Indra Gunawan

DSM methods to improve planning and scheduling in asset management

Asset Management and Maintenance Journal, 2012; 25(1):50-55

DSM Methods To Improve Planning And Scheduling In Asset Management

INDRA GUNAWAN School of Applied Sciences and Engineering
Monash University (Gippsland Campus) Australia indra.gunawan@monash.edu

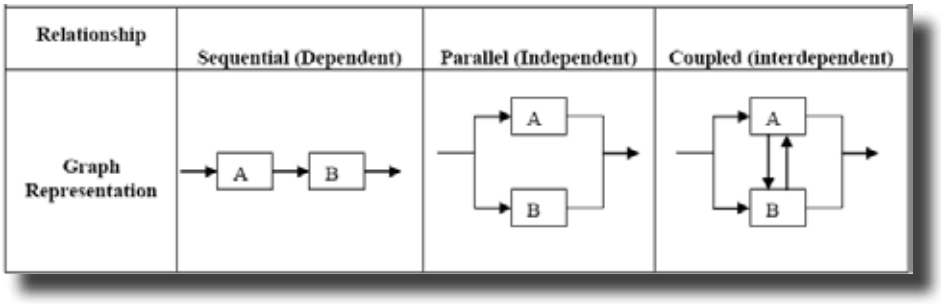
In this paper, the Design Structure Matrix (DSM) method is presented to improve planning and scheduling in asset management. The main advantage of the DSM over traditional project scheduling tools such as Critical Path Method (CPM) or Gantt chart is in compactness and ability to present an organized and efficient mapping among tasks that is clear and easy to read regardless of size. Three DSM methods: Path Searching, Powers of the Adjacency Matrix, and Reachability Matrix methods are discussed. As a case study, DSM methods are implemented to reduce design iteration or rework in an engineering project. New project duration is optimised compare to initial project duration.

Introduction

Design Structure Matrix (DSM) also known as Dependency Structure Matrix, Problem Solving Matrix and Design Precedence Matrix is a project management tool used for representing and analysing dependencies (Eppinger et al., 1994, 2003, Maheswari et al., 2005, M. E. Sosa, 2008, Mori et al., 1999, P. Sage et al., 2009, T. R. Browning, 2009, U. Lindemann et al. 2009, Yassine et al., 1999, 2004). The DSM was developed by Warfield (in the 70's) and Steward (1981). The approach then developed by Massachusetts Institute of Technology (MIT) research in the design process in 1990.

There are three basic types of task dependencies as shown in Figure 1: **Sequential (dependent), Parallel (independent)** and **Coupled (interdependent)**.

Figure 1 Graphical Representation od the ThreeTypes of Task Dependencies



Traditional project management tools such as CPM or Gantt chart normally can manage the first two types of task dependencies. But if the relationship between tasks is coupled, then the DSM approach should be implemented to improve the efficiency of project schedule development for large and complex projects.

In the next sections, the DSM approach is presented. Three DSM methods: Path Searching, Powers of the Adjacency Matrix, and Reachability Matrix methods are introduced. An application of these methods in a petroleum oil field development project is analysed. Finally, the summary of the DSM methods with respect to project scope is discussed.

Design Structure Matrix (DSM)

The DSM is a compressed, matrix representation of a project. The matrix contains a list of all tasks. It shows what information is required to start a certain task and where that information from that task feed into, which other tasks in the matrix use the output information (Banerjee et al., 2007, Browning, 2001, Chen C. et al, 2004, Cho et al., 2001, Danilovic, 2007, Eppinger et al., 2003). In a DSM model, a project task is assigned to a row and a corresponding column. The rows and columns are named and ordered identically. Each task is defined by a row of the matrix. We represent a task's dependencies by placing marks ("x", "o" or "1") in the columns to indicate the other tasks (columns) on which it depends. Reading across a row reveals all the input tasks and reading down a column reveals the output tasks as shown in Figure 2.

Figure 2 A DSM Representation of a Project

	A	B	C	D	E	F
A						X
B	X				X	
C						
D		X				
E			X			
F						

The diagonal tasks of the matrix do not have any interpretation in describing the system, they are either left empty, blacked out, filled in with the task labels or task duration. This is done to separate the upper and lower triangles of the matrix and to show more clearly the tracing dependencies. The marks below the diagonal indicate forward flow of information. For example, task B needs information from task A. The marks above the diagonal indicate a feedback from a later (downstream) task to an earlier (upstream) one. For example, task A needs information from task F.

To fully carry out the DSM approach, we must determine suitable system decomposition by correctly determining the dependency relationships gathered. Then, we must decompose the system into significant system elements (subsystems or modules) by gathering engineering managers' feedback from different areas of an organization and collecting a list of different sub-systems that include the system entirely. Decomposition can be hierarchical or non-hierarchical (also called network decomposition). In hierarchical decomposition, the system can be divided into sub-systems or modules and those modules are divided into finer components. In non-hierarchical a system hierarchy is not obvious. After the system elements/activities have been identified they are listed in order down the rows and across the columns.

DSM Methods

The DSM can be used to improve the planning, execution and management of complex projects using different algorithms, which are partitioning, clustering, tearing, banding, simulation and eigenvalue analysis.

Partitioning (Steward, 1981; Yassine, 2001) is the process of rearranging the order of activities in such a way so that the dependency relationships are brought either close to the diagonal as possible (this form of the matrix is known as block triangular) or below the diagonal, changing the DSM into a lower triangular form. Fewer elements in the system will be involved in the iteration cycle. The outcome is a faster development process. There are many approaches used in DSM partitioning, they are similar but different in how they identify cycles (circuits or loops) of information.

The algorithm for the formation of a partitioned DSM is explained below:

i) Consider an activity DSM.

ii) Observe for any marks along the upper diagonal (feedback/loop/circuits) of the matrix. If there are no marks along the upper diagonal it means that the matrix is partitioned. Stop the procedure or continue with the next step.

iii) Check for empty rows (activities that do not have input from the rest of the activities in the matrix) and move all the empty rows to the top of the matrix and the corresponding columns to the left of the matrix and leave out these activities from further consideration. Empty rows represent the start activities. The remaining activities in the matrix form the active matrix.

iv) From the active matrix, check for any empty columns (deliver no information to other activities in the matrix) and move all these empty columns to the right and the corresponding rows to the bottom of the active matrix and leave out these activities from further consideration. Empty columns represent the finish activities.

v) Repeat steps iii and iv until there are no empty rows and columns in the active matrix. Repeating the above process allows to identify the dependent activities.

Steps i-v are known as the Topological Sorting Algorithm.

vi) Determine circuits/loops by Path Searching, Powers of the Adjacency Matrix, and Reachability Matrix methods.

vii) Merge/condense all the activities in the loop to form a block.

viii) Repeat the final condensed matrix to find the block sequence.

In the Path Searching method, information flow is traced forwards or backwards until a task is come across twice. To trace out the circuits/loops choose a mark in a row, then read up to the column value. Go to the same row value of this column and read up to the column. Continue reading across the rows until that same mark, read from the first column appears again. All the tasks between the first and second occurrence of the task form a loop of information flow. When all loops in the DSM have been identified, and all tasks have been scheduled, the sequencing is complete and the matrix is in block triangular form.

In the Powers of the Adjacency Matrix, the matrix is converted into a binary matrix by replacing the marks in the DSM with "1" and all other empty cells as "0". Raising the DSM to the n-th power shows which task can be traced back to itself in n steps by observing a "1" entry for that task along the diagonal of the matrix. In the resultant square matrix, cells with a value greater than one are replaced by a value of one.

The following steps show the Reachability Matrix algorithm:

1) Convert the diagonal elements (that are either filled in with the task labels, left empty or blacked out) and make them equal to "one" or "x".

2) Create a table with four columns

3) In the first column, we list all the elements in the matrix

4) In the second column, we list the set of all the input elements for each row in the table. To recognize the input elements in the matrix, is indicated by an entry of "x" or "one" to the corresponding row in the DSM. We include the element itself as an input.

5) In the 3rd column, we list the set of all output elements for each row in the table. To recognize the output elements in the matrix, is indicated by an entry of "x" or "one" to the corresponding column in the DSM. We include the element itself as an output.

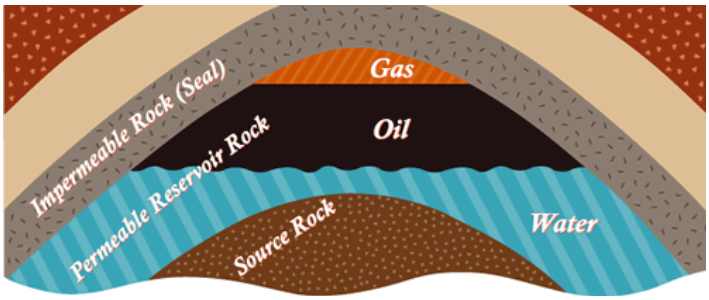
6) In the fourth column, we list the intersection of the input and output sets for each element in our table. We can now identify top level elements and remove them from the table. An element is in the top level hierarchy of the matrix if its input set is equal to the intersection set.

Case Study: Petroleum Oil Field Development Project

(Although not a Maintenance related example it will however illustrate the application of DSM)

The objective of the Petroleum Oil Field Development (POFD) project is to design a development plan for a new oil field discovered after drilling a number of wells. The development plan consists of oil producers, water/gas injectors and surface facility to handle the produced oil, water and gas. Figure 3 shows a cross section of an oil field before development. In this project, three DSM methods will be implemented to improve planning, execution and managing the project by reducing the number of feedbacks.

Figure 3. A cross section of an oil field before development



The project is divided into five areas:

Conduct Reservoir Rock Type (RRT) Study, Build Static Model, Conduct Special Core Analysis (SCAL) Study, Build Dynamic Model, and Conduct Pressure Volume Temperature (PVT) Study. This project is an activity based performed by one team involving: Team leader/manager, Reservoir Engineers, Petroleum Engineers, Geologists, and Petrophysists. Project duration is estimated about 100 days and this project involves 24 tasks as follows:

1.1 Review & Prepare Data
1.2 Collect Samples
1.3 Define Reservoir Rock Types (RRT)
1.4 Prepare Data for Static Model
2.1 Input Data
2.2 Build Reservoir Framework
2.3 Build 3D Property Model (s)
2.4 Manipulate & Rank Models
2.5 Build 3D Flow Simulation Grid (s)
3.1 Study Existing Data Sources
3.2 Conduct Coring
3.3 Conduct Rock Characterization
3.4 Conduct Geo-mechanical Studies
3.5 Conduct Special Core Analysis (dynamic displacement experiments)
3.6 Do Routine & Special Core Interpretation
4.1 Input Data
4.2 Initialize Reservoir Dynamic Model
4.3 Do History Matching
4.4 Do Development Predictions
5.1 Study Existing Data Sources
5.2 Collect Samples
5.3 Conduct Specialized PVT Study
5.5 Develop PVT Applications

Constructing The DSM

We interviewed a reservoir engineer specialist to determine the inputs and outputs for the list of tasks and the task durations (days) involved in the project. We input the marks and the task durations (along the diagonal) into the matrix as shown in Figure 4.

Partitioning The DSM

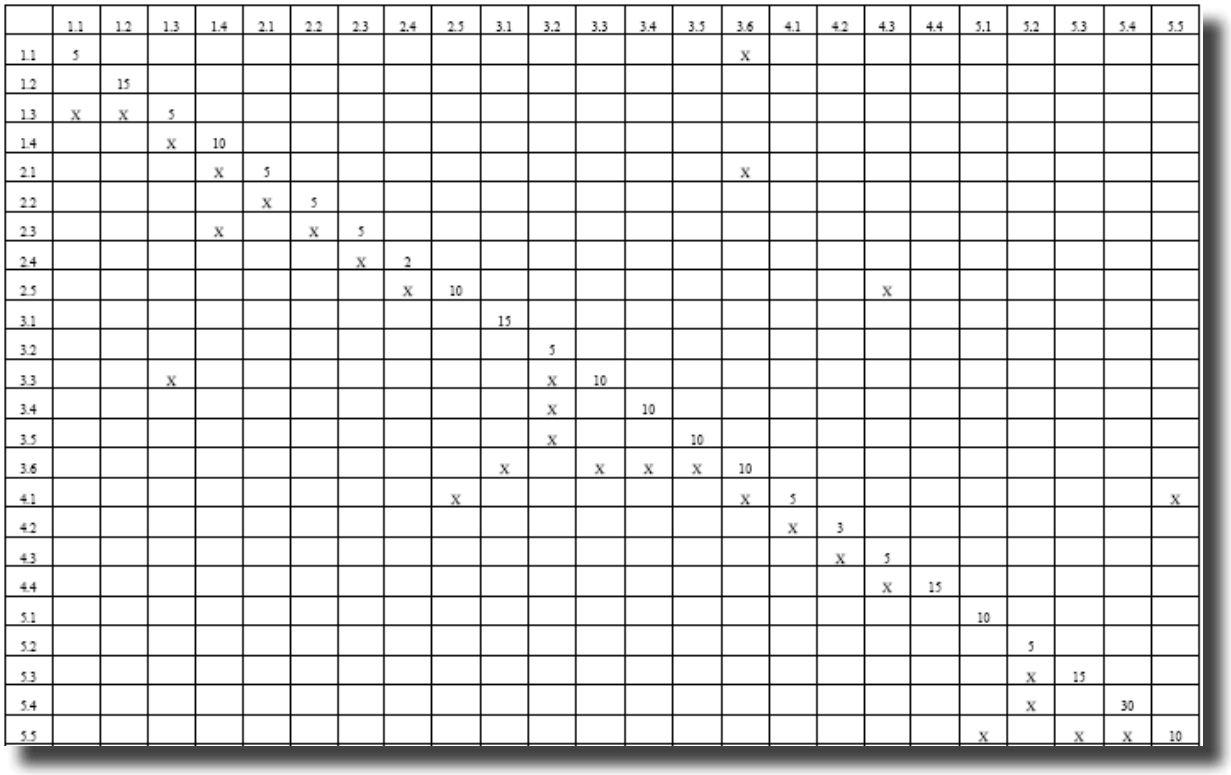
The aim of partitioning the DSM is to maximize the availability of information required, and minimize the amount of iterative loops within the process. The process is ordered to minimize the number of dependencies above the diagonal. Partitioning the matrix sequences the tasks that do not contribute to iterative loops and indicates the tasks that are within iterative loops, but does not sequence the tasks within the loops. This is because the tasks that contribute to a loop are all inter-related, and any of them can be the first task carried out in the completion of the loop. It is desirable that the tasks within a loop are ordered to reduce the number of estimates and iteration within the process. The first step of the process is the topological sorting before we identify loops/circuits using three DSM methods: Path Searching, Powers of the Adjacency Matrix, and Reachability Matrix methods.

Tasks 1.2, 3.1, 3.2, 5.1 and 5.2 do not depend on any information from any other tasks, as shown with empty rows. We can schedule this set of tasks first and leave out from further consideration. Next we will schedule the tasks that depend only on tasks 1.2, 3.1, 3.2, 5.1 and 5.2. Tasks 3.4 and 3.5 depend only on task 3.2; we will schedule these tasks after task 5.2. Tasks 5.3 and 5.4 depend only on task 5.2. We will schedule these tasks after tasks 3.4 and 3.5. Task 5.5 depends on only tasks 5.1, 5.3 and 5.4. We will schedule task 5.5 after task 5.4. Task 4.4 does not deliver information to any other tasks in the matrix, as shown by an empty column. We will move task 4.4 to the right and corresponding row to the bottom of the matrix as shown in Figure 5 and leave out this task from further consideration.

1. Path Searching Method

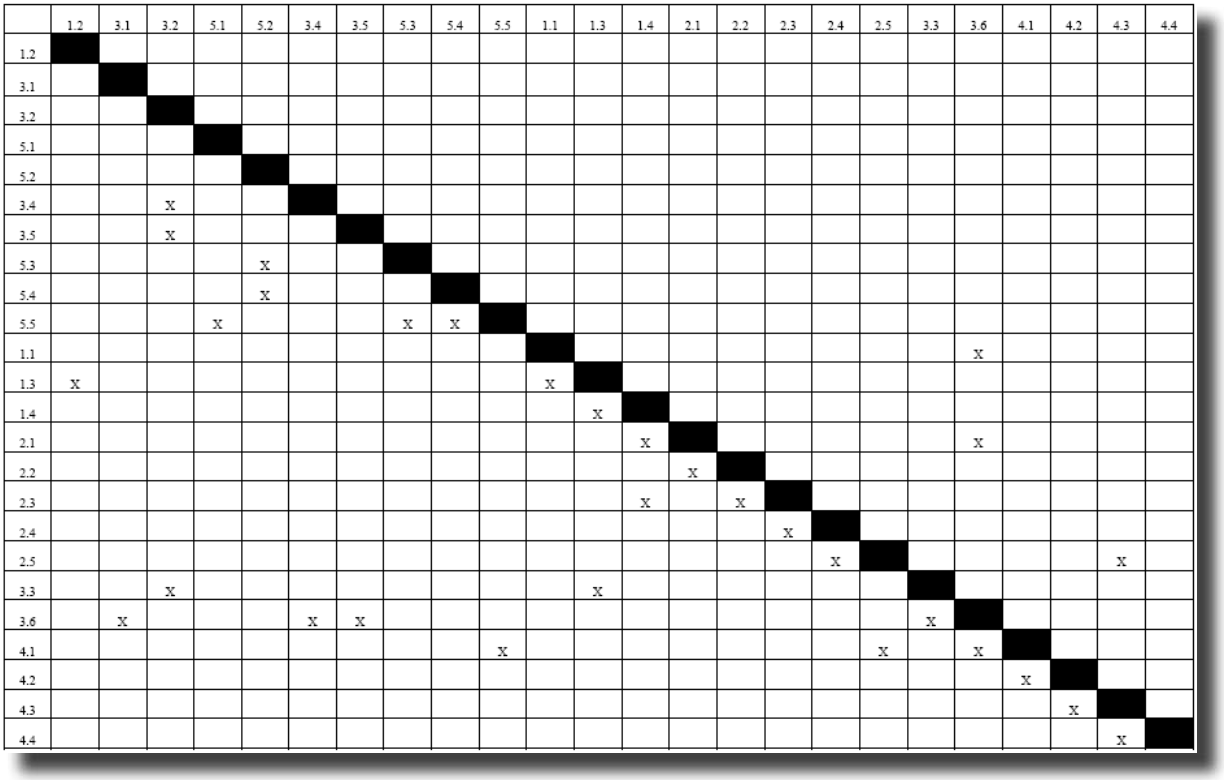
We trace forward starting with the remaining tasks that contain marks above the diagonal. We read across row 1.1, identify a mark and read up to column 3.6. We read across row 3.6, identify a mark and read up to column 3.3 (we ignore the other marks across the row of 3.6 because we have already scheduled those tasks using topological sorting). We read across row 3.3, identify a mark and read up to column 1.3. We read across row 1.3, identify a mark and read up to column 1.1. From this information tasks 1.1, 1.3, 3.3 and 3.6 are involved in a circuit because as we read across the rows, the first row we read (i.e. task 1.1) appeared again as we read across row 1.3, identified a mark and read up to column 1.1.

Figure 4 DSM representation of the Petroleum Oil Field Development Project



	1.1	1.2	1.3	1.4	2.1	2.2	2.3	2.4	2.5	3.1	3.2	3.3	3.4	3.5	3.6	4.1	4.2	4.3	4.4	5.1	5.2	5.3	5.4	5.5
1.1	5														X									
1.2		15																						
1.3	X	X	5																					
1.4			X	10																				
2.1				X	5										X									
2.2					X	5																		
2.3				X		X	5																	
2.4							X	2																
2.5								X	10									X						
3.1										15														
3.2											5													
3.3			X								X	10												
3.4											X		10											
3.5											X			10										
3.6										X		X	X	X	10									
4.1									X						X	5								X
4.2																X	3							
4.3																	X	5						
4.4																		X	15					
5.1																				10				
5.2																					5			
5.3																					X	15		
5.4																					X		30	
5.5																				X		X	X	10

Figure 5 Topological Sorting



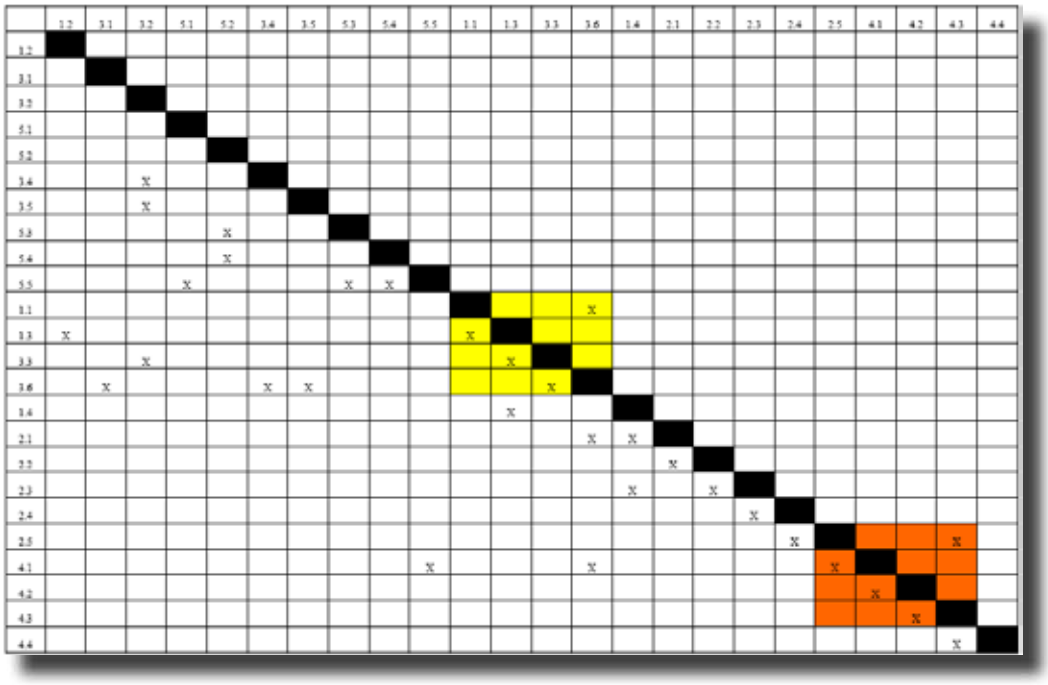
	1.2	3.1	3.2	5.1	5.2	3.4	3.5	5.3	5.4	5.5	1.1	1.3	1.4	2.1	2.2	2.3	2.4	2.5	3.3	3.6	4.1	4.2	4.3	4.4
1.2																								
3.1																								
3.2																								
5.1																								
5.2																								
3.4			X																					
3.5			X																					
5.3					X																			
5.4					X																			
5.5				X				X	X															
1.1																				X				
1.3	X										X													
1.4												X												
2.1													X							X				
2.2														X										
2.3													X		X									
2.4																X								
2.5																	X						X	
3.3			X									X												
3.6		X				X	X												X					
4.1										X								X		X				
4.2																					X			
4.3																						X		
4.4																							X	

We will rearrange the tasks so that the tasks involved in the circuit (i.e. tasks 1.1, 1.3, 3.3 and 3.6) are scheduled after each other in the matrix. We will schedule these tasks after scheduling the first tasks in topological sorting and leave out these tasks from further consideration. These tasks will form a block in the matrix as shown in Figure 6.

We trace forward task 2.5 (because it contains a mark above the diagonal) we read across row 2.5, identify that mark above the diagonal and read up to column 4.3. We read across row 4.3, identify a mark and read up to column 4.2.

We read across row 4.2, identify a mark and read up to column 4.1. We read across row 4.1, identify a mark and read up to column 2.5. From this information tasks 2.5, 4.1, 4.2 and 4.3 are involved in a circuit because as we read across the rows, the first row we read (i.e. task 2.5) appeared again as we read across row 4.1, identified a mark and read up

Figure 6 The Final Partitioned Matrix



to column 2.5. These tasks will form another block in the matrix as shown in Figure 6.

2. Powers of the Adjacency Matrix Method

We convert the active matrix as shown in Figure 5 into a binary matrix by replacing the "X" marks by "1" and all other empty cells with "0". Raising the DSM to the 4th power reveals that tasks 1.1, 1.3, 2.5, 3.3, 3.6, 4.1, 4.2 and 4.3 are involved in a 4 step circuit. This is indicated by a "1" entry for the tasks along the diagonal of the matrix as shown in Figure 7. In the resultant square matrix, cells with a value greater than one are replaced by a value of one.

Raising the matrix to the 5th power reveals that the DSM is not involved in any other circuits indicated by the "0" along the diagonal of the matrix. By using trial and error to determine which tasks are involved in which circuit, we will condense (schedule after each other) all the tasks in each circuit to form two blocks. The first involves tasks 1.1, 1.3, 3.3 and 3.6 and the second block involves tasks 2.5, 4.1, 4.2 and 4.3. We will convert the binary matrix back to the active matrix as shown in Figure 6.

Figure 7 The Binary Matrix Raised to the 4th Power

	1.2	3.1	3.2	5.1	5.2	3.4	3.5	5.3	5.4	5.5	1.1	1.3	1.4	2.1	2.2	2.3	2.4	2.5	3.3	3.6	4.1	4.2	4.3	4.4
1.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1.3	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1.4	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2.1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
2.2	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
2.3	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0
2.4	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0
2.5	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	0	0	0
3.3	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
4.1	1	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0
4.2	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0
4.3	0	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0
4.4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0

3. Reachability Matrix Method

We will construct a table as shown in Table 1 listing the input elements, output elements, the intersection of the input and output sets for each element and the level of hierarchy. We can observe from the column of the input elements that elements 1.2, 3.1, 3.2, 5.1 and 5.2 are in the top level hierarchy because the input values of these elements are equal to the intersection values. We will remove these elements from the table and continue until we reach the corresponding input & output values as shown in Table 2. We will rearrange the original DSM, and schedule the elements starting with the top level hierarchy through to the 11th level hierarchy elements as shown in Figure 6. The circuits we identified will form two blocks in the DSM. The first involves tasks 1.1, 1.3, 3.3, 3.6 and the second block involves tasks 2.5, 4.1, 4.2 and 4.3.

Conclusions

DSM is a new approach to asset management, used to represent, analyse dependencies among tasks and show the order in which tasks are preformed. This approach provides a way of managing feedbacks. The outcome from partitioning the DSM is a faster development process that can be done by optimizing the availability of information. From the case study, we can see that the number of feedbacks in the petroleum oil field development project have been reduced, hence reducing the project cost significantly.

In general, the DSM methods can be applied to identify loops/circuits for project planning and scheduling improvement. When the number of tasks involved in the project is small, the Path Searching method can be done effectively. If the project

is more complex, then the Powers of the Adjacency Matrix and the Reachability Matrix methods should be implemented. It can be observed from the case study that the Reachability Matrix method is also efficient for small projects.

Elements	Input Elements	Output Elements	Intersection	Level
1.1	1.1, 3.6	1.1, 1.3	1.1	
1.2	1.2	1.2, 1.3	1.2	1
1.3	1.1, 1.2, 1.3	1.3, 1.4, 3.3	1.3	
1.4	1.3, 1.4	1.4, 2.1, 2.3	1.4	
2.1	1.4, 2.1, 3.6	2.1, 2.2	2.1	
2.2	2.1, 2.2	2.2, 2.3	2.2	
2.3	1.4, 2.2, 2.3	2.3, 2.4	2.3	
2.4	2.3, 2.4	2.4, 2.5	2.4	
2.5	2.4, 2.5, 4.3	2.5, 4.1	2.5	
3.1	3.1	3.1, 3.6	3.1	1
3.2	3.2	3.2, 3.3, 3.4, 3.5	3.2	1
3.3	1.3, 3.2, 3.3	3.3, 3.6	3.3	
3.4	3.2, 3.4	3.4, 3.6	3.4	
3.5	3.2, 3.5	3.5, 3.6	3.5	
3.6	3.1, 3.3, 3.4, 3.5, 3.6	1.1, 2.1, 3.6, 4.1	3.6	
4.1	2.5, 3.6, 4.1, 5.5	4.1, 4.2	4.1	
4.2	4.1, 4.2	4.2, 4.3	4.2	
4.3	4.2, 4.3	2.5, 4.3, 4.4	4.3	
4.4	4.3, 4.4	4.4	4.4	
5.1	5.1	5.1, 5.5	5.1	1
5.2	5.2	5.2, 5.3, 5.4	5.2	1
5.3	5.2, 5.3	5.3, 5.5	5.3	
5.4	5.2, 5.4	5.4, 5.5	5.4	
5.5	5.1, 5.3, 5.4, 5.5	4.1, 5.5	5.5	

Table 1 Element's inputs, outputs, intersections and level of hierarchy

Elements	Input Elements	Output Elements	Intersection	Level
4.4	4.4	4.4	4.4	11

Table 2 Removing the 10th level hierachy elements

References

[1] Banerjee, A., Carrillo, E. and Paul, A. (2007). Projects with sequential iteration: Models and complexity. IIE Transactions, Vol. 39, No. 5, pp. 453-463.

[2] Browning.T.R. (2001). Applying the Design Structure Matrix to System Decomposition and Integration problems: A Review and New Directions. IEEE Transactions on Engineering Management, pp 292-300.

[3] Chen.C, Khoo.L and Jiao.A. (2004). Information deduction approach through quality function deployment for the quantification of the dependency between design tasks. International Journal of Production Research, Vol 42,pp 4623- 4637.

[4] Cho and Eppinger. (2001). Product development process modeling using advanced simulation. Design engineering technical conferences, Pittsburgh, pp 1-9.

[5] Danilovic, M. and Browning, T. R. (2007). Managing complex product development projects with design structure matrices and domain mapping matrices. International Journal of Project Management, April 2007, Vol. 25 No. 3, pp. 300-314.

[6[Eppinger, Whitney, Smith and Gebala. (1994). A Model-Based Method for Organizing Tasks in Product Development. MIT Sloan School of Management, pp 1-20

[7] Eppinger and Ulrich. (2003). Product Design and Development. New York McGraw-Hill.

[8] Maheswari.J, Varghese.K. (2005). A Structured Approach to Form Dependency Structure Matrix for Construction Projects. International Symposium on Automation and Robotics in Construction, Indian Institute of Technology Madras, pp 1–6.

[9] M. E. Sosa. (2008). A Structured Approach to Predicting and Managing Technical Interactions in Software Development. Research in Engineering Design, Vol. 19, No. 1, pp. 47-70.

[10] Mori.T, Kondo.K, Ishii.K and Ohtomi.K. (1999). Task Planning For Product Development by Strategic Scheduling of Design Reviews. ASME Design Engineering Technical Conferences, Las Vegas, pp 1-12. [11] P. Sage and W. B. Rouse. (2009). Handbook of Systems Engineering and Management. Wiley, New York, 2 edition.

[12] Steward, D. (1981). The Design Structure Matrix: A Method for Managing the Design of Complex Systems. IEEE Transactions on Engineering Management, Vol. 28, No. 3, pp. 71-74.

[13] T. R. Browning. (2009). The Many Views of a Process: Towards a Process Architecture Framework for Product Development Processes. Systems Engineering, Vol. 12, No. 1, pp. 69-90.

[14] U. Lindemann, M. Maurer and T. Braun. (2009). Structural Complexity Management - An Approach for the Field of Product Design. Springer, Berlin.

[15] Yassine.A, Falkenburg.D and Chelst.K. (1999). Engineering design management and information structure approach. International Journal of Production Research, Vol 37 pp 2957 – 2975.

[16] Yassine, Whitney and Zambito. (2001). Assessment of Rework Probabilities for Simulating Product Development using the Design Structure Matrix (DSM). ASME International Design Engineering Technical Conferences, Pennsylvania, pp 1-9.

[17] Yassine. A. (2004). An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) method. Product development research laboratory, University of Illinois, pp 1-17.

Indra Gunawan is a Senior Lecturer and Coordinator of **"Postgraduate Programs in Maintenance & Reliability Engineering"** in the School of Applied Sciences and Engineering at Monash University. He obtained his Ph.D. degree in Industrial Engineering from Northeastern University, USA. His main areas of research are reliability engineering, production and operations management, application of operations research, applied statistics, probability modeling, and project management.