第二章　文獻探討

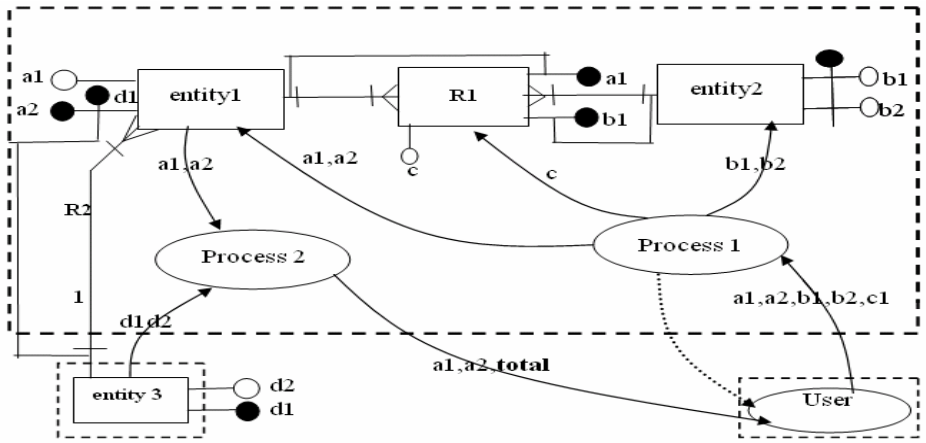
2.1 功能點分析法要項(terms)之呈現方式

功能點分析法未調整功能點估算所需要項包含了靜態的檔案(data)概念與動態的交易(transaction)概念。透過實體關係圖表達功能點分析法中靜態檔案概念的呈現，如資料項(Data Element Type,DET)、紀錄(Record Element Type,RET)、邏輯檔案(Logical Files)。

而以資料流程圖表達功能點分析法中動態交易(transaction)概念的呈現，如外部輸入(External input,EI)、外部輸出(External Output,EO)、外部查詢(External inquiry,EQ)。

為了能夠讓抽象之概念具體化，故[2]擷取實體關係圖與資料流程圖之部分元素並進行整合，做為未調整功能點要項之呈現，此種整合兩者圖示而成之新的設計規格圖示稱為 ER-DFD。

透過 ER-DFD 設計規格進行功能點估算前，必須先瞭解 ER-DFD 如何呈現未調整功能點之估算要項。



資料來源：[2]

圖 2-1 ER-DFD 範例

如圖 2-1 為 ER-DFD 之範例，ER-DFD 引入實體關係圖之實體(entity)、關聯基數(relationship cardinalities)、屬性(attribute)、主鍵(primary key)、外部鍵(foreign key)等概念，引入資料流程圖程序(process)、資料流(dataflow)及系統界線(system boundary)等概念。其中：

1. 實體(entity)：以實線方形表示。如 entity 1、entity 2 及 entity 3。
2. 關聯(relationship)：若關聯基數為 1 對 1、1 對多，則以線段表示，線段連接實體的兩端對於關聯基數不同而有不同之表示。若關聯基數為多對多，則需轉化為兩個 1 對多的關聯，而連接此兩個 1 對多之關聯稱為關聯實體(associative entity，如圖 2-1 的 R1)，視同實體並以實線方形表示。
3. 屬性(attribute)：鍵值(key)屬性以黑色圓形加線段表示。非鍵值屬性則以白色圓形加線段表示。複合鍵值(如 b1、b2 兩者形成一鍵值)，則以一鍵值屬性跨過非鍵值屬性表示。而外部鍵(foreign key)則從具有鍵值之實體延伸出一線段到外部鍵所在實體之外部鍵屬性(如 R1 之 a1 屬性與 b1 屬性均為外部鍵)。

4. 程序(process)：以實線橢圓形表示。如 process 1、process 2、user 等。

5. 系統界線(system boundary)：以虛線方形表示。系統界線區隔出欲估算之系統、外部系統及使用者。系統界線內稱為內部系統，系統界線外則稱為外部系統。

(1) 外部系統：提供資料至內部系統以利程序之進行，如 process 2 參考 entity 3 之 d1,d2 屬性。

(2) 使用者(user)：以橢圓形表示。使用者泛指操作系統的人[3]，在 ER-DFD 中以程序表示，如 user 傳遞 a1,a2,b1,b2,c1 屬性給 process1。

6. 資料流(dataflow)：本研究於 ER-DFD 圖示上將資料流區分為兩類：

(1)一般資料流(data flow)：以「實心線段加上箭號」表示，代表的執行該程序時所需傳遞之資料，稱為欄位(field)。每一個資料流均須標示出所傳遞欄位資訊，若是資料流上欄位的名稱與實體的屬性名稱相同時，表示兩者所指的是相同資料，當資料流上的屬性資訊並未對應到實體的任何屬性，則代表此欄位是經由計算所衍生出來的資訊，稱為衍生欄位(derived field)。如圖 2-1 的 total 欄位。

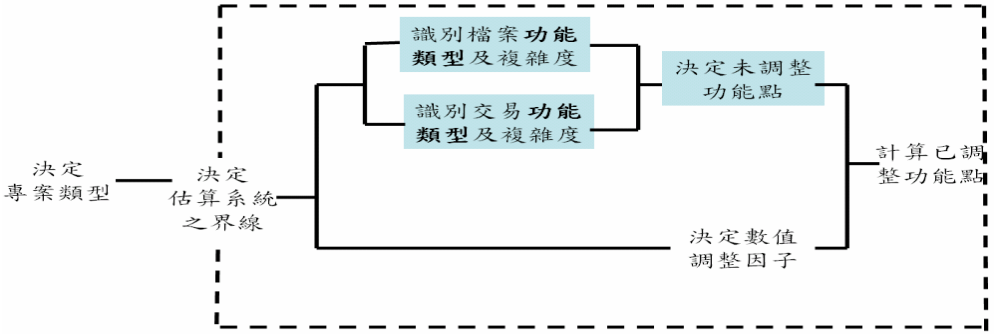
(2)系統回應資料流：以「虛線加上箭號」表示，代表傳遞內部系統回應(system response)訊息至外部，如在程序處理過程中發生錯誤的錯誤訊息(error message)、確認程序是否應該進行下一步處理或已處理完成的確認訊息(confirmation message)。或者由外部進入內部系統之觸發訊息。系統回應資

料流表示著一種訊號的傳遞，故該資料流上並無屬性資訊。

2.2 功能點分析法估算歷程

功能點分析法將使用者需求之功能與軟體系統開發所使用之技術分開估算，因此在需求分析階段早期即可進行使用者所需功能規模之估算[2]。

功能點分析法估算歷程如圖 2-2 所示，本研究將之歸納為六個階段：



資料來源：[3] part I p2-3

圖 2-2 功能點分析法估算歷程示意圖

一、決定功能點估算功能點專案對象類型：

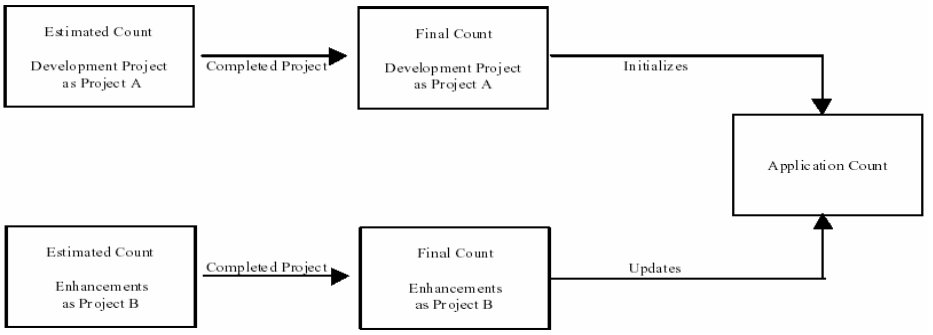
對象包括「新開發專案(development project)」、「欲調整專案(enhancement project)」及「現存應用系統(Application)」。這三種專案類型之估算內容差異如下：

1. 新開發專案：重新依據使用者需求重新開發一個符合使用者需求之軟體系統[14]。隨著專案開發階段運作，必須時時注意每一階段之軟體功能是否變更，並再次進行未調整功能點之估算，以確保各個估算要項與所需求之功能能夠一致。

2. 欲調整專案之計數：當專案將目前的軟體系統功能加以變更，所做的新增、修改、刪除之功能性部分[14]，強調估算所變更的功能。

3. 現存應用系統之計數：現存系統提供給使用者的功能之估算。另外，如圖 2-3

所示，當「新軟體專案開發完成」為最初的現存系統計數，當對此軟體系統產生調整需求時，即進行專案的調整(enhancement)，當欲調整專案完成時，該現存系統之功能點計數必須隨著功能的調整進行更新功能點計數，更新功能點計數後的系統，就取代之前現存系統之計數而產生新的現存系統計數。因此，現存系統之計數除了針對現有的系統進行功能點計數外，對於新開發完成的軟體系統或者已調整完成的軟體系統也屬於現存系統之計數。



資料來源：[3]

圖 2-3 新開發專案與欲調整專案對現存系統計數之關係

由於本研究針對已提出的實作系統之研究進行改良，故僅針對開發專案之類型為對象。

二、決定估算系統之系統界線(Application boundary)

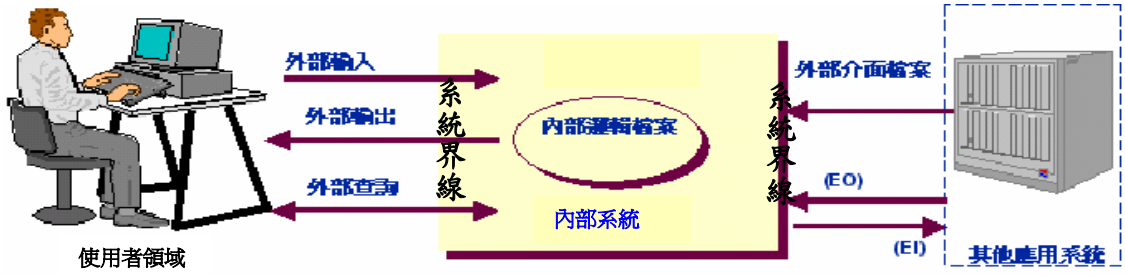
在確認應用程式界線之前首先必須決定計數的範圍(counting scope)，計數範圍可能涵蓋所採購的套裝軟體、也可能是包含所有要外包(outsourcing)的應用程式也可能僅是侷限於應用程式內執行某特定功能的模組(如人力資源系統的薪資計算模組)。而應用程式界線則是將計算範圍區隔出功能點所要計算的要項。若沒有定義出應用程式界線，則以下所探討之功能型態就無法被辨識。

本研究同[2]之假設，從新開發專案所給定的 ER-DFD 已經標明出系統界線。

應用程式界線區隔出「欲測量的系統(本研究稱為**內部系統**)」、「其它應用系統(本研究稱為**外部系統**)」或「使用者領域(user domain)」三者(如圖 2-4)。系統界線是區分功能點估算要項之重要依據。

三、識別系統所具備的「功能類型(Function Type)」

功能型態可分為兩種：「檔案功能類型(Data Function Type)」、「交易功能類型(Transactional Function Type)」。



資料來源：[14]

圖 2-4 系統所具備功能型態關係圖

1.「檔案功能類型(Data Function Type)」：檔案功能代表系統提供給使用者之功能必須要符合內部系統檔案需求及外部系統檔案需求[3]。如圖 2-4，檔案功能類型可區分成「內部邏輯檔案(Internal Logical Files)」及「外部介面檔案(External Interface Files)」。檔案(file)在內部系統內維護(maintain)者，稱之為「內部邏輯檔案」；檔案若是由外部系統維護者，稱之為「外部介面檔案」。檔案(file)指的是一組具有邏輯相關的資料(資料在 ER-DFD 稱為實體(entity))，並非一般所指的實體檔案[3]。識別應用系統所具備的檔案功能類型是屬於「內部邏輯檔案」、「外部介面檔案」之識別規則，請參考第 3 章。

本研究首先進行識別內部系統所具備的之檔案功能類型，下一步則是識別該檔案類型所使用到的「資料項(Data Element Type,DET)」個數及「記錄(Record

Element Type,RET)」個數，根據此兩者並對照表 2-1 即得出檔案功能類型之複雜度(complexity)，根據複雜度之等級對照表 2-2 即可得出該檔案功能類型之未調整功能點。

	1 to 19 DET	20 to 50 DET	51 or more DET
1 RET	Low	Low	Average
2 to 5 RET	Low	Average	High
6 or more RET	Average	High	High

表 2-1 ILF 及 EIF 的複雜度矩陣

	Function Points		
Components	Low	Average	High
ILF	7	10	15
EIF	5	7	10

表 2-2 ILF 及 EIF 功能點轉換表

2.「交易功能類型(Transactional Function Type)」：交易功能代表軟體系統提供給使用者依所需功能對不同檔案進行處理的能力[3]。

交易功能類型可區分成「外部輸入(External Input,EI)」、「外部查詢(External Inquiry,EQ)」及「外部輸出(External Output,EO)」。

識別應用系統所具備的交易(Transactional)類型是屬於「外部輸入」、「外部查詢」或「外部輸出」之識別規則，請參考第 3 章。

當交易類型確定後，下一步則是識別該交易使用的「資料項(Data Element Type)」個數」及「檔案類型參考(File Type Referenced)」個數，分別查表 2-3、表 2-4 即得出交易功能類型之複雜度，根據複雜度等級查表 2-5 即可得出該交易功能類型之未調整功能點。

	1 to 4 DET	5 to 15 DET	16 or more DET
0 to 1 FTR	Low	Low	Average
2 FTRs	Low	Average	High
3 or more FTRs	Average	High	High

表 2-3 外部輸入複雜度矩陣

	1 to 5 DET	6 to 19 DET	20 or more DET
0 to 1 FTR	Low	Low	Average
2 to 3 FTRs	Low	Average	High
4 or more FTRs	Average	High	High

表 2-4 外部查詢／外部輸出複雜度矩陣

複雜度等級 / 估算要項	低	中	高
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EQ	4	5	7
EO	3	4	6

表 2-5 各個估算要項之複雜度所對應之未調整功能點評比

四、計算未調整功能點(unadjusted function point)的總數

估算上述五種功能類型所使用的資料項數(Data Elements)、紀錄數(Record Elements)及檔案類型數參考數(File Type Reference)，即可得出相關複雜度，由複雜度即可得出對應之未調整功能點。將所有未調整功能點予以加總，得出內部系統之總未調整功能點，未調整功能點計算公式如表 2-6。

$$\boldsymbol{UFP=}\sum_{i\in Types}\sum_{j\in Complexity}N_{ij}W_{ij}$$

Types={ILF、EIF、EI、EO、EQ}，*Complexity*={Low、Average、High}。 N_{ij}：第 *i* 種類型所具備第 *j* 種複雜度之個數。 W_{ij}：第 *i* 種類型所具備第 *j* 種複雜度所對應之權重(即未調整功能點)。
表 2-6 未調整功能點計算公式

五、計算「數值調整因子(value adjustment factor,VAF)」

[3]歸納出了 14 項可能受到技術層面所影響之特性，稱為「一般系統特性(General System Characteristic,GSC)」(如附錄 1)，其目的是要對「未調整功能點」進行調整，得出最後「已調整功能點」。

$VAF=0.65+0.01\times TDI$ (1)0.65 及 0.01：常數值。(2)TDI：$\sum_{i=1}^{14} GSC_i$
表 2-7 數值調整因子計算公式

根據每一項特性均所賦予的相關的描述，給予 0 到 5 的評比，0 表沒有影響，5 表強烈影響，如表 2-8。來決定此特性之影響程度(Degree of Influence,DI)。將 14 項個別 DI 加總得到總影響程度(total DI,TDI)，VAF 即透過表 2-7 之公式所求得。一般而言，VAF 對未調整功能點之調整範圍可達±35%[5]。

等級	影響程度(Degree of influence)	意涵
0	No present，or no influence	沒有影響
1	Incidental influence	偶爾發生
2	Moderate influence	中等程度
3	Average influence	平均程度
4	Significant influence	顯著影響
5	Strong influence throughout	強烈影響
表 2-8 GSC 之影響程度		

六、計算「已調整功能點(adjusted function point)」

不同類型的軟體開發專案，有不同已調整功能點之計算方式。

1. 新開發專案計數：

下列公式適用於新開發軟體專案之已調整功能點計算：

DFP=(UFP+CFP)×VAF DFP：開發專案之功能點計數。UFP：未調整功能點計數。CFP：資料轉換功能之功能點計數。VAF：數值調整因子。
表 2-9 適用於新開發專案之已調整功能點數值之計算

2. 欲調整專案計數：

下列公式適用於欲調整專案之已調整功能點數值之計算：

EFP=[(ADD+CHGA+CFP)×VAFA]+(DEL×VAFB) EFP：欲調整專案之功能點數目。ADD：**新增功能**之未調整功能點數目。CHGA：**功能變更**後之未調整功能點數目。CFP：資料轉換功能之功能點數目。VAFA：調整後之數值調整因子。DEL：**功能刪除後**之未調整功能點數目。VAFB：調整之前之數值調整因子。
表 2-10 適用於計算欲調整專案之已調整功能點數值之計算

3. 現存應用系統計數：

下列公式適用於現存應用程式之已調整功能點數值之計算：

(1) 已開發完成之軟體專案計數 AFP=ADD×VAF AFP：現存應用程式之功能點數目。ADD：軟體開發完成之未調整功能點數。VAF：該軟體系統之數值調整因子。 (2) 已調整完成之軟體專案計數 AFP=[(UFPB+ADD+CHGA)-(CHGB+DEL)]×VAFA AFP：現存應用程式之功能點數目。UFPB：調整前之未調整功能點數。ADD：新增功能之未調整功能點數。CHGA：功能變更後之未調整功能點數。CHGB：功能變更前之未調整功能點數。DEL：功能刪除後之未調整功能點數。VAFA：調整後之數值調整因子。
表 2-11 適用於現存應用程式之已調整功能點數值之計算

2.3 XML

XML 為全球資訊協會於 1996 年底所制訂(World Wide Web

Consortium,W3C)，用來儲存結構性資料的標記語言。由於 XML 能夠自訂描述資料型態的標記以及跨平台的特性，與 HTML 相較，更能突顯出不同資料所代表的意義，因此被公認為新一代的網際網路語言。XML 技術的發展方興未艾，包括微軟(Microsoft)、奇異電器(GEIS)、波音公司(Boeing)、福特汽車(Ford)、第一商務(Commerce One)、宏道(BroadVision)、甲骨文(Oracle)、戴爾(Dell)、思愛普(SAP)等企業都致力於轉換系統符合 XML 標準的架構。由於 XML 卓越的跨平台資料處理能力，在資料交換的世界裡扮演舉足輕重的角色[15]。

本研究將 ER-DFD 轉化為結構化之 XML 檔案(如附錄 2)，而非以一般文字檔案之目的在於：

1.軟體系統之間的資料交換：

由於現今軟體規模及複雜度日益增加，對於軟體能否在所規劃的時程內完成及在開發時所需投入之人力，不免需要在軟體開發過程之中進行各方面之估算，例如軟體規模(size)、開發工作量(effort)，開發所需技術及資源等等[1]。因此，以估算軟體規模而言，若真實世界有不同團隊投入 ER-DFD 設計規格自動化估算功能點系統之開發，由於並無規範系統如何對 ER-DFD 進行讀取，因此可能出現不同可供系統讀取的格式。造成同一個 ER-DFD 設計規格在面對多個功能點估算系統時，就必須額外撰寫將 ER-DFD 轉成符合該系統讀取需求之自訂格式。因此本研究定義出表達 ER-DFD 之 XML 自訂標籤，並已表示成符合 XML 規範之 DTD 格式，期望此自訂標籤能夠成為表達 ER-DFD 之統一格式(如附錄 3)。

2.功能點資料庫建構：

XML 對於資料庫之處理具有完整之支援性，資料庫軟體可以直接解讀 XML 文件，並擷取出該文件之資料，並更新到相關的資料庫欄位中，因此對於建構專案歷史資料庫極為適合。

3.提供便利的搜尋處理能力：

物件導向語言 Borland Delphi 7.0，提供處理 XML 之元件，可以很容易地對 XML 檔案進行所需標籤之搜尋、讀取與建立。XML 是以樹狀架構方式呈現資訊，圖 2-5 為人力資源系統(HR system)之 XML 架構示意圖。利用 Borland Delphi 7 之 TXMLDocument 類別所宣告的物件”XMLDocument1”將 XML 檔案載入至記憶體等待後續處理，透過 XMLDocument1 物件所提供的屬性，可對 XML 檔案之節點名稱、屬性、屬性值等進行處理。如

HR 代表 XML 樹狀結構之根節點(root node)，根節點下層(指 L1 層)的所有節點均為其子節點。對於相關節點與屬性的存取方式，Delphi 7 以如下方式存取：

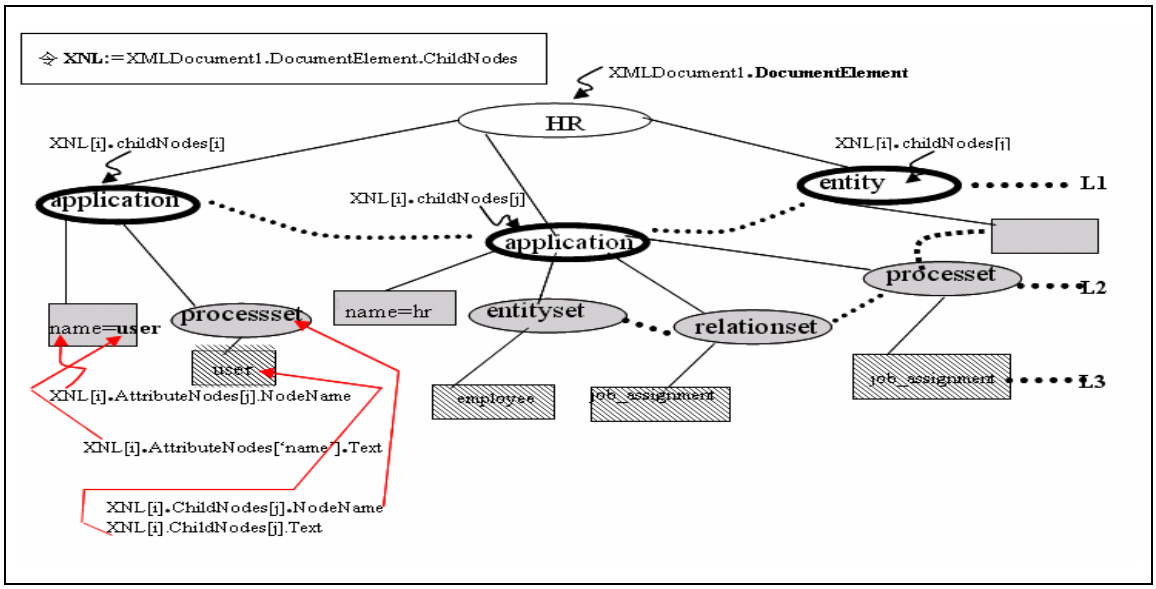


圖 2-5 人力資源系統(HR system)之 XML 架構示意圖

1.存取節點名稱：

XNL 代表 DocumentElement 之所有子節點之集合。
存取 L1 層之節點，則將 L1 視為 DocumentElement 之子節點,，以存取 application 節點為例：

(1) 令 XNL:=XMLDocument1.DocumentElement.ChildNodes;

NodeName 代表某一特定節點的節點名稱。

(2) if(XNL[i].NodeName=’application’) then .....

存取 L2 層之節點,則將 L2 視為 L1 之子節點,以存取 enityset 節點為例:

ChildNodes 為某特定節點之子節點集合。

if(XNL[i].**ChildNodes[j]**.NodeName=’entityset’) then.....

類推，要存取 L3 層之節點，即為

if(XNL[i].ChildNode[j].ChildNode[k].NodeName=’某節點名稱’)

then .....

2.存取屬性節點名稱：

XNL[i].**AttributeNodes[j]**.NodeName;

3.存取屬性節點之**值**：

name 為欲存取節點之屬性名稱。

XNL[i].AttributeNodes[‘name’].Text;

4.存取兄弟節點，如從 entityset 節點欲存取 relationset 節點：

XNL[i].NextSibling.NodeName