ch3 - 一般原則

本章介紹与解釋一般離散事件模擬軟体的基本概念與方法。

3.1 離散事件模擬觀念。

系統：實体的集合（例如人与机器）彼此交互影响 (system) 透過時間完成任務.

模式（Model）：一個系統抽象的代表，通常包括結構、数学或邏輯関係，而這関係以狀態、實体、屬性、集合、處理、事件、活動與延後描述系統。

系統狀態（System state）：変数的集合，這些変数包含在任何時間足以描述系統之所有需要的訊息。

實体（Entity）：系統中任何事物或成份，需要被明確的表示（例如顧客、服務者、机器）。（在模式中）

屬性（Attributes）：實体的特性（例如等候顧客的優先順序，在零工式現場工作被處理的途徑）

表單（List）：實体的集合以某種邏輯排序（例如顧客在等候線以FIFO或優先順序排列）

事件（Event）：一立即發生使系統狀態改變（例如一位新顧客到達）。

事件通知（Event notice）：一事件在目前或未來的記錄，伴隨著執行該事件的必需資料，例如它含事件的型態與事件時間。

事件表單（Event List）：未來事件的事件通知表單，通常以時間的發生順序排序，亦稱為未來事件表單 future event list（FEL）。

活動（Activity）：一段特定長度的時間（例如，服務時間或到達時間間隔），當它開始通常會知道（雖然它可能以統計分配定義）。

延誤（Delay）：一段未指定長度的時間，該時間無法得知直到其結束（例如，一位顧客在一 LIFO 等候線，其延誤無法得知，因其取決於等候的開始，決定於未來的到達

時鐘（Clock）：一個變數代表模擬時間，稱其為 CLOCK，在範例中便於觀察與遵循。

不同的模擬軟體使用不同的術語（terminology）表示上述相同或類似的觀念。例如表單（List

有時稱為集合(Sets)、等候伐(Queues)或鍊(Chain)表單或集合通常用來保存實体與事件通知，在表單中的實体也通常以某種規則排列，例如FIFO、LIFO或實体的屬性(例如優先順序或到期日)。未來事件表單(FEL)通常以事件通知中的時間排序。活動通常代表一服務時間，到達時間間隔或其他處理時間。活動的期間通常以下列方式定義 1.確定性－正好5分鐘，2.統計的－例如隨机由2,5,7中選取，其產生机率均等，3.取決於系統變数或實体之函数型式－例如鉄礦船之裝載時間為該船允許之載重量與每小時裝載噸数之函数。活動的期間可以計算從事件開始，該期間不受其他事件影响。為了追踪活動與期望完成時間，當模擬事件開始，便創造事件完成時間放在事件通知上。例如目前模擬時間CLOCK=100檢查時間為5分鐘，於是在事件通知上

創造事件完成時間為 t=5。與活動不同，延誤（Delay）的期間並未由建構模式者事先指定，而是由系統的條件決定。一般而言，延誤的結束是由一些條件或相關事件發生。例如，顧客在等候線的延誤取決於顧客在等候線的人數或每位顧客的服務時間，或服務者的狀態。有時候延誤稱為條件等待（Conditional wait），而活動稱為無條件的等待（Unconditional wait）。活動的完成是一事件通常稱為主要事件（primary event），亦即在 FEL 未來事件表單上記錄一項事件通知。延誤通常將實體記錄在另一表單上，例如代表一等候線，直到系統的條件允許處理實體，延誤的完成有時稱為一條件或次要事件，這些事件不會出現在事件通知或 FEL 上。

Example 3.1 (Able and Baker)

考慮在 Example 2.2 的 Able-Baker carhop 例子

系統狀態（System state）

$L_Q(t)$：在時間 t 等候被服務之汽車數目。

$L_A(t)$：在時間 t，Able 的狀態，0代表空閒，1代表忙碌。

$L_B(t)$：在時間 t，Baker 的狀態，0代表空閒，1代表忙碌。

實体（Entities）：顧客（汽車）與服務者在本例均不需被明確表示在本例，除非需要顧客的平均數要用為衡量指標。

事件（Events）

到達事件

Able 完成服務

Baker 完成服務

活動（Activities）

表 2.11 中定義的 汽車到達之間隔時間

表 2.12 中定義的 Able 服務時間分配

表 2.13 中定義的 Baker 服務時間分配

延誤（Delay）顧客在等候線等待直到 Able 或 Bake 空閒。

上述模式成分的定義提供模式靜態的描述。除此之外，成份間的動態關係

互動也是有需要，需要被回答的問題如下

1. 事件如何影响系統狀態，實体屬性，等候表單內容。
2. 活動如何被定義？(確定机率或其他相關係) 什麼事件記錄活動開始或結束？活動可否開始不需管系統狀態，或是活動的開始必需基於某種系統條件？(例如机器活動無法開始，除非机器是閒置，沒有故障，沒有在保養。)
3. 那一事件驅動延誤的開始或結束？在何種條件下延誤開始或結束？
4. 在時間点0時之系統狀態，何種事件應必產生使模擬開始？

離散模擬透過一序列的系統快照(snapshot)，代表系統隨時間的演變。在一給定時間 t (CLOCK = t) 包含在時間 t 的系統狀態，並將目前進行中所有活動列於 FEL 中，Figure 3.1 顯現一系統模型。

Clock	*System State*	*Entities and Attributes*	*Set 1*	*Set 2*	...	*Future Event List, FEL*	*Cumulative Statistics and Counters*
t	$(x, y, z, \ldots)$					$(3, t_1)$ – Type 3 event to occur at time t_1 $(1, t_2)$ – Type 1 event to occur at time t_2	

Figure 3.1 Prototype system snapshot at simulation time t.

3.1.1 The Event-Scheduling / Time-Advance Algorithm

事件排程/時間前進演算法，這項机置將模擬時間前進並確保所有事件的發生以未來事件表單正確的時間順序進行。安排未來事件代表將一事件之開始時間，結束時間置於未來事件表單上，但是在真實世界未來事件並不需要安排，只是發生。在模式中，這類事件通常代表某些活動的結束。

在任何時間 t，未來事件表單包含所有事先安排的事件与時間，且時間依發生順序排序，亦即時間必須滿足 $t < t_1 \leq t_2 \leq t_3 \leq \ldots \leq t_n$

t 代表目前模擬時間，下一個將發生事件的時間是 t_1，稱為立即事件（imminent event），當模擬時間前進至 t_1 時，立即事件就從未來事件表單上移除。在 t_1 時，新的事件或許

會產生，如果產生將產生事件通知並放置於未來事件表單適當位置。在時間 t_1 系統快照被更改後，模擬時鐘被前進至新的立即事件與執行。這項過程重覆直到模擬時間結束。這一連串的行動必須執行將時間前進與建立系統快照，稱為事件排程/時間前進演算法，其步驟列於 Figure 3.2

Old system snapshot at time t

CLOCK	*System State*	...	*Future Event List*	...
t	(5, 1, 6)		$(3, t_1)$– Type 3 event to occur at time t_1 $(1, t_2)$– Type 1 event to occur at time t_2 $(1, t_3)$– Type 1 event to occur at time t_3 . . . . $(2, t_n)$– Type 2 event to occur at time t_n	

Event-scheduling/time-advance algorithm

Step 1. Remove the event notice for the imminent event (event 3, time t_1) from FEL

Step 2. Advance CLOCK to imminent event time (i.e., advance CLOCK from t to t_1).

Step 3. Execute imminent event: update system state, change entity attributes, and set membership as needed.

Step 4. Generate future events (if necessary) and place their event notices on FEL ranked by event time. (*Example:* Event 4 to occur at time t^*, where $t_2 < t^* < t_3$.)

Step 5. Update cumulative statistics and counters.

New system snapshot at time t_1

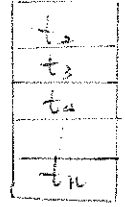
CLOCK	*System State*	...	*Future Event List*	...
t_1	(5, 1, 5)		$(1, t_2)$– Type 1 event to occur at time t_2 $(4, t^*)$– Type 4 event to occur at time t^* $(1, t_3)$– Type 1 event to occur at time t_3 . . . . $(2, t_n)$– Type 2 event to occur at time t_n	

Figure 3.2. Advancing simulation time and updating system image

當模擬進行時，FEL的長度與內容經常改變，於是在模擬中有效管理將對電腦程式代表欲研究之模式產生重大影响。对於表單的管理稱為(list processing 表單處理)。主要的表單處理程序是從FEL中，將立即事件移除，增加新事件，與偶爾移除一些事件(called cancellation of an event)。立即事件通常在表單的頂端，從上面移除最為有效率。增加新事件(移除舊事件)通常需要對表單執行搜尋。

Figure 3.2 示範在FEL移除與增加事件。事件3的事件時間是生在 t_1，代表服務時間的完成，即然它是在時間 t 的立即事件，於是在 step 1，將事件3從FEL上移除。在step 4，事件4(到達事件)產生時間在 t^* 被產生，一種決定它在FEL上正確位置的方法是執行由上而下的搜尋。

If $t^* < t_2$, place event 4 at the top of the FEL.
If $t_2 \le t^* < t_3$, place event 4 second on the list.
If $t_3 \le t^* < t_4$, place event 4 third on the list.
⋮
If $t_n \le t^*$, place event 4 last on the list.



另一種方法是執行由下而上的搜尋。最無效率的方法是讓FEL任意沒有順序放置，如此便需要在表單執行一次搜尋。

系統快照在時間0的時点，定義系統的起始條件又稱為外生事件（exogenous events）。例如在Figure 3.2，在$t=0$時，系統狀態(5, 1, 6)可能代表在系統中三個不同地点的顧客抵目。外生事件的發生是不受系統影响，常見的例子是顧客的到達。在時間0，第一項到達事件被產生並且安排在FEL上。當時鐘被前進到第一項到達事項，第二項到達事件便被產生。首先是產生一到達時間間隔a^*，加到目前時間CLOCK=t，即可得到事件時間

$t + a^* = t^*$ 用以代表新事件的放置位置於FEL上。這種代表產生外部到達

事件的方法，稱為 bootstrapping。bootstrapping 在 Figure 3.3 中示範。

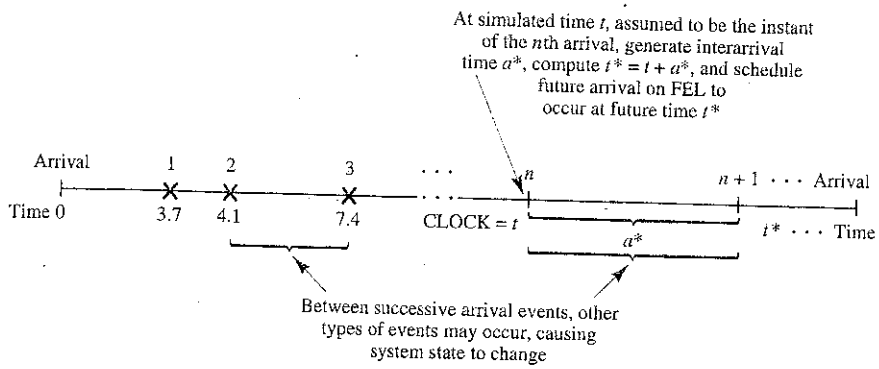


Figure 3.3. Generation of an external arrival stream by bootstrapping.

前面三項事件到達時間間隔分別是3.7, 0.4 和 3.3 分鐘。到達時間間隔正負的結果是一主要事件的例子。

第二個例子是未來事件如何產生，當一位顧客完成服務，在目前時間 CLOCK = t，如果下一位顧客出現，將產生下一位顧客新的服務時間 s^*，下一次服務完成時間將會前進至 $t^* = t + s^*$ 並且將其放置於 FEL 上。除此之外，服務完成事件將會被產生而且安排在一項到達事件的時間點，如果至少有一位服務者是空閒

服務時間是活動的例子。服務的開始是一條件事件，因為它的發生是由顧客的到達和服務者是空閒所趨動。服務的完成是主要事件的例子。條件事件是由主要事件所趨動，例如開始服務，僅有主要事件會產生在 FEL 上。

第三個重要例子，是机器的正常運作時間與故障時間交互產生。在時間点 0，會產生正常運作時間並且安排正常運作結束事件。當正常運作結束，將產生一故障時間，故障時間結束事件將置於 FEL 上。正常運作時間與故障時間將以交互產生的方式遍過整個模擬程序。正常運作時間與故障時間是活動的例子。正常運作時間與故障時間的結束是主要事件。

每件模擬必需有一停止事件，通為 E，定義為模擬將執行多久。通常有 2 種方法停止模擬。

1. 在時間 0，安排一模擬停止事件，在未來的時間 T_E，於是模擬在 $[0, T_E]$ 之間執行，例如 $T_E = 40$ 小時。

2. 模擬長度 T_E 由模擬本身決定，一般而言，T_E 是某項特別事件發生的時間。例如 T_E 是第一百位顧客完成服務的時間；T_E 是一複雜系統的故障時間；T_E 是戰勝或全部被殺的時間；T_E 是在配送中心運送一日中最後一箱的時間。

3.1.2 World Views

當使用模擬軟體或做手的模擬、模式者、展者通常採用一種 World View 導向建展模式，最流行的是事件-安排導向，在前面已經介紹過，其他為 process-interaction world view 過程交互導向，和 activity-scanning world view 事件掃瞄導向。軟體可能僅支持某種導向，並無法支持所有導向，瞭解不同的導向可提供建構模式的替代方法。

事件-安排導向，模式建構者強調在事件和對系統狀態的影响；過程交互導向模式建構者強調過程，過程交互方法是流行因為其接近直覺，有些軟体包許模式建構者敘述過程通過區塊式網絡建構而其交互影响則自動處理。

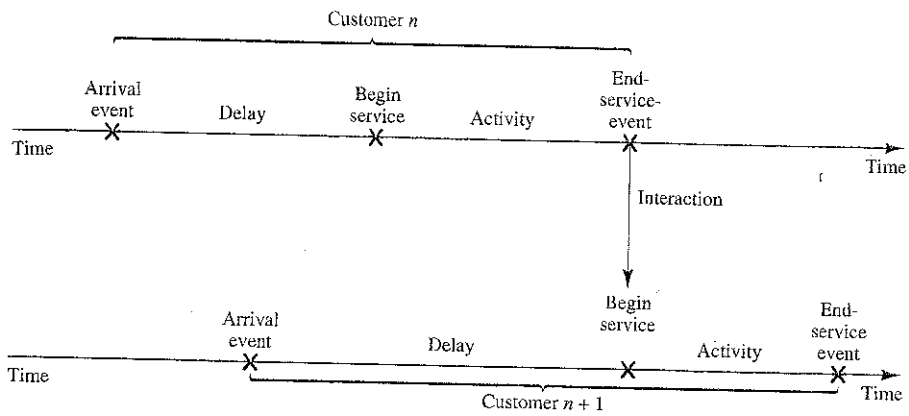


Figure 3.4. Two interacting customer processes in a single-server queue.

Figure 3.4 是過程交互導向的例子，更精確的敘述，過程是將事件依據時間順序排列，而實体則隨著時間推移經過系統。通常許多過程在系統中同時產生而過程間的交互會十分複雜。

事件-安排導向與過程交互導向

均使用一變長(動)時間推進，亦即模擬時鐘會前進至FEL上的下一立即事件。而事件掃描導向，則以一固定時間推進，再決定在該時間点，一些活動是否開始進行。模式建構者專注於允許活動開始的條件，當模擬時間前進，活動開始的條件均會被檢查，如果條件成立，則對應的活動開始。這類方法在概念上較簡單，也較易瞭解与其他分析者易於修改，但是重覆掃描以決定活動是否開始會使電腦之執行時間变慢。於是事件掃描的方法被改良成三階段方法(three-phase approach)，其綜合事件安排和事件掃瞄允許变動時間推進與避免不必要的掃瞄，而保存了事件掃瞄的主要优点。

三階段方法將活動分成B与C兩大類

B類活動：主要事件为無條件事件。
C類活動：條件事件。

B類活動事件可以事先安排，如同事件安排方法，FEL上只包含B類活動在每次將時間推前時，只掃瞄C類活動。

三階段方法重覆執行三階段直至模擬完成。

階段A：將立即事件由FEL移除，並將模擬時鐘推前，並將有相同事件時間的事件由FEL上移除。

階段B：執行所有B類活動，並將其從FEL上移除（例如改變系統狀態，使資源變成可用）。

階段C：掃瞄驅動C類活動的條件，與滿足C類活動。重覆掃瞄所有C類活動。

三階段方法改良執行效率。

3.13 使用事件安排，以手模擬

模擬表用以記錄系統快照（snapshots）當時

商在前推前。

example 3.3 (single-channel queue)

重新考慮, example 2.1 中, 雜貨店只有一位結帳者的範例. 系統包含在線上等候的顧客與正在結帳的顧客. 模式的成份如下.

System state 系統狀態, (LQ(t), LS(t)) 其中 LQ(t) 是等候線上的顧客, LS(t) 在時間 t, 正在接受服務的顧客和目 (0 或 1)

Entities 實體, 服務者与顧客並沒有明確的在模式中建構。

Events 事件

到達 (A), 離開 (D), 停止事件 (E), 在時間 6

Activities 活動

到達時間間隔, 於 Table 2.6 中定義,

服務時間, 於 Table 2.7 中定義

Delay 延後

顧客在等候線的花費的時間

Figure 3.5 与 3.6 顯示, 到達与離開事件, 模擬表於 3.1 顯示。

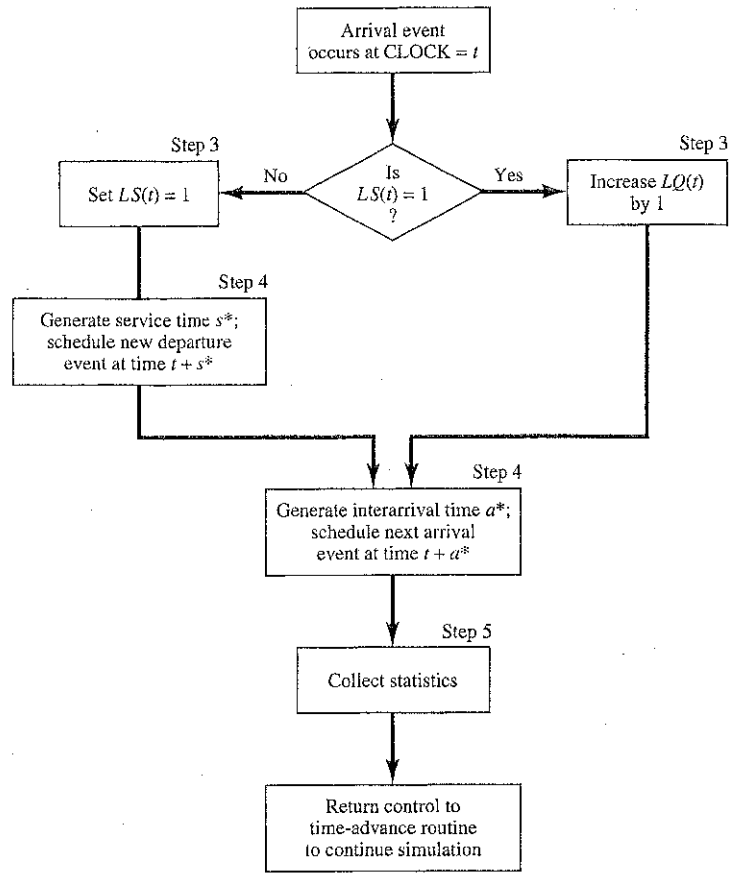


Figure 3.5. Execution of the arrival event.

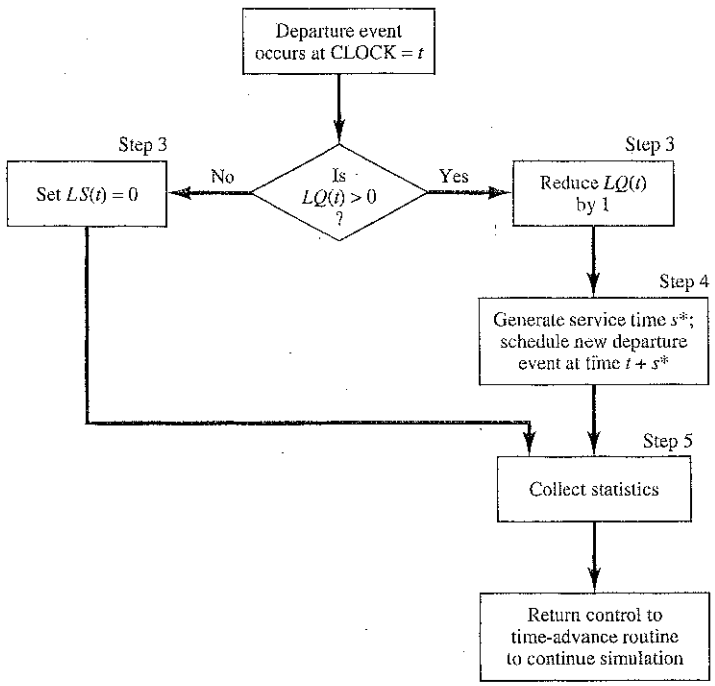


Figure 3.6. Execution of the departure event.

Interarrival Times	8	6	1	8	3	8	···
Service Times	4	1	4	3	2	4	···

起始狀態，第一位顧客於時間 0 到達，並開始接受服務。於表 3.1 中 CLOCK = 0，LQ(0) = 0

Table 3.1. Simulation Table for Checkout Counter (Example 3.3)

	System State				*Cumulative Statistics*	
Clock	$LQ(t)$	$LS(t)$	*Future Event List*	*Comment*	B	MQ
0	0	1	$(D,4)$ $(A,8)$ $(E,60)$	First A occurs $(a^* = 8)$ Schedule next A $(s^* = 4)$ Schedule first D	0	0
4	0	0	$(A,8)$ $(E,60)$	First D occurs: $(D,4)$	4	0
8	0	1	$(D,9)$ $(A,14)$ $(E,60)$	Second A occurs: $(A,8)$ $(a^* = 6)$ Schedule next A $(s^* = 1)$ Schedule next D	4	0
9	0	0	$(A,14)$ $(E,60)$	Second D occurs: $(D,9)$	5	0
14	0	1	$(A,15)$ $(D,18)$ $(E,60)$	Third A occurs: $(A,14)$ $(s^* = 4)$ Schedule next D	5	0
15	1	1	$(D,18)$ $(A,23)$ $(E,60)$	Fourth A occurs: $(A,15)$ (Customer delayed)	6	1
18	0	1	$(D,21)$ $(A,23)$ $(E,60)$	Third D occurs: $(D,18)$ $(s^* = 3)$ Schedule next D	9	1
21	0	0	$(A,23)$ $(E,60)$	Fourth D occurs: $(D,21)$	12	1

LS(0)=1，離開事件与到達事件均在 FEL 上，模擬示安排在 CLOCK=60 時停止。只有 B(所有服務者忙碌時間)与 MQ(最大等候线長度)兩个統計量會被收集。Comments 這栏是協助讀者瞭解 a^* 和 s^* 是到達間隔時間与服服。在 CLOCK=0，立即事件是(D, 4)，CLOCK 被推前至 4，(D.4) 將由 FEL 移除。即此 LS(t)=1 ($0 \le t \le 4$)，B=0 增加为 B=4，並將 LS(4)=0，FEL 上只剩下 (A,8) 与 (E,60) 兩事件。CLOCK 再~~要~~進前至 8，(A,8) 從 FEL

上移除，產生 $S^*=1$, $a^*=6$. 所以 FEL 上有 (D, 9) (A, 14) (E, 60)。其餘部分由讀者自行閱讀，篇幅有限僅模擬至 CLOCK=2。

Example 3.4（續前例）

分析者為了瞭解平均服務時間與顧客在系統中(response time, 包含等候與服務)超過4分鐘的比例。為了搜集上述資料必需記錄每位顧客的到達與離開時間。因此以 (D, 4, C1) 代表第一位顧客在 time 4 時離開，並且存放在 "Checkout line"。

Entities 實體：(C_i, t) 代表顧客 C_i 在時間 t 的到達時間

Event notices 事件通知：

(A, t, C_i) 顧客 C_i 在未來時間 t 到達

(D, t, C_j) 顧客 C_j 在未來時間 t 離開

Set 集合 "CHECKOUT LINE" 目前在 checkout counter 所有顧客，包含接受服務與等候兩部分。

S, F, 與 N_D 三個新的統計量將被收集，S 代表目前已經離開所有顧客接受服務時間之總和。F 代表在系統中超過 4

4分鐘的人數。N_D 代表直至目前離開系統的人數。

模擬表顯示於 Table 3.2

Table 3.2. Simulation Table for Example 3.4

	System State		List	Future Event List	Cumulative Statistics		
Clock	$LQ(t)$	$LS(t)$	"CHECKOUT LINE"		S	N_D	F
0	0	1	(C1, 0)	(D, 4, C1) (A, 8, C2) (E, 60)	0	0	0
4	0	0		(A, 8, C2) (E, 60)	4	1	1
8	0	1	(C2, 8)	(D, 9, C2) (A, 14, C3) (E, 60)	4	1	1
9	0	0		(A, 14, C3) (E, 60)	5	2	1
14	0	1	(C3, 14)	(A, 15, C4) (D, 18, C3) (E, 60)	5	2	1
15	1	1	(C3, 14) (C4, 15)	(D, 18, C3) (A, 23, C5) (E, 60)	5	2	1
18	0	1	(C4, 15)	(D, 21, C4) (A, 23, C5) (E, 60)	9	3	2
21	0	0		(A, 23, C5) (E, 60)	15	4	3

例如在時間 4. 顧客 C1 離開，C1 由 CHECKOUT LINE 移除。S 變成 4，$N_D=1$，F=1。在時間 21 是執行 (D, 21, C4) 離開事件，服務時間 (Response time for C4) = CLOCK TIME − 到達時間 = 21 − 15 = 6

於是 S 增加 6 分鐘，F 與 N_D 各增加 1。

以模擬 21 分鐘而言，平均服務時間是

$S/N_D = 15/4 = 3.75$ 分鐘，在系統中超過 4 分鐘的比例是 $F/N_D = 3/4 = 0.75$，實際應用時應將服務時間加長，以增加模擬的可靠度。

Example 3.5 The Dump Truck Problem

有6輛卡車用來拖煤，Figure 3.7 提供一作業概要。每一卡車可由2部裝載机器中

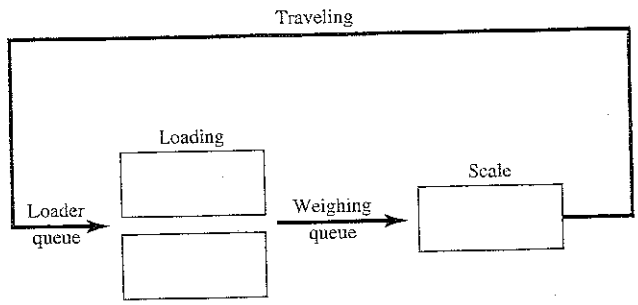


Figure 3.7. Dump truck problem.

的一部裝載，完成後立刻前移過磅。裝載與過磅均採用 FIFO 法。完成過磅後，卡車卸煤然後再回來裝載（旅行時間）。裝貨時間，過磅，旅行時間之分配分別顯示於 Table 3.3, 3.4 与 3.5。

本模擬的目的為分析裝載机器與磅秤的使用率（忙碌時間的百分比）。模式的成分如下：

System state 系統狀態

LQ(t)：在時間 t，等候裝載卡車数目

Table 3.3. Distribution of Loading Time for the Dump Trucks

Loading Time	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
5	0.30	0.30	1–3
10	0.50	0.80	4–8
15	0.20	1.00	9–0

Table 3.4. Distribution of Weighing Time for the Dump Trucks

Weighing Time	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
12	0.70	0.70	1–7
16	0.30	1.00	8–0

Table 3.5. Distribution of Travel Time for the Dump Trucks

Travel Time	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
40	0.40	0.40	1–4
60	0.30	0.70	5–7
80	0.20	0.90	8–9
100	0.10	1.00	0

L(t)=在時間t，正在裝載卡車的台(1或2)

WQ(t)：在時間t，等待過磅的卡車的台

W(t)：在時間t，正在過磅的卡車的台。

Event notices 事件通知

(ALQ, t, DTi)：在時間t，卡車DTi到達裝載等候佇

(EL, t, DTi)：在時間t，卡車DTi結束裝載。

(EW, t, DTi)：在時間t，卡車DTi結束過磅。

Entities 實體：六輛卡車 (DT1, DT2, ..., DT6)

Lists 表單

裝載等候伫，過磅等候伫

Activities 活動：裝載時間，過磅時間，旅行時間

Delays 延後：在裝載機器的等候，在過磅時的等

模擬結果顯示於 Table 3.6，假設起始狀態為5輛卡車在裝載，1輛卡車在過磅。

為了估計裝載機器與磅秤的使用率，必須蒐集兩累積統計量，B_L與B_S。

B_L：從時間0至時間t，兩裝載機器的所有忙碌時間。

B_S：從時間0至時間t，磅秤的所有忙碌時間。

既然從時間0至20，兩裝載機器均是忙碌狀態，所以在 $t=20$，$BL=40$。從時間20～24，只有一部裝載機器是忙碌，於是在這段時間 BL 只增加4，而成為44。在時間25～26間（$L(25=0$），兩部裝載機器均閒置，所以 BL 維持不變。以部分模擬結果，模擬時間僅76分鐘。

$$\text{平均裝載機器使用率} = \frac{49/2}{76} = 0.32$$

$$\text{平均磅秤使用率} = \frac{26}{76} = 1.00$$

這些估計值不能視為長期穩定狀態的精確估計值，更長的模擬時間需要用以減少時間0的假設（有5部卡車在裝載機器）。如果分析者對系統中短期的轉移行為有興趣，只短時間的模擬可用以作代表。

Table 3.6. Simulation Table for Dump Truck Operation (Example 3.5)

	System State				Lists			Cumulative Statistics	
Clock					Loader	Weigh	Future Event		
t	$LQ(t)$	$L(t)$	$WQ(t)$	$W(t)$	Queue	Queue	List	B_L	B_S
0	3	2	0	1	$DT4$		$(EL, 5, DT3)$	0	0
					$DT5$		$(EL, 10, DT2)$		
					$DT6$		$(EW, 12, DT1)$		
5	2	2	1	1	$DT5$	$DT3$	$(EL, 10, DT2)$	10	5
					$DT6$		$(EL, 5+5, DT4)$		
							$(EW, 12, DT1)$		
10	1	2	2	1	$DT6$	$DT3$	$(EL, 10, DT4)$	20	10
						$DT2$	$(EW, 12, DT1)$		
							$(EL, 10+10, DT5)$		
10	0	2	3	1		$DT3$	$(EW, 12, DT1)$	20	10
						$DT2$	$(EL, 20, DT5)$		
						$DT4$	$(EL, 10+15, DT6)$		
12	0	2	2	1		$DT2$	$(EL, 20, DT5)$	24	12
						$DT4$	$(EW, 12+12, DT3)$		
							$(EL, 25, DT6)$		
							$(ALQ, 12+60, DT1)$		
20	0	1	3	1		$DT2$	$(EW, 24, DT3)$	40	20
						$DT4$	$(EL, 25, DT6)$		
						$DT5$	$(ALQ, 72, DT1)$		
24	0	1	2	1		$DT4$	$(EL, 25, DT6)$	44	24
						$DT5$	$(EW, 24+12, DT2)$		
							$(ALQ, 72, DT1)$		
							$(ALQ, 24+100, DT3)$		
25	0	0	3	1		$DT4$	$(EW, 36, DT2)$	45	25
						$DT5$	$(ALQ, 72, DT1)$		
						$DT6$	$(ALQ, 124, DT3)$		
36	0	0	2	1		$DT5$	$(EW, 36+16, DT4)$	45	36
						$DT6$	$(ALQ, 72, DT1)$		
							$(ALQ, 36+40, DT2)$		
							$(ALQ, 124, DT3)$		
52	0	0	1	1		$DT6$	$(EW, 52+12, DT5)$	45	52
							$(ALQ, 72, DT1)$		
							$(ALQ, 76, DT2)$		
							$(ALQ, 52+40, DT4)$		
							$(ALQ, 124, DT3)$		

Table 3.6. Continued

	System State				Lists			Cumulative Statistics	
Clock					Loader	Weigh	Future Event		
t	$LQ(t)$	$L(t)$	$WQ(t)$	$W(t)$	Queue	Queue	List	B_L	B_S
64	0	0	0	1			$(ALQ, 72, DT1)$	45	64
							$(ALQ, 76, DT2)$		
							$(EW, 64+16, DT6)$		
							$(ALQ, 92, DT4)$		
							$(ALQ, 124, DT3)$		
							$(ALQ, 64+80, DT5)$		
72	0	1	0	1			$(ALQ, 76, DT2)$	45	72
							$(EW, 80, DT6)$		
							$(EL, 72+10, DT1)$		
							$(ALQ, 92, DT4)$		
							$(ALQ, 124, DT3)$		
							$(ALQ, 144, DT5)$		
76	0	2	0	1			$(EW, 80, DT6)$	49	76
							$(EL, 82, DT1)$		
							$(EL, 76+10, DT2)$		
							$(ALQ, 92, DT4)$		
							$(ALQ, 124, DT3)$		
							$(ALQ, 144, DT5)$		