運用 UML 在軟體發展專案管理之研究與探討

The application of Unified Modeling Language(UML) in project management

林至中/葉春秀
朝陽科技大學資訊管理系
E-mail:jjlin/s8714602@mail.cyut.edu.tw

摘　要

此篇文章是擴展 UML(Unified Modeling Language)以及利用由上而下的分解技術以應用在軟體發展專案管理的控制與管理之上。UML 包含了各種的模式工具，並提供了一個完整的分析與標準的溝通界面，我們採用了當中的動作圖(Activity Diagram)、循序圖(Sequence Diagram)及類別圖(Class Diagram)以分別表達專案活動之間的關係、使用到的元件之間的關係、以及建構這些活動與元件的定義　然而，除了這些 UML 可以表達的關係外我們認為應該還須表達出活動的事件進行與其所使用的元件之間的關係，才能將軟體發展的行為完整的表達出來。另外，也應該增加控制符號來呈現出有關專案中的活動的執行條件/結果。我們的模型也特別的利用物件導向的階層架構經由由上而下的分解技術以提供不同階層的專案管理人員有關專案進度之資訊。這個模型可以提供專案活動在執行時的及時資訊，以便讓專案管理人員能夠對活動之排程做適當的調整。

關鍵字：專案管理、由上而下的分解技術、物件導向 UML，資訊系統發展。

1.前言

傳統專案管理的模型(Gantt、CPM、以及 PERT)[1-3]較著重在專案活動的時間排程控制上，故有許多無法表達的缺點：(1)他們沒有提供專案管理人員可瞭解的資訊及訂立專案活動的程序；(2)他們沒有呈現活動適當的階層架構以及他們所包含的子活動；(3)他們沒有指出一個活動的再執行而對於活動重新排程的情況；以及(4)他們沒有提供一個活動開始執行所需要被觸發的條件。而針對傳統模型之不足，陸續有許多模型的提出以改善傳統模型所無法表達的缺點，如使用 Petri net[4-6]、UCLA 模型[7]，以提供專案活動的瞭解度及訂立專案的活動進度，或DesignNet、PM-Net 模型[10,11]以區別專案活動中不同種類的元件的存取狀況（如資源、產品、或狀態報告）及活動之間可能的階層架構。尤其是 PM-net[10-11]利用 DFD 由上而下的分解技術[12-13]，將活動由上而下的分解出相關的子活動直到最底層[8-9]；最底層的執行狀況再由下而上的將活動本身及其存取元件的資訊由下傳到上層。

在 80 年代至 90 年代中，物件導向已蔚為風潮。更由於物件導向有易維護性、再使用性與擴充性的特色，也逐漸被工商業界大量使用在系統分析與設計上，並有取代傳統結構化分析方法的趨勢，例如UML[14,22,24-25]。在相關的文獻探討中，專家仍希望UML 能達成下列的主要目標[25]：(1)提供系統分析及設計人員一種易於表達，方便使用的視覺化模型建立語言，使其彼此間可以發展及交換有意義的模型；(2)系統分析及設計人員可以根據所發展系統的特殊需要來定義及擴充其觀念；(3)不針對某種特別程式語言；(4)讓系統分析及設計人員能清楚了解此種模型建構語言；(5)促進物件導向工具市場的發展；(6)提供較高階的系統發展觀念，諸如，互助(Collaborations)，框架(Frameworks)，模型(Patterns)，及元件(Components)，以利於系統模組的再利用。

UML 是由 Rational Software 整合了現今物件導向系統分析及設計的主要三大理論〔15-21〕，融合而成為一個統一化的語言。UML 是一種模式語言，但並不是一種設計的方法。模式語言是一種用來把設計表達出來的表示式（大部份是採用圖形的方法來表達），而程序是指在設計過程中這個方法所建議採用的進行步驟〔14〕。UML 已提供了許多模式技術的方法，不同的模式所表現出來的資訊也不盡相同，如類別圖(Class Diagram)定義出系統所要使用的物件；循序圖(Sequence Diagram)表示

出物件之間訊息的傳遞；動作圖(Activity Diagram)表示出動作進行的順序。在本文中，我們即利用三種圖形分別表達專案中的活動之間的關係、使用到的元件之間的關係、以及建構這些活動與元件的定義。故本研究主要是以 UML 現有的模型架構，再加上三種的控制符號（AND、OR、XOR）及物件動作圖（Object-Activity Diagram）來完整的表示出專案活動的進行。

2. 介紹

為了能夠成功地完成專案活動的發展，適當的對資訊系統監督與控制是主要的關鍵。為了達到上述目標，適當的模型對於描述專案發展的程序已經有很好的影響，並可以提供管理人員更多必要性的資訊，讓專案管理人員可以檢視其活動的狀況及專案的進度。雖然現在已經有存在一些不完整的模型特性來提供專案管理人員 做有效率的管理，我們仍認為一個好的模型應該具有下列的特性：

(1) 一個模型應該能夠提供不同階層的管理者在適當的階層上提供他們相關的專案進度，以達到管理的目的。

(2) 一個模型應該能夠呈現專案執行的進度以及專案是否在訂立的時間排程內完成其活動。

(3) 一個模型應該能夠彈性的支援軟體發展程序之動態的改變(如新增、刪除、重覆其專案活動)，以便指出在設計程序上正確的發展。

(4) 一個模型應該指出在發展程序上任何動態改變的影響(如重新執行、中斷或放棄活動等)。

(5) 一個模型應該能夠充份的讓專案管理人員更容易去瞭解及監督其發展的程序。

我們相信物件導向模型的許多特性，例如繼承（generalization- specialization），包含(whole-part)，以及物件的封裝(encapsulation)，可以達到以上所述的觀點。為了證明此項觀點，我們利用了UML模型，來提供對於軟體發展程序的管理；其中，類別圖用來描述發展過程中所用到的相關元件的型態及型態之間結合的關係；循序圖用來描述這些元件之間的關係以及訊息傳送的情形；動作圖用來描述軟體發展活動之間的關係。另外，我們的模型也特別的利用由上而下的分解技術以及物件導向的generalization-specialization、whole-part特性來提供不同階層有關專案活動及其使用的元件的資訊；下層的活動/元件的狀況推導出上層的活動/元件的狀況。

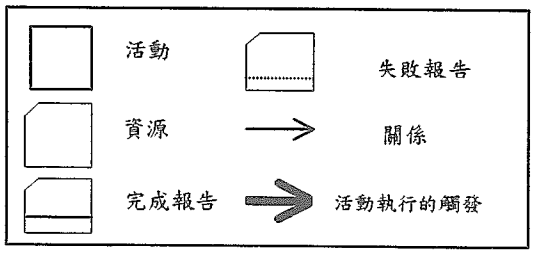
3. UML 的應用

此章節中我們將UML應用在專案管理的活動上，並延伸了UML的符號定義，用更清楚的符號表示來說明專案活動的進行。一般來說，軟體專案發展(如圖二 A)的程序包含了：需求分析、初步設計、細部設計、撰寫程式及測試等活動。我們將利用需求分析(以下簡稱 RA)以及初步設計(以下簡稱 PD)相關的活動/子活動來做案例描述(如圖二 B)。活動的類別描述在本文中都有詳細圖形定義說明(如圖一)。以下我們將針對UML中的類別圖、動作圖、循序圖以及物件動作圖來專案活動為案例來做說明。

3.1 類別圖

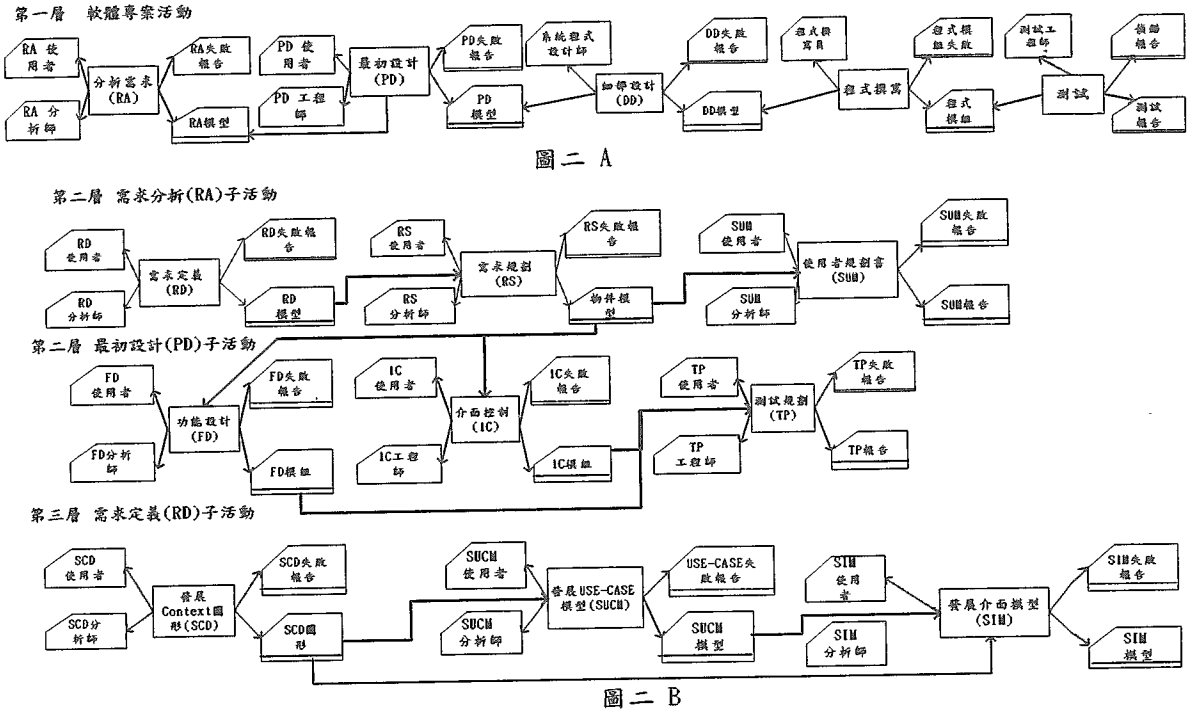
類別是為了定義系統物件的抽象化表示。類別圖是用來描述系統中所需的相關類別以及類別之間彼此的 靜態關係，靜態關係包含了類別之間的關聯性以及繼承與包含的關係。關聯是指類別之間彼此的結構關聯（如一對多，多對多）。在圖二中類別之間的關聯皆為一對一的關係。

圖二所呈現的是在專案中相關的類別彼此的關係，這其中的類別型態可包含：資源、完成報告、失敗報告等(如圖一的定義)。在圖二A所表示為整個軟體專案所需執行的相關活動步驟，在圖二B及圖二C中，定義RA活動及 PD 活動相關資源/報告的類別與各類別之間的關係。從圖二B中，可看出活動及其組成的子活動之間的



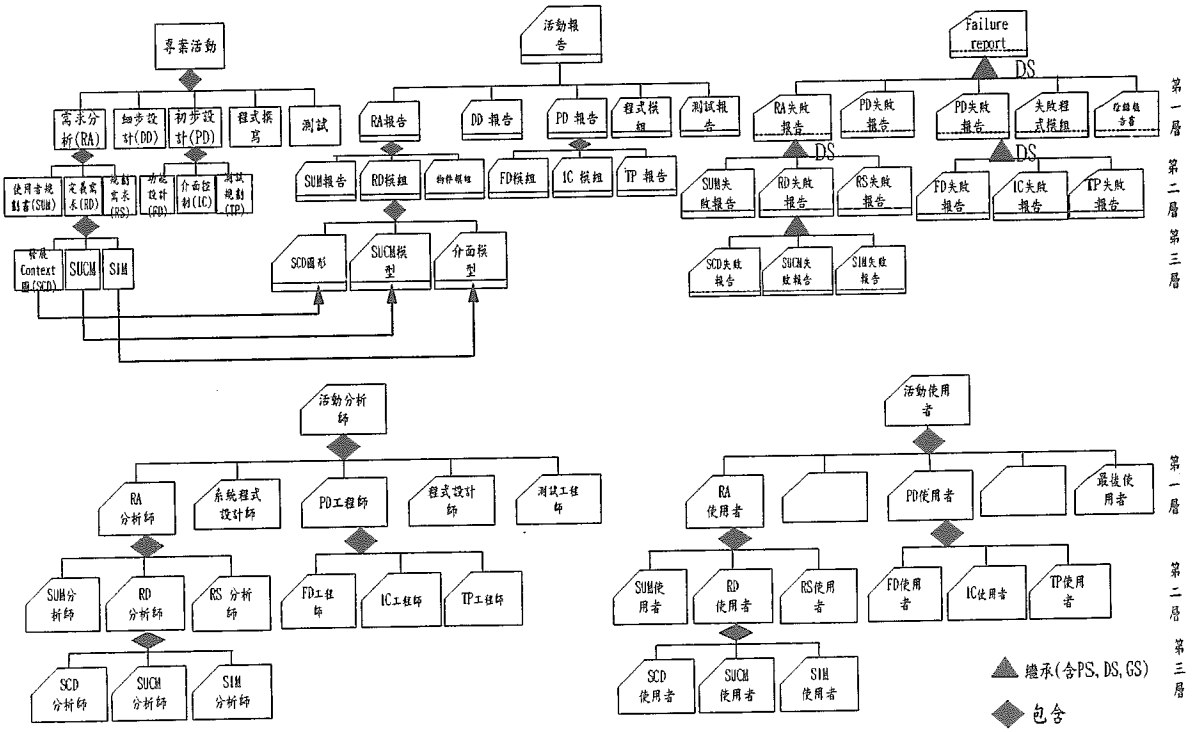
圖一 類別圖 圖形定義

階層關係，也可以瞭解各(子)活動與其相關的資源/報告之間的互動情形。因此，我們可以利用由上而下的分

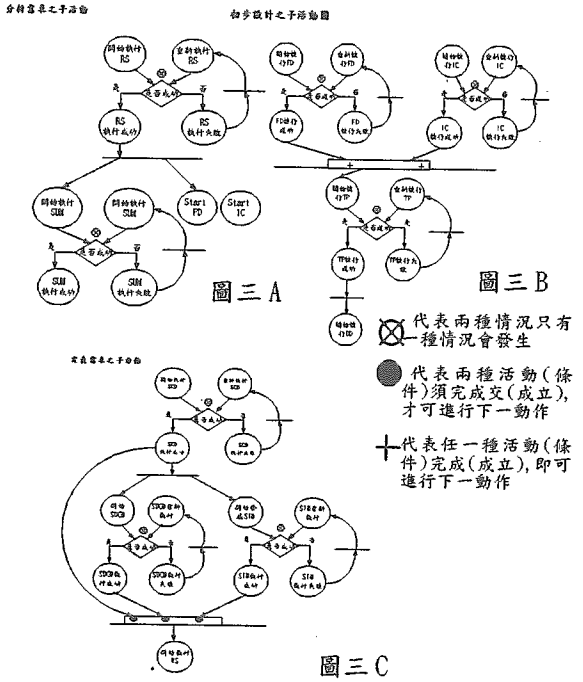

第一層 軟體專案活動
RA 使用者
RA 分析師
分析需求(RA)
RA失敗報告
RA模型
PD 使用者
PD 工程師
最初設計(PD)
PD失敗報告
PD模型
系統程式設計師
細部設計(DD)
DD失敗報告
DD模型
程式撰寫員
程式撰寫
程式模組失敗
程式模組
測試工程師
測試
偵錯報告
測試報告
第二層 需求分析(RA)子活動
RD使用者
RD分析師
需求定義(RD)
RD失敗報告
RD模型
RS使用者
RS分析師
需求規劃(RS)
RS失敗報告
物件模型
SUM使用者
SUM分析師
使用者規劃書(SUM)
SUM失敗報告
SUM報告
第二層 最初設計(PD)子活動
FD使用者
FD分析師
功能設計(FD)
FD失敗報告
FD模組
IC使用者
IC工程師
介面控制(IC)
IC失敗報告
IC模組
TP使用者
TP工程師
測試規劃(TP)
TP失敗報告
TP報告
第三層 需求定義(RD)子活動
SCD使用者
SCD分析師
發展Context圖形(SCD)
SCD失敗報告
SCD圖形
SUCM使用者
SUCM分析師
發展USE-CASE模型(SUCM)
USE-CASE失敗報告
SUCM模型
SIM使用者
SIM分析師
發展介面模型(SIM)
SIM失敗報告
SIM模型

圖二 A

圖二 B


專案活動
需求分析(RA)
細步設計(DD)
初步設計(PD)
程式撰寫
測試
使用者規劃書(SUM)
需求定義(RD)
需求規劃(RS)
功能設計(FD)
介面控制(IC)
測試規劃(TP)
發展Context圖形(SCD)
SUCM
SIM
活動報告
RA報告
DD 報告
PD 報告
程式模組
測試報告
SUM報告
RD模組
物件模組
FD模組
IC 模組
TP 報告
SCD圖形
SUCM模型
介面模型
Failure report
DS
RA失敗報告
PD失敗報告
失敗程式模組
SUM失敗報告
RD失敗報告
RS失敗報告
FD失敗報告
IC失敗報告
TP失敗報告
SCD失敗報告
SUCM失敗報告
SIM失敗報告
第一層
第二層
第三層
活動分析師
系統程式設計師
PD工程師
程式設計師
測試工程師
RA 分析師
SUM分析師
RD 分析師
RS 分析師
FD工程師
IC工程師
TP工程師
SCD 分析師
SUCM 分析師
SIM 分析師
活動使用者
RA 使用者
PD使用者
最後使用者
SUM使用者
RD 使用者
RS使用者
FD使用者
IC使用者
TP使用者
SCD 使用者
SUCM 使用者
SIM 使用者
繼承(含PS、DS、GS)
包含

解，將下層活動的進度與狀態往上呈現到上一層次的活動。至於從圖二C中，則可以瞭解各類別在不同階層所形成的分類/組成架構。因此，從第一層的類別對應到圖二B中的第一層類別，我們可以清楚知道RA動作及PD動作所需要的資源。及其執行時將會產生的文件（成功或失敗）；同樣的，圖二C中的第二層與第三層的類別為對應到圖二B中的第二層與第三層的類別；特別在圖二C中，RA失敗報告有一DS (Disjoint Subtype)的表示，其所代表的意思是RA失敗報告的產生只有可能是因為RD失敗報告或是RS失敗報告或是SUM失敗報告的產生而造成；PD失敗報告的產生只有可能是以下三種情況之一：FD失敗報告的



圖三A　圖三B　圖三C

產生或是IC失敗報告的產生或是TP失敗報告的產生所造成。這主要是回應一個真實的情況；只要某一組成的子活動失敗，則上一層中的活動也進入失敗狀況，而產生一個失敗報告。因此，我們利用下一層次的動作所造成的資源/報告的狀態，而得到上一層次資源/報告的執行狀態。

在圖二B與圖二C的例子中，利用三層的類別定義來表達分析需求(以下簡稱RA)以及初步設計(以下簡稱PD)的活動：第一層為軟體專案活動RA以及PD相關活動；第二層為完成RA以及PD所需進行的子活動，包含：發展定義需求(以下簡稱RD)、發展規格需求(以下簡稱RS)、發展使用者手冊(以下簡稱SUM)、功能設計(以下簡稱FD)、介面控制(以下簡稱IC)、測試規劃(以下簡稱TP)；第三層為完成RD所需進行的子活動，分為發展Context圖形(以下簡稱SCD)，發展USE-CASE模型(以下簡稱SUCM)，以及發展介面模型(以下簡稱SIM)。而在第二層RD所完成的報告（成功的報告）之輸出粗線所代表的是必須等RD之動作完成之後才能開始進行RS以及SUM的進行；也就是說，必須等到RD在第三層的子活動(SCD圖形、SUCM模型、SIM模型)完成後，才能進行第二層的後續活動。由此表示出其活動執行的優先順序。

3.2 動作圖

圖四 循序圖

動作圖可算是UML中的一項創舉，它結合了Petri-net、Odell的事件圖、以及SDL狀態式等相關的技術。動作圖可充分將不同動作之間的流程與相互關係詳細描述

出來，並將類別定義間的動作執行情形表達出來。在圖中我們增加了UML所沒有的三個控制符號來強化執行的控制，包含：AND(●)，表示輸入的條件皆須要全部完成才能繼續執行下一個動作（條件）；OR(+)，表示輸入的條件只要有任何一個條件成立即可繼續執行下一個動作；XOR(⊗)，代表兩種情形只有一個條件會成立。透過這三個控制符號，除了可以很瞭解到動作的執行情形，更可清楚的知道動作的執行條件為何。圖三是表達出圖二 B 中的分析需求(RA)、初步設計(PD) 、需求定義（RD）子活動的運作情形；以圖三 A 為例，對應到圖二 B 中 RA 的子活動包括 RD、RS 、以及 SUM 等，其中 RD 因另有三個子活動 SCD、SUCM、SIM 而描述在圖三 C 中，以圖三A的動作圖中可知道必須先成功完成 RD 的動作後，才往下再繼續執行 RS，若是 RS 執行失敗後，RS 將會啟動重新執行的動作，直到 RS 成功完成。由圖三 A 及圖三 C 中可知，RD 的三個子活動必須先執行完成，才能將執行移到下一個活動 RS。而 RS 完成後，可開始 PD 相關子活動的進行，其動作進行的優先順序為先成功完成 FD 以及 IC 活動後，才能進行 TP 動作的進行（如圖三B）。

由動作圖中，除了可看出類別的定義之後更詳細的動作進行，並可將動作優先順序明確的表達出來，以及動作的控制條件很明確的表達。

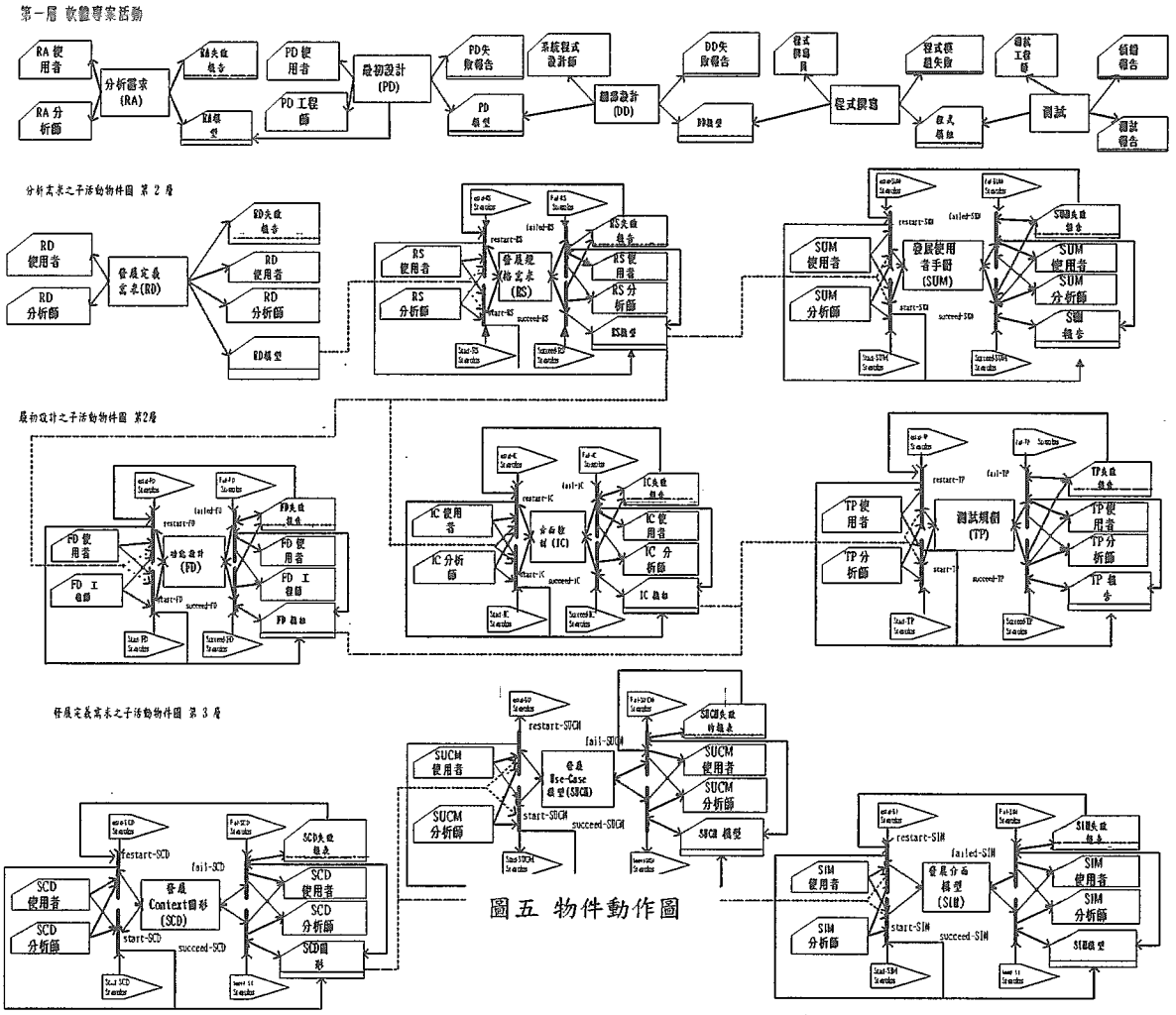
3.3 循序圖

循序圖是透過垂直的生命線(Lifeline)來表達出物件彼此間的訊息傳達。在 UML 中，循序圖是利用一個方格子（□）來表示物件，但我們認為這樣不足以充分表示出物件的定義。故我們將物件的表示區分為：活動、資源、完成報表、失敗報表等，除了以實線箭號來表示訊息傳遞以及傳遞所具備的條件，虛線代表訊息的回傳，以及一個循環符號(*)來表示出訊息會重覆地被接收，並在訊息的傳達也加上了 AND、OR 、以及 XOR 的符號控制。在透過循序圖的表示下，可瞭解物件之間彼此的互動情形，以及執行的順序，訊息傳遞即可視為物件的方法呼叫，並把類別圖相關的類別視為橫軸上的物件型態。以圖四為例，在第二層 RS 活動的循序圖中，RS 活動需使用到 RS 使用者以及 RS 分析師還有 RD 的完成報告，在圖中可看到有一"AND"(●)的符號，表示 RS 使用者以及 RS 分析師與 RD 的完成報告是執行 RS 活動的必要條件，故除了需先將相關的資源找齊後，也須 RD 相關的動作完成才能執行 RS 的活動，而 RS 右邊的兩個報告分別是失敗的 RS 報告以及動作成功後所產生之物件模型報告。由符號的表示，可以瞭解這兩個報告只有一種情形會存在，不會同時產生失敗的報告以及物件模型。當產生物件模型報告後，代表 RS 活動已完成，並且往後會有 SUM 活動以及 FD 活動會使用到 RS 活動的物件模型。

3.4 物件動作圖

由先前各個模式圖形我們可得知，類別圖是定義系統所要使用的物件；也就是說，是定義專案活動及相關的存取元件，且將這些活動及元件之間的關係表達出來；循序圖則是根據類別圖的定義而將出物件之間的關 係表達出來；也就是說，循序圖所表達的是專案發展過程中，所使用到的活動/元件之間的關係表達出來。動作圖則是將物件的動作之間的順序表示出來；也就是說，將專案發展的活動之間的表示出來。但對於整個專案的執行來做分析，專案發展對於活動的事件進行與其所使用的元件之間的關係則無法表達出來，故專案的管理不僅需要以上三種圖形，還須包括活動的事件與一個元件之間的關係圖才能將整體的專案程序完整的表達出來，故我們根據動作圖與循序圖的分析畫出物件動作圖。在圖五中我們可以更清楚的瞭解專案元件與活動的進行之間的關係。比較特別的是，在物件動作圖的最後一層，我們使用了 Petri-net 來表達活動可能會發生的事件與其它相關物件的存取關係。至於其他階層，由於上一層的活動執行結果完全是由下一層的子活動的執行狀況來決定，所以上一層並不需要Petri-net 來表示其活動之事件進行，而只需呈現出目前活動是由下一層所推導出來的執行狀態。因此，如果上一層的活動是由許多子活動所組成，我們就只將此上一層的活動的相關物件畫出。

依據在圖五的最後一層中所顯示，要完成 SCD 活動必須先將所需之資源：SCD 使用者以及 SCD 分析師找齊之後，才能啟動 SCD 的動作（start-SCD 事件的發生），此一活動被啟動後就會產生一個 SCD 圖形物件來等待成



圖五 物件動作圖

功訊息的傳達；若活動成功完成，則此一 SCD 圖形的狀態即改為成功的狀態(Succeed-SCD 事件的發生)。若是執行結果為失敗的動作，則會觸發 fail-SCD 來產生一個 SCD 失敗報告。專案管理人員可依據此失敗報告檢討其失敗原因，並重新再觸發執行的動作(restart-SCD)。

若以第一層的 RA 而言，要完成此項動作所需要的資源分別是 RA 使用者以及 RA 分析師;若執行此 RA 後，可能產生的報告包括失敗報告；或是活動執行成功的 RA 報告。而在執行完成後，所再產生的 RA 使用者以及 RA 分析師代表是活動執行後，資源可重新的再被使用。甚至一個使用者或是分析師可能會同時扮演兩種不同的角色。

在 SCD 中，要完成這項動作必須先將所需之資源:SCD 使用者以及 SCD 分析師找齊之後，才能啟動規格說明的動作（start-SCD），當一 transition 被啟動後，就會產生一個需求說明的成功物件來等待成功訊息的傳達；若是執行結果為失敗的動作，則會觸發 failed-SCD（transition）來產生一個失敗的 SCD 報告，並將此活動再檢討其失敗之原因，重新再觸發執行的動作 restart-SCD，並原先等待成功訊息的報表將會呈現出其失敗。若以第一層的分析需求而言，要完成此項動作所需要的資源分別是 RA 使用者以及 RA 分析師;若執行失敗後，可能產生的失敗的報告或是活動執行成功所產生的成功的分析報告。而在執行完成後，所再產生的 RA 使用者以及 RA 分析師代表是活動執行後，資源可重新的再被使用，甚至一個使用者或是分析師可能會同時扮演兩種不同的角色。

4. 分析與討論

利用 UML 來分析軟體專案的進行時相關的類別與活動甚至是物件與方法之間的溝通方

式是一個很明確且清楚的模式標準語言組，UML 集合各物件導向大師之大成，不僅可幫助使用者能夠一步一步地找出分析的目標，例如 Use-Case，Scenario 等，都是在幫助系統分析師能夠更直接地找出所需的物件。但

在推行之餘我們仍提出幾點我們認為UML所不足之處：

(1) 如何能夠更明確地表達出活動進行的階層關係，我們發現利用UML在專案活動的分析上，對於同一階層的相關活動之間的先後關係我們可以很清楚地分析出專案活動與其存取元件之間的關係，但是對於上下階層銜接，較不能有效地完整表達出來。

(2) 對於活動執行的條件規範，UML較無法呈現出其他特殊的條件，如OR、XOR等。

(3) 對於活動進行狀況的表達，UML並沒有一個模式可以將軟體專案的進行整體的表達出來，如元件對活動事件之間的關係圖。

5. 結論

我們將物件導向模型以及UML的技術應用在專案的管理與控制上。這是利用了物件導向的特性，來定義以及加強UML模型，以提供軟體發展程序一個階層的表示。這幾個好處幫助專案管理人員去瞭解以及對程序的描述，更重要的是去獲得及監督他們所想要知道的程序狀況及進度。

物件導向已經是一個很重要的觀念，並已經愈來愈受到電腦軟體工業的注意。物件導向技術的特性可以讓軟體系統很容易被瞭解、維護以及再使用。一個軟體發展程序是一個設計的程序，並且是一個展開的程序，對於專案模型與控制上，使用物件導向的技術相信是一個合理的方法。我們的模型也致力於在這項工作的表現。現在一個專案管理的環境都建立在較完整的模型上，這也包含了符合本文在軟體發展程序的設計上圖形的表示。

最後，本論文之發展是由國科畫補計之計畫(NSC88-2213-E-324-006).

參考文獻

[1] E. Davis, Ed., *Project Management: Techniques, Applications, and Managerial Issues*, Industrial Engineering and Management Press, 1983.

[2] E. Horowitz and S. Sahni, *Fundamentals of Data Structures in Pascal*, Computer Science Press, 1985.

[3] J. Wiest and F. Levy, *A Management Guide to PERT/CPM*, Prentice-Hall, 1977.

[4] J. Peterson, "Petri Nets", ACM Computer Surveys, Vol. 9, No. 3, Sep. 1977, pp. 223-252.

[5] J. Peterson, *Petri Net Theory and The Modeling of Systems*, Prentice-Hall, 1981.

[6] E. Yiannis and L. Thomas, "Specification and Analysis of Parallel/ Distributed Software and Systems by Petri Nets with Transition Enabling Function", *IEEE Transaction on Software Engineering*, Vol. 18, No. 3, March 1992, pp. 252-261.

[7] V. Cerf, "Multiprocessors, Semaphores and A Graph Model of Computation", *Ph. D. Dissertation*, Computer Science Department, UCLA, 1972.

[8] L. Belady and C. Evangelisti, "System Partitioning and Its Measure", *The Journal of Systems and Software*, Vol. 2, 1981, pp. 23 -29.

[9] Y. Wand and R. Weber, "A Model of Systems Decomposition", *Proc. Of Int'l Conf. on Information Systems*, Boston, Dec. 4-6, 1989, pp. 41-51.

[10] L. Liu and E. Horowitz, "A Formal Model for Software Project Management", *IEEE Transaction on Software Engineering* Vol. 15, No. 10, Oct. 1989, pp. 1280-1293.

[11] K. Lee and I. Lu, "PM-Net: A Software Project Management Representation Model", *The Journal of Information and Software Technology*, Vol. 36, No. 5, 1994, pp. 295-308.

[12] T. DeMarco, *Structured Analysis and System Specification*, Yourdon Press, 1978.

[13] C. Gane and T. Sarson, *Structured Systems Analysis: Tools and Techniques*, Prentice -Hall, 1976.

[14] Martin Fowler with Kendall Scott " UML Distilled :applying the standard object

modeling language" , ADDISION-WESLEY, 3 rd Printing, September. 1997

[15] Chung, K. H., and et. Al. An Object-oriented Simulation Model of Job-shop production scheduling system, Proceeding of 7th Workshop on Object-oriented Technologies and Application, pp108-115, . Tainan, Taiwan, 1996.

[16] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy & William Lorensen, "Object-Oriented Modeling and Design", Prentice-Hall, Inc., 1991.

[17] Grady Booch, James Rumbaugh & Ivar Jacobson, "UML Notation", Santa Clara, CA: Rational Software Corporation, 1997.

[18] Grady Booch, James Rumbaugh & Ivar Jacobson, "UML Semantics", Santa Clara, CA : Rational Software Corporation, 1997.

[19] Grady Booch, James Rumbaugh & Ivar Jacobson, "UML Summary", Santa Clara, CA : Rational Software Corporation, 1997.

[20] Peter Hruschka, "UML: A Course or a Blessing for the OO Community?", American Programmer, March 1997.

[21] Rational Software Corporation, "Unified Modeling Language For Real-Time System Design version 2.0", Santa Clara, CA:Rational Software Corporation.

[22] 物澤 OT 中心 OOAD 組, "UML 發展的背景分析", 物件導向雜誌(JOF), 1996.8~9, p99~105.

[23] 邱郁惠, "UML 何物耶?", PC Magazine 中文版, April 6 1997, p265~270.

[24] 鍾克雄, 高嘉鴻, 林耀欽, 盧志山, 曾崇城, "The Application of Unified Modeling Language(UML) in Real Time System".

[25] 火昌華 , "UML 的興起及市場現況", RUN PC, 第 48 期, July, 1998