國立雲林科技大學工業工程與管理所
Graduate school of Industrial Engineering & Management,
National Yunlin University of Science & Technology

系統可靠度實驗室 System Reliability Lab.
http://campusweb.yuntech.edu.tw/~qre/index.htm

Project scheduling for collaborative product development using DSM

作　　者：Chun-Hsien Chen, Shih Fu Ling, Wei Chen
出　　處：International Journal of Project Management 21 (2003) 291-299
報 告 者：郭秉裕
指導老師：童超塵　教授

Content

- Introduction
- A brief review on DSM
- Project scheduling
- Project rescheduling
- A case study
- Conclusion

Keywords: New product development; Managing projects; International project; Collaborative product development; Design structure matrix (DSM)

Introduction (1/3)

- 新產品發展(new product development; NPD)為一導引產品從概念到產品上市之流程或準則，需為眾多知識領域投入之跨學科活動。在NPD專案中，工作預收率(rate of advance)將強烈影響在此相同專案中其他工作進展。故此期況下，解決龐大複雜排程涉及與大量動態工作(dynamic tasks)互聯為一真正挑戰。
- 然而利用Concurrent Engineering、Critical Path Methods等技術，在排序顯示及控制產品流程呈現出為較弱方面，因其在路徑中僅允許單向進展(one-way progression)，而忽略意疊代(iteration)和回饋循環(feedback loops)。

Introduction (2/3)

- 疊代(Iteration)為產品發展流程中基本特點，其常發生於當上游工作(upstream task)發現某排序(sort)發生錯誤或不相容時或當修正後訊息經由上游工作傳遞至下游工作(downstream task)而進行重覆(repeated)。
- 設計/相依結構矩陣(Design/Dependency Structure Matrix; DSM)經由資訊流分析(information flow analysis)證實，對於計畫和管理產品發展專案為有效工具。此技術能夠直覺表示在大量專案工作中其複雜相依性和提升發展流程中潛在疊代可見度。

Introduction (3/3)

- 產品發展由於其不確定性與固有多樣性，其排程、監控及管制為管理NPD專案基本要素。NPD專案中工作複雜資訊流and/or相依造成專案排程困難和不穩定性，其固有疊代特性更加惡化此情況，此篇利用DSM建構模式，並解決在NPD專案中所遭遇排程複雜性及工作互相關連問題。

A brief review on DSM (1/3)

- DSM被產品發展流程所採取三個主因:
  1. 解決多數以graph-based技術，來表示工作數目與複雜度。
  2. 矩陣易於電腦操作及儲存。
  3. 藉著重新排序工作順序而得到新的專案工作架構。
- 三種DSM modeling在產品發展中各具有不同的應用，根據所將採取之問題本質來選擇適當模式。

Table 1
Common DSM classifications [4][a]

Approach	Application
Parameter-based modeling	System architecture analysis, product re-design
Task-based modeling	Project planning, NPD process analysis
Hybrid design modeling	Coupled task activities, NPD management

[a] DSM, Design Structure Matrix; NPD, New Product Development.

A brief review on DSM (2/3)

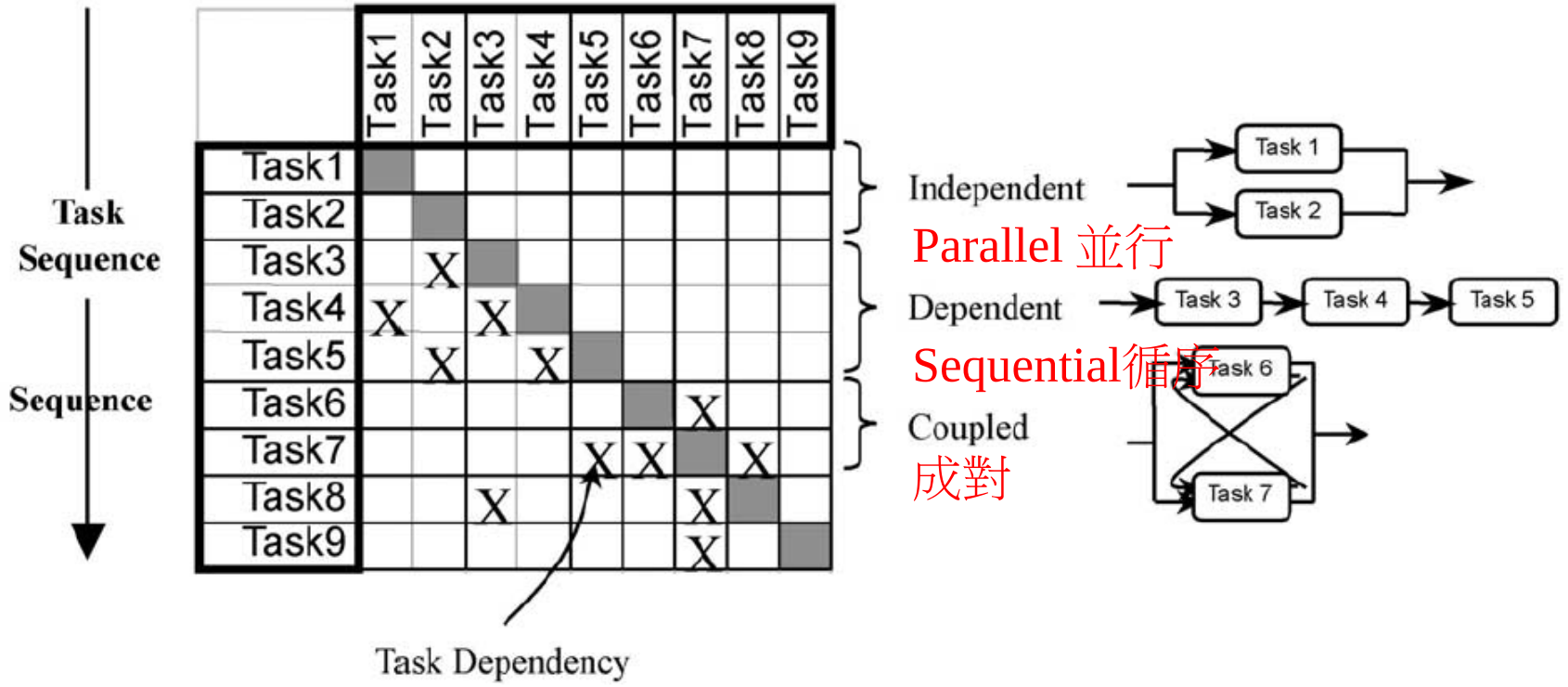


Fig. 1. Sample task-based design structure matrix.

A brief review on DSM (3/3)

- 疊代為多數NPD專案中所常見，尤其在產品設計階段，設計疊代造成工作重置且需額外溝通與交涉而導致發展階段延長。為了加速疊代設計流程，DSM提議操作矩陣元素以致使疊代行為從矩陣中所移除，分割(partitioning)和撕裂(tearing)為主要兩程序，可將矩陣轉換成下三角形(lower triangular)型式矩陣。
- 因此，工作執行順序重命令(re-ordered)、疊代次數減少(reduced)且降低工作數目涉入疊代迴圈內，以提高更迅速之發展流程。

Project scheduling (1/3)

- 專案排程建構包括五個主要步驟：
  1. read the task of the DSM in sequuence
  2. read the estimate duration for this task
  3. Correctly position it into the schedule according to its relationship with other tasks
  4. Repeat step 1-3 for all the tasks until no tasks left; and
  5. Confirm the feasibility of the schedule generated
- 為了簡化問題複雜性，在建構NPD專案之新排程上，本篇對於成對性或循環性區塊(block)技術呈現上，起始假設僅執行一次，即初步專案排程忽略疊代。

Project scheduling (2/3)

- 專案時間計算程序(註:上對角線矩陣之marks忽略條件下):
  1. Start with the terminal task of the project, i.e. the task which has no successors.
  2. Select the closest predecessor as the next task (node) and trace back for all the predecessors along the same path.
  3. Continue step 2 to trace back the whole matrix until the start task of the project is found.
  4. Link all the passed tasks in the same pathtogether and sum up the tasks total duration, which forms a potentially feasible path for the project to execute.
  5. Repeat steps 1~4 to search all the paths and calculate every path's duration.
  6. Compare all the path durations. The longest one is the critical path.

Project scheduling (3/3)

- 預期專案依照所訂定排程如期般執行多為不切實際，由於協同產品(collaborative product)本質為動態不確定性，如消極意外事件delay, iteration等。必須在排程中預留某特定數量時間-寬放時間(slack time)，不僅確保專案保持在排程中，也改善排程處理釘子(snags)能力。

Project rescheduling (1/4)

- 專案重排(project rescheduling)已成為管理NPD專案重要一部分，其兩主要功能為：
  1. 當消極意外事件發生時，及時顯示因工作延遲所造成排程相關影響。
  2. 迅速修正/調整目前排程。
- 本篇主要根據DSM來處理工作延遲，其發生狀況包含下類：
  1. The delayed tasks are not coupled
  2. The delayed tasks are coupled
  3. The iteration of coupled blocks

Project rescheduling (2/4)

- 當工作延遲或疊代發生時，正運行之整個專案應被計算與顯示於排程上，對於上述兩種情況其處理步驟些許不同，詳細如下：

- 處理工作延遲步驟

1. 獲取得延遲或將被延遲之工作資訊(如工作名稱和延遲時間)
2. 經由DSM中對應工作延遲所在行，往下追溯其將被影響之工作(tasks/successor)
3. 經由步驟2中已找出之工作，追溯其另將被影響之工作，繼續此程序，直到所有被影響工作定義為止
4. 步驟3獲得所有定義被影響之工作，修正其開始和完成時間
5. 最終在專案排程中顯示整個專案之運行

Project rescheduling (3/4)

- 處理疊代部分

1. 獲取得需要疊代之工作方塊(task block)資訊
2. 計算疊代方塊時間
3. 計算疊代方塊內各工作起始(beginning)和終了(ending)時間
4. 經由DSM中對應疊代方塊所在行，往下追溯其將被影響之工作
5. 經由步驟4中已找出之工作，追溯其另將被影響之工作，繼續此程序，直到所有被影響工作定義為止
6. 步驟6獲得所有定義被影響工作，修正其開始和完成時間
7. 最終在專案排程中顯示整個專案之運行

Project rescheduling (4/4)

- 以專案重排程演算法去找尋根據DSM所建立排程之要徑，其概念類似以連鎖倒推策略(backward chaining stregy)運用在知識工程(knowledge engineering)。

本研究提出三種不同專案重排程方法如下：

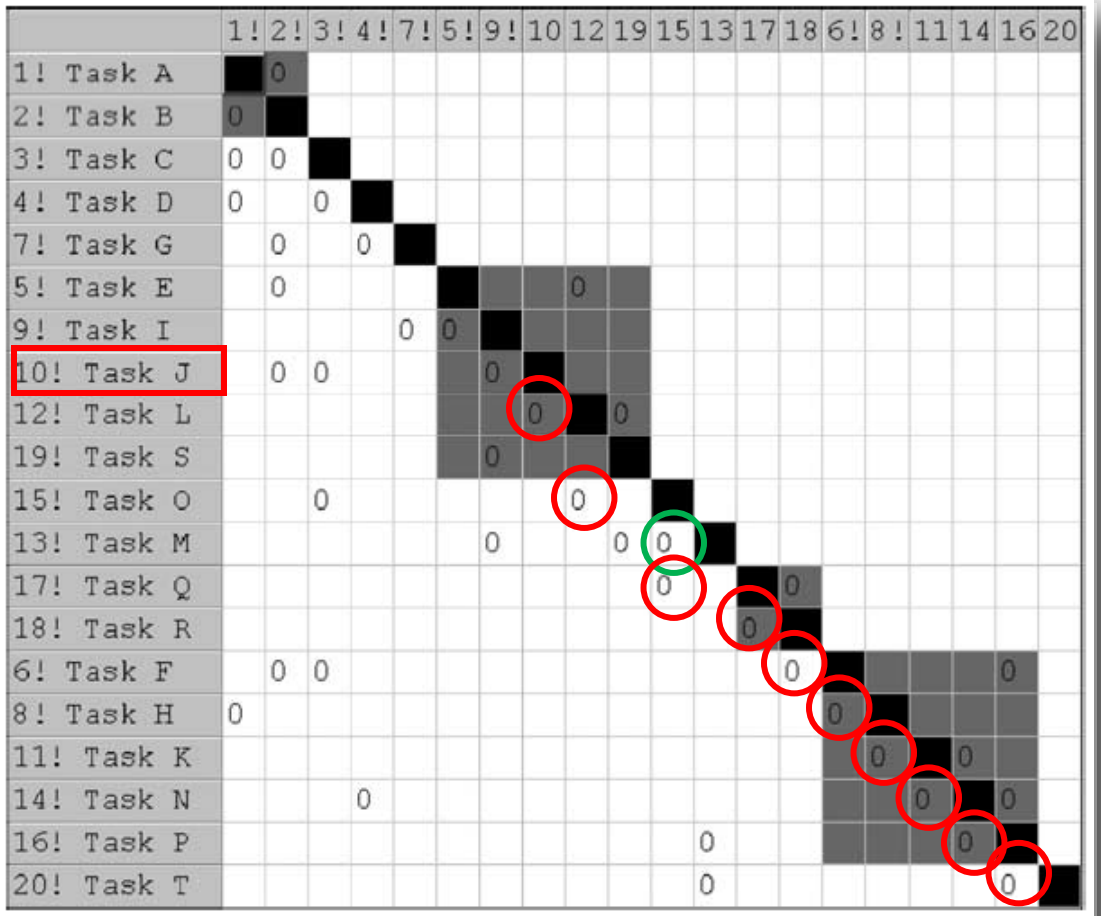
1. Revise schedule by averaging
2. Revise schedule based on the weight of tasks
3. Revise schedule manually

A case study (1/7)

Table 2
Task information

No.	Task List	Duration (t)	Successor (dependency)	Weight
1	Task A	1	B, C, D, H	5
2	Task B	1.5	A, C, E, F, G, J	5
3	Task C	0.5	D, F, J, O	8
4	Task D	1	G, N	3
5	Task E	2.5	I	3
6	Task F	1	H	3
7	Task G	1.5	I	4
8	Task H	4	K	10
9	Task I	1	J, M, S	7
10	Task J	2	L	5
11	Task K	1.5	N	2
12	Task L	5	E, O	10
13	Task M	1	P, T	2
14	Task N	3	K, P	5
15	Task O	0.5	M, Q	3
16	Task P	2	F, N, T	2
17	Task Q	1	R	1
18	Task R	1.5	F, Q	3
19	Task S	2.5	L, M	6
20	Task T	2		1

A case study (2/7)



	1!	2!	3!	4!	7!	5!	9!	10	12	19	15	13	17	18	6!	8!	11	14	16	20
1! Task A		0																		
2! Task B	0																			
3! Task C	0	0																		
4! Task D	0		0																	
7! Task G		0		0																
5! Task E		0							0											
9! Task I					0	0														
10! Task J		0	0				0													
12! Task L								0		0										
19! Task S							0													
15! Task O			0						0											
13! Task M							0			0	0									
17! Task Q											0			0						
18! Task R													0							
6! Task F		0	0											0					0	
8! Task H	0														0					
11! Task K																0		0		
14! Task N				0													0		0	
16! Task P												0						0		
20! Task T												0							0	

Fig. 3. The partitioned Design Structure Matrix (DSM).

A case study (3/7)

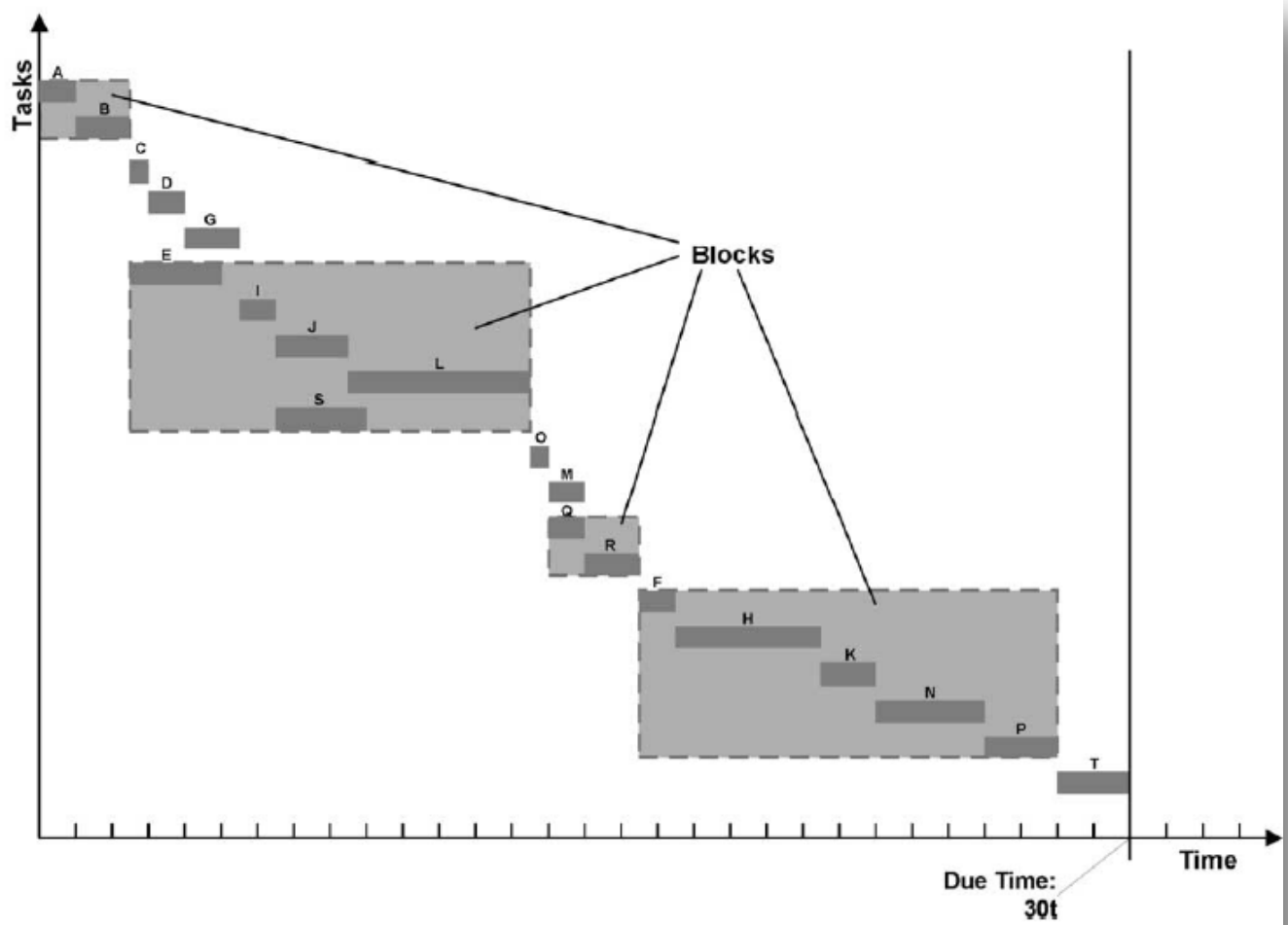


Fig. 4. Project schedule.

A case study (4/7)

Revise schedule by averaging

Postulating k tasks should be shortened.

$$t_{is} = \frac{T_d}{\sum_{j=1}^{k} t_j} \times t_i$$

where t_{is}: the shortened time of each task; T_d: the overdue time of the entire schedule; t_j: the duration of the jth task ($1 \leqslant j \leqslant k$); t_i: the duration of the ith task to be shortened ($1 \leqslant i \leqslant k$).

$t_{Ls}=$	$t_{Os}=$	$t_{Qs}=$	$t_{Rs}=$	$t_{Fs}=$
0.23t;	0.02t;	0.05t;	0.07t;	0.05t;
$t_{Hs}=$	$t_{Ks}=$	$t_{Ns}=$	$t_{Ps}=$	$t_{Ts}=$
0.19t;	0.07t;	0.14t;	0.09t;	0.09t.

Revise schedule based on the weight of tasks

$$t_{is} = \frac{T_d}{\sum_{j=1}^{k}\left(t_j^2 / \left(w_j \times \sum_{j=1}^{k} t_j\right)\right)} \times \left(t_i^2 / w_i \times \sum_{j=1}^{k} t_j\right)$$

where t_{is}: the shortened time of the ith task; T_d: the overdue time of the entire schedule; t_j: the duration of the jth task ($1 \leqslant j \leqslant k$); w_j: the weight of the jth task ($1 \leqslant w_j \leqslant 10$); t_i: the duration of the ith task which should be shortened ($1 \leqslant i \leqslant k$); w_i: the weight of the ith task that should be shortened ($1 \leqslant w_i \leqslant 10$).

$t_{Ls}=$	$t_{Os}=$	$t_{Qs}=$	$t_{Rs}=$	$t_{Fs}=$
0.16t;	0.01t;	0.07t;	0.05t;	0.02t;
$t_{Hs}=$	$t_{Ks}=$	$t_{Ns}=$	$t_{Ps}=$	$t_{Ts}=$
0.11t;	0.07t;	0.12t;	0.13t;	0.26t.

A case study (5/7)

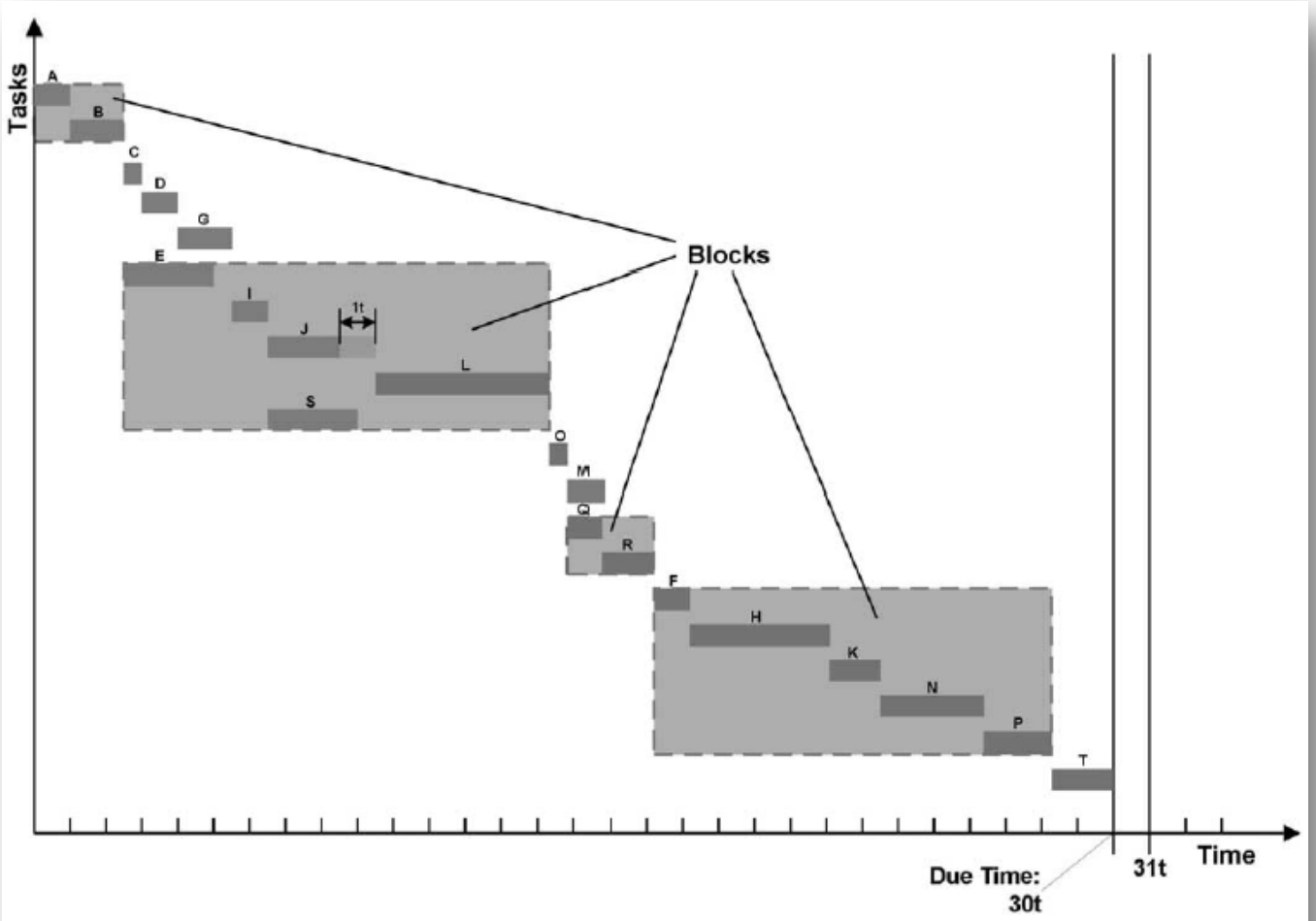


Fig. 5. Revised project schedule based on the weight of tasks (for task delay).

A case study (6/7)

For the first iteration:

$$T_1^i = \alpha_1 \times T^i \tag{1}$$

where T_1^i: the duration of the first iteration; α_1: the coefficient assessed by scheduler to predict the duration time of the first iteration; T^i: the original duration of coupled block, which can be obtained from the schedule.

For the kth task in a block with p tasks at the *first* iteration,

$$t_{k1} = T^i + \alpha_1 \times \sum_{q=1}^{k} t_q \quad \text{for } 1 \leqslant k \leqslant p \tag{4}$$

where t_{k1}: the time when the kth task in the block finished at the first iteration; T^i: the original duration of coupled block; α_1: the coefficient assessed by scheduler to predict the duration of the first iteration; t_q: the duration of the qth task in the block, $1 \leqslant q \leqslant k$.

$T_{i1} = 7.8t$;
$t_{E1} = 14.5t$; $t_{I1} = 15.1t$; $t_{J1} = 16.3t$; $t_{L1} = 19.3t$; $t_{S1} = 20.8t$.

A case study (7/7)

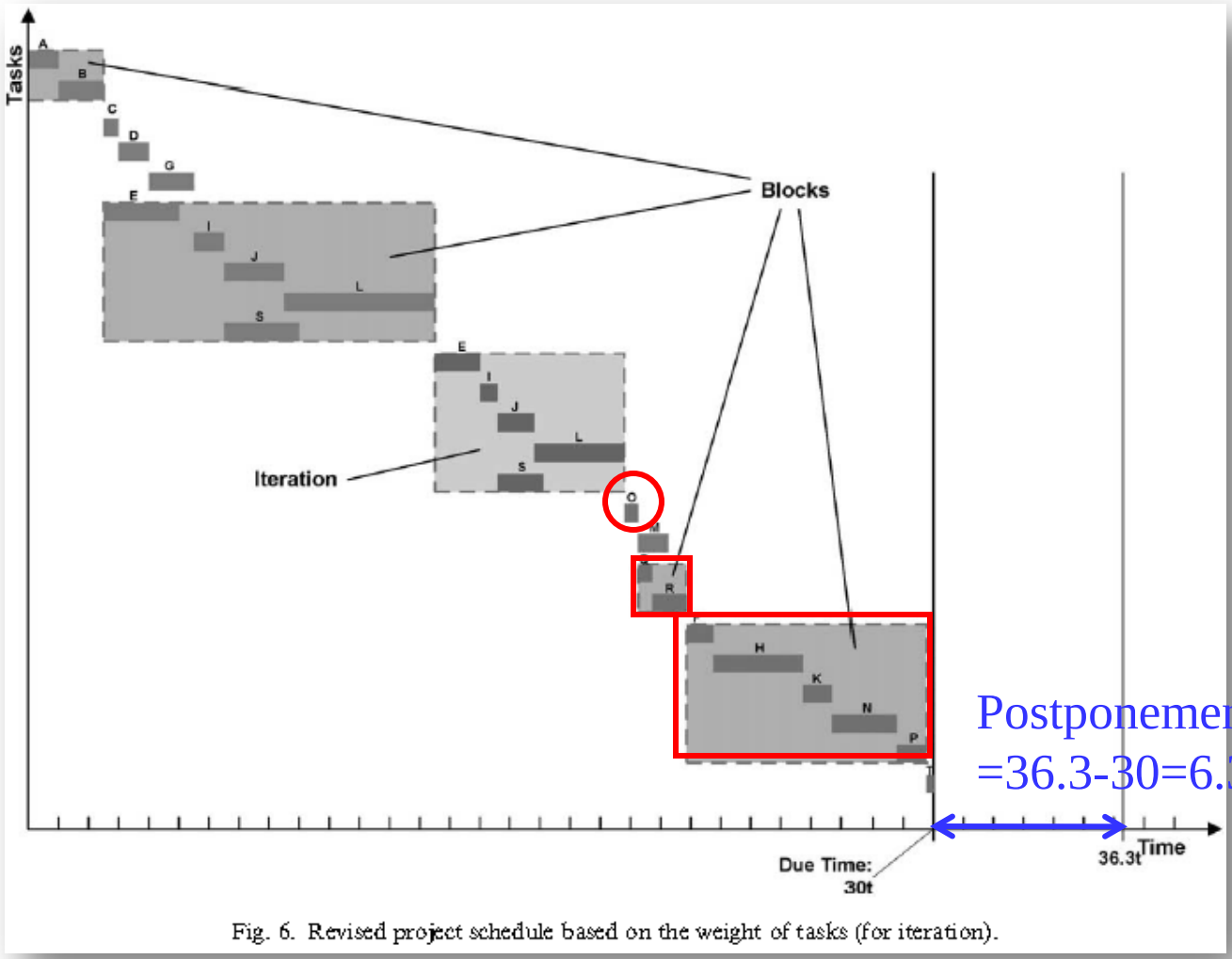


Fig. 6. Revised project schedule based on the weight of tasks (for iteration).

Postponement(Td)
=36.3-30=6.3

Conclusion

- 本篇以DSM根據之專案排程與重排程架構顯示，一簡單專案排程案例被用來演示其所提出排程與重排程技術功能，其結果對於NPD專案中所遭遇工作互相關連與複雜性能足以掌握，而此原架構對於複雜之協同NPD專案亦能有效監視與控制。