OPERATIONS AND SUPPLY CHAIN MANAGEMENT
Vol. 2, No. 1, January 2009, pp. 1-10
ISSN 1979-3561|EISSN 1979-3871

Application of Numerical Design Structure Matrix Method in Engineering Projects Management

Indra Gunawan
Department of Mechanical and Manufacturing Engineering
Auckland University of Technology, Auckland, New Zealand
E-mail: indra.gunawan@aut.ac.nz

Abstract
In this paper, ways of improving planning, execution and management of projects using Numerical Design Structure Matrix (NDSM) method are presented to address interdependency of feedback and iteration, which is common in engineering projects management. The NDSM is an alternative approach to traditional project management tools such as Program Evaluation and Review Technique (PERT), Critical Path Method (CPM), and Gantt chart that can only allow the modelling of sequential and parallel processes in projects. As a case study, the model is tested on a set of tasks in a complex petroleum oil field development project, where task sensitivity and information variability attributes are derived. By applying the NDSM method, project duration is optimized and hence total cost of the project is reduced significantly.

Keywords: *numerical design structure matrix, partitioning, project management, scheduling, tearing.*

1. Introduction
Since potential savings could be optimized by minimizing the amount of rework in engineering projects management, it is desirable to have management methodologies that consider rework (Austin et al., 2000; Browning, 2001; Chen, 2004; Cho and Eppinger, 2001).

In this paper, a method to account for feedback and iteration, a matrix based tool called the Numerical Design Structure Matrix (NDSM) is presented. The NDSM method is derived from Design Structure Matrix (DSM) which also known as the Dependency Structure Matrix, the Problem Solving Matrix (PSM) and the Design Precedence matrix. The DSM is a project management tool used for representing and analysing dependencies (Banerjee et al., 2007; Eppinger et al., 1994; Eppinger and Ulrich, 2003; Maheswari and Varghese, 2005; Mori et al., 1999). The DSM was developed by Warfield (in the 70's) and Steward (1981). The method received attention by Massachusetts Institute of Technology (MIT) research in the design process in 1990. A Domain Mapping Matrix (DMM) was introduced to compare two DSMs on different project domains (Danilovic et al., 2007).

In the next sections, DSM layout is discussed. Then, the NDSM method that incorporates task sensitivity and information variability attributes is presented. Finally, the NDSM operations (partitioning and tearing) are demonstrated in the petroleum oil field development project.

2. DSM Layout
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 (Eppinger and Ulrich, 2003). In the 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 1.

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.

3. Partitioning the DSM

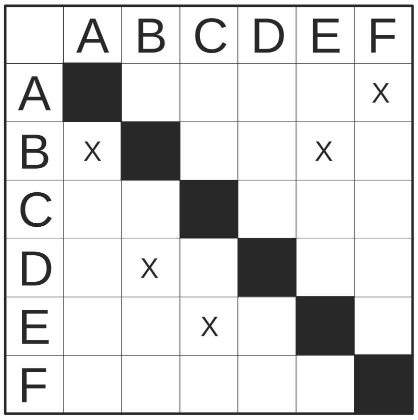
The DSM can be used to improve the planning, execution and management of complex projects using different algorithms, which are partitioning, clustering, tearing, banding, simulation and eigenvalue analysis. Partitioning (Steward, 1981; Yassine et al., 1999, 2001; Yassine, 2004) is the process of rearranging 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 a Path Searching method.
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 (Steward, 1981). To trace out the

Figure 1. A DSM representation of a project



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

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.

4. Tearing the NDSM

The Numerical DSM compared to the DSM could contain a multiple of attributes that provide more detailed information on the relationships between the different system elements. By improving the description/capture of relationships between elements this provides a better understanding of the system and allows for a more accurate estimation in tearing algorithm. For example if task B depends on information from task A. If the information from task A is expected or have little impact on task B, then the information dependency could be eliminated.

Steward (1981) suggested the use of level numbers instead of "X" marks, for certain marks in the matrix. Level numbers reflect the order in which the feedback marks should be torn. The mark with the low dependency is torn and the matrix is reordered (partitioned) again. This process is repeated until all feedback marks disappear. Engineers/managers are interviewed about the information dependency strength between tasks.

Off-diagonal elements in the Numerical Design Structure Matrix are called "task volatility". Task volatility (TV) describes the volatility (instability) of dependent tasks (located in the rows) with respect to changes in information from input tasks (located in the columns). Task volatility implies a probability that the dependent task will be reworked to some degree. This number is located in the matrix at the intersection of the row of the dependent task and column of the input task.

Two measures to simplify the strength of coupling and dependency between two tasks (i.e. Task sensitivity and Information variability) are defined. The two measures are converted into a single dependency strength measure and used in the DSM analysis.

Task sensitivity describes how sensitive the completion of a dependent task is to change or modifications of information from an input task. Each task's sensitivity to changes in information from a particular input task is different. Task sensitivity depends on the level of dependency between two particular tasks. If a task is very sensitive to changes in information from an input task, then a small change from an input task has a huge impact on its final results. On the other hand if a task is insensitive (or weakly sensitive) to changes in information from an input task, it has to change considerably before the dependent task is affected by the change.

The sensitivity attribute is made up of descriptions of four different levels of dependency. A numerical value is allocated to each dependency level as shown in Table 1 (Yassine et al., 1999).

A task sensitivity value of 0 (i.e. weak) means that the information from an input task is of no importance to the successor task. If the sensitivity value is 3, this means that the information from an input task is very critical to successfully complete the successor task, and the successor task cannot proceed without that information from that input task.

Information variability (uncertainty) describes the likelihood that the information provided by an input task would change after being initially released. Information variability is related to the stability of a particular task's information, each task has its own variability value (i.e. the information from a particular task has its own probability of changing). The information provided by an input task would change after being initially released could be an estimate of time of the assessment, from the actual value. This matter is related to the

Table 1. Levels of task sensitivity

Value	Description	
0	Weak	The information from the input task is irrelevant (trivial information)
1	Low	A major part of the task can be preformed without information from the input task (verification information)
2	Medium	The task can be started without complete information from the input, but partial information is necessary
3	High	It is impossible for the task to proceed without complete information from the input task

Table 2 Levels of information variability

Value	Description		Likehood of Change
0	Definite	A relatively certain outcome will result	Very low
1	Stable	An outcome can be identified as highly probably (90% sure)	Low
2	Unknown	A value of intervals can be identified, but there is no way to conclude which value is more likely	Medium to High
3	Unstable	It is not possible to identify any limits on the outcome	Very High

Table 3. Ranges of dependency strength and their significance

Dependency Strength	Description
0 - 2	• Dependency is weak • Low risk of rework
3 – 5	• Dependency is moderate • Moderate risk of rework
6 - 9	• Highly sensitive to change • High risk of rework

variability of the input task and the impact of this variability on the successor task.

The variability attribute is made up of descriptions of four different levels of dependency. A numerical value is allocated to each dependency level as shown in Table 2 (Yassine et al., 1999).

A task displays a high variability if the input task is unable to guess a value, or range of values for the output of the task. On the other hand if a task displays low variability the input task can provide a good estimate on the output value of the task. Marks in the DSM are replaced by numerical dependency strength, the product of information variability and task sensitivity. Information variability and task sensitivity are related but are independent. Task sensitivity is a measure related to a dependent task, whereas information variability is related to an input task. Therefore it is suitable to multiply task sensitivity and information variability. The dependency strength can range from 0 to 9.

A low value for either task sensitivity or information variability neutralizes the influence of the other. 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 situation, the dependent task can be started based on a good guess on the predecessor information. On the other hand, if a task is extremely sensitive to its predecessor information (i.e. a sensitivity value of 3) and large variability (i.e. a variability value of 3) means a strong link (i.e. an overall product value of 9).

The resultant matrix will be occupied with elements between 0 and 9, where 0 indicates a non-existent relationship between the two tasks considered and 9 indicates an extremely binding relationship as shown in Table 3 (Yassine et al., 1999).

First the DSM is partitioned; engineers/managers are interviewed about the appropriate values of task sensitivity and information variability. The two values are converted into a single dependency by multiplying them together resulting in a one-dimensional NDSM. The main aim of tearing is to break the information cycle in each block (i.e. task subsets involved in a cycle) and establish a possible starting executions sequence for this subset of tasks.

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.

In order to comply with the above criterion, an index Pi is developed to capture the input to output ratio for each task. The relative position of a task within a block is determined by its Pi value.

Each block in the matrix we tear separately. For each block in the partitioned NDSM we calculate the block in-degrees (BIi) and the block out-degrees (BOi) for all the tasks within that block.

The block in-degrees (BIi) is the sum of the row of task i and the block out-degrees is the sum of the column of task i, considering only the subset of task and marks contained inside the block.

We now calculate the ratio Pi= BIi / BOi index. The task within the block with the minimum Pi value is scheduled first because it requires minimum input and delivers maximum output. This is the same as tearing all the links, within the block, into that task. The scheduled task and all its corresponding marks (above the diagonal) are removed from the block.

Using path searching procedure, we check if the cycle is broken, due to the removal of the scheduled task. If an information cycle is come across again the process of finding new Pi values is repeated for the other tasks within the block. The tasks Pi values are ranked starting with the lowest Pi value within the block, we tear all the feedback mark starting with the highest ranked task.

5. Case Study - Petroleum Oil Field Development Project

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. In this project, DSM operations will be implemented to improve planning, execution and managing the project by reducing the number of feedbacks and reducing the project duration using partitioning and tearing algorithms.

This 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. The project is an activity based performed 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 Standard PVT Study
5.4 Conduct Specialized PVT Study
5.5 Develop PVT Applications

In the next sections, all these tasks will be put into the DSM structure. Then, the partitioning and tearing operations will be implemented to optimize the scheduling of this project.

6. Constructing the Petroleum Oil Field Development Project in DSM

We interviewed reservoir engineers 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 2.

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 path searching procedure which is presented later.

6.1 Topological Sorting

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 an empty row. We will schedule these tasks fist and leave out from further consideration. 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 and leave out this task from further consideration. 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 and leave out this task from further consideration.

Figure 2. 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	10																					
1 4			x	5																				
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	x	10

6.2 Identifying Loops/Circuits using Path Searching Procedure

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. 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.

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. These tasks will form another block in the matrix.

6.3 Tearing the Blocks

We interviewed reservoir and geologist engineers to determine the appropriate values of task sensitivity and information variability as shown in Figure 3. The two values are converted into a single dependency strength by multiplying them together resulting in a one-dimensional NDSM.

Each block in the matrix we tear separately. In block 1 and 2 in the partitioned NDSM we determine the block in-degrees (BIi), the block out-degrees (BOi) and the ratio Pi for all the tasks within block 1 and 2 as shown in Table 4 and 5 respectively.

Therefore task 3.6 is scheduled first within block 1 and task 2.5 is scheduled first within block 2 because they require minimum input and delivers maximum output.

Figure 3. (Task Sensitivity, Information Variability) partitioned DSM of the Petroleum Oil Field Development Project

	1, 1	3, 1	3, 2	5, 1	5, 2	3, 4	3, 5	5, 3	5, 4	5, 5	1, 1	1, 3	3, 3	3, 6	1, 4	2, 1	2, 2	2, 3	2, 4	2, 5	4, 1	4, 2	4, 3	4, 4
1, 2	■																							
3, 1		■																						
3, 2			■																					
5, 1				■																				
5, 2					■																			
3, 4			1, 2			■																		
3, 5			2, 2				■																	
5, 3					3, 1			■																
5, 4					2, 2				■															
5, 5				3, 1				2, 1	1, 2	■														
1, 1											■			3, 3										
1, 3	1, 2										3, 2	■												
3, 3			1, 1									1, 2	■											
3, 6		2, 2				1, 2	3, 2						2, 2	■										
1, 4												3, 2			■									
2, 1														3, 2	3, 2	■								
2, 2																3, 1	■							
2, 3															3, 2		3, 1	■						
2, 4																		2, 1	■					
2, 5																			3, 1	■			1, 2	
4, 1										2, 1				2, 2						3, 2	■			
4, 2																					3, 1	■		
4, 3																						3, 1	■	
4, 4																							2, 1	■

This results in tearing mark (3.6, 3.3) from block 1 and mark (2.5, 4.3) from block 2, hence turning the DSM into the lower triangular form.

7. Constructing the Petroleum Oil Field Development Project in CPM Chart

Since there are no marks above the diagonal, hence turning the DSM into the lower triangular form, we can apply traditional project management tools such as PERT/CPM. From the critical path of the CPM chart shown in Appendix 1, it is noticed that the project duration is now 86 days. The original duration was 100 days; therefore we have saved 14 working days, hence reducing the total cost of the development plan for a new oil field discovered after drilling a number of wells.

8. Conclusions

The NDSM is a new approach to project management, used to represent, analyze dependencies among tasks and show the order in which tasks are performed. This method provides a way of managing feedbacks in engineering projects management. The main advantage of the NDSM over traditional project management tools is in its compactness and ability to present an organized and efficient mapping among tasks that is clear and easy to read regardless of size.

In partitioning the DSM, we used a path searching method to identify loops/circuits. Partitioning the DSM resulted in minimizing the amount of iterative loops by half, therefore reducing the amount of delay in the project. We initially had 4 feedbacks, partitioning resulted in 2 feedbacks, hence we reduced the number of dependencies. Analyzing the partitioned DSM revealed which tasks were parallel, which were sequential and which were coupled. The outcome from partitioning the DSM was a faster development process by optimising the availability of information in the project.

Tearing the NDSM resulted in breaking the information cycle in block 1 and 2 in the project based on the task sensitivity and variability of tasks involved in the circuits. Tasks that require minimum input and deliver maximum output are scheduled first.

Since we reduced all the feedback marks in the DSM, we applied a traditional project management tool and constructed a CPM chart after interviewing engineers about the appropriate task durations. The original project duration was 100 days. By applying partitioning and tearing algorithms we reduced the project duration to 86 days (saving 14 working days), therefore reducing the total cost of the development plan for a new oil field discovered after drilling a number of wells.

In conclusion, this paper describes algorithms for partitioning and tearing which are the fundamental operations in NDSM. The NDSM method has been successfully implemented in the engineering projects management as shown in the case study of the Petroleum Oil Field Development project which reduces significant number of design iterations involved.

Table 4. Summary results of block in-degrees, block out-degrees, Pi ratio and rank for block 1

Task	BIi	BOi	Pi	Rank
1.1	9	6	1.5	-
1.3	6	2	3	-
3.3	2	4	0.5	-
3.6	4	9	0.44	1

Table 5. Summary results of block in-degrees, block out-degrees, Pi ratio and rank for block 2

Task	BIi	BOi	Pi	Rank
2.5	2	6	0.33	1
4.1	6	3	2	-
4.2	3	3	1	-
4.3	3	2	1.5	-

References

Austin, S., Baldwin, A., Li, B., and Waskett, P. (2000). Analytical design planning technique a dependency structure matrix tool to schedule the building design process. *Construction Management and Economics*, 18 (2), pp. 173-182.

Banerjee, A., Carrillo, E. and Paul, A. (2007). Projects with sequential iteration: Models and complexity. *IIE Transactions*, 39 (5), pp. 453-463.

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*, 48 (3), pp. 292-300.

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*, 42 (21), pp. 4623- 4637.

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

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*, 25 (3), pp. 300-314.

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

Eppinger and Ulrich. (2003), *Product Design and Development*, McGraw-Hill, New York.

Maheswari, J. and 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.

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.

Steward, D. (1981). The Design Structure Matrix: A Method for Managing the Design of Complex Systems. *IEEE Transactions on Engineering Management*, 28 (3), pp. 71-74.

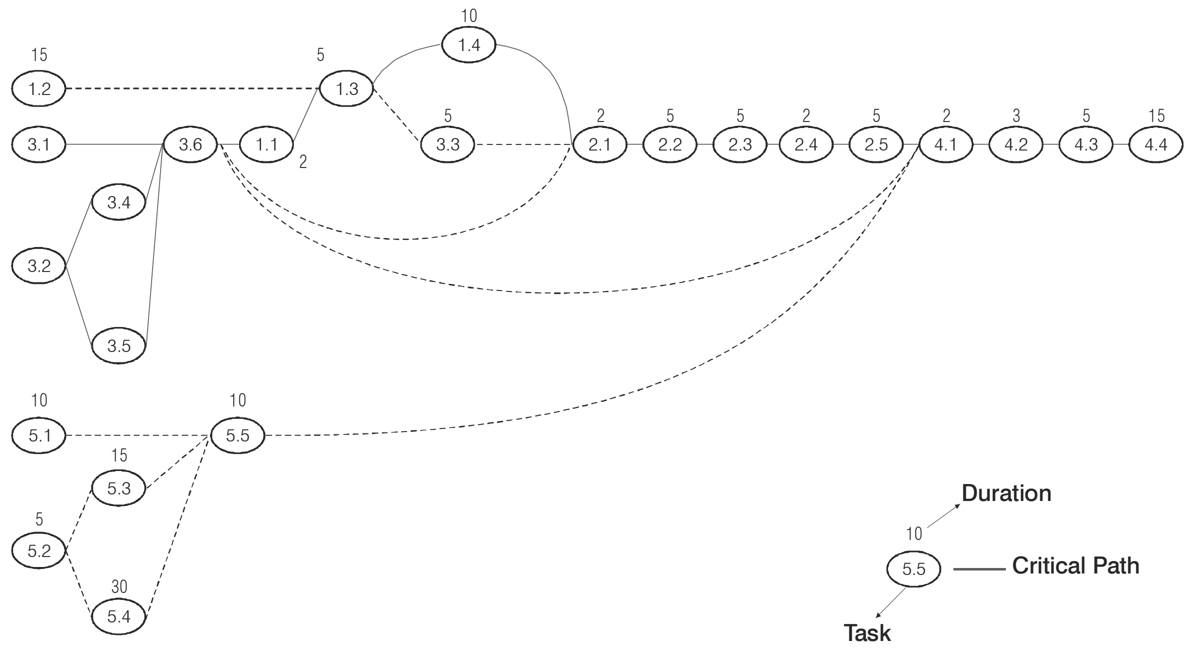
Yassine, A., Falkenburg, D., and Chelst, K. (1999). Engineering design management: and information structure approach. *International Journal of Production Research*, 37 (13), pp. 2957 – 2975.

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.

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 in the Department of Mechanical and Manufacturing Engineering at Auckland University of Technology, New Zealand. 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 engineering systems design. His work has appeared in *International Journal of Reliability, Quality and Safety Engineering; Reliability Engineering and System Safety; Applied Mathematical Modelling; International Journal of Modelling and Simulation; International Journal of Performability Engineering and other publications.*

Appendix 1. CPM chart of the Petroleum Oil Field Development Project


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