency models.

The use of matrices in system modeling can be traced back to the 1960s, if not earlier. However, it was not until the 1990s that the methods received relatively widespread attention.

Earlier works used graphs for system modeling, although the use of graphs in managing complex structures is generally recognized. For example, consider a system that is composed of three elements (or sub-systems): element “A”, element “B”, and element “C”. DSM works under the assumption that – for the modeling purpose – the three elements completely describe the system and characterize its behavior. A graph may be developed to represent this system pictorially. The system 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.



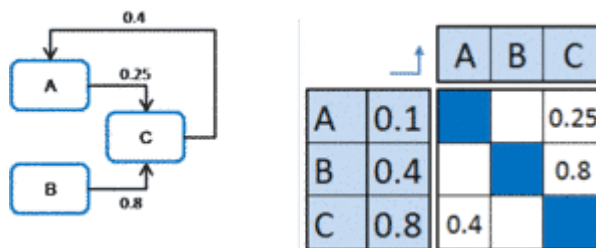
The matrix representation of a digraph (i.e. directed graph) has the following properties:

- it is binary (i.e. a matrix populated with only zeros and ones)
- it is square (i.e. a matrix with equal number of rows and columns)
- it has  $n$  rows and columns ( $n$  is the number of nodes of the digraph)
- it has  $k$  non-zero elements, where ( $k$  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  $ij$  (row  $i$ , column  $j$ ) is unity (or marked with an X). 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, although many find it intuitive to think of these diagonal cells as representative of the nodes themselves.

Binary matrices are useful in systems modeling because they can represent the presence or absence of a relationship between pairs of elements in a system. A major advantage of the matrix representation over the digraph is that its compactness and ability to provide a systematic mapping among system elements allows for a detailed analysis of a limited set of elements in the context of the overall structure. As such, it therefore provides a qualitative way of dependency modeling in a formal approach.

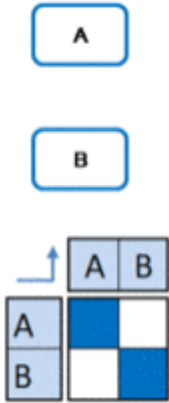
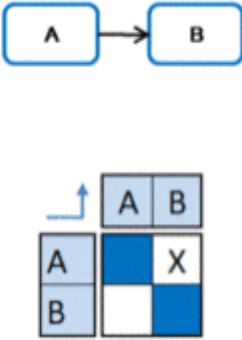
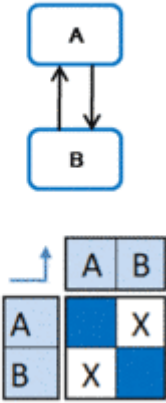
A matrix can also represent weighted dependencies. In such a case, a numerical (<https://dsmweborg.wordpress.com/numerical-dsms/>). DSM is used. Equally, an additional column can be used to represent the weight of an element.



If the system is a project represented by a set of tasks to be performed, then off-diagonal marks in a single column of the DSM represent all of the tasks whose output is required to perform the task corresponding to that column (i.e., read a column to see a task's inputs, which are the outputs of other tasks). Similarly, reading along a specific row reveals which tasks receive information from the task corresponding to that row (i.e., read along a row to see where a task's outputs go to become other tasks' inputs). In many cases, the order of tasks down the matrix corresponds to a timeline. In such a case, marks above the diagonal represent forward information transfer to later (i.e. downstream) tasks. This

kind of mark is called a forward mark or forward information link. Marks below the diagonal depict information fed back to earlier listed tasks (i.e., a feedback mark) and indicate that an upstream task is dependent on a downstream task. Please note that not all DSMs are written this way; while the convention of “row impacts column” prevails in some areas. This second convention is the transpose of the matrix from the first convention. Both conventions convey equivalent information, and anyone familiar with one convention can usually adapt to the other fairly quickly and easily. The DSM literature contains many instances of both conventions. The reason for the existence of two conventions is that square-matrix-based methods sprung from at least two historical roots (discussed below).

There are three basic building blocks for describing the relationship amongst system elements: parallel (or concurrent), sequential (or dependent) and coupled (or interdependent).

Configuration of Relationships	parallel	sequential	coupled
Graph and DSM representation			

In the parallel configuration, the system elements do not interact with each other (at least not for the type of representation that is represented in the digraph). Understanding the behavior of the individual elements allows us to better understand the behavior of the system. If the system is a project, then system elements would be e.g. project tasks to be performed, and usually the relations would be the directed information exchange between the tasks. As such, task B is said to be independent of task 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. For example, design parameters of element B are selected based on the design parameters of element A. In terms of project tasks, task A should be performed before task B.

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 “Circuit” or “Cycle”.

A recent book, <http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=12819>) **Design Structure Matrix Methods and Applications** (<https://mitpress.mit.edu/books/design-structure-matrix-methods-and-applications>), provides a comprehensive introduction to DSM.