國立交通大學

土木工程研究所

碩士論文

運用物件導向技術於 IFC 建築資訊

Object Oriented Technology Applied to IFC Building Information

研究生：沈秉廷

指導教授：林昌佑 博士

中華民國 九十七 年 六 月

運用物件導向技術於 IFC 建築資訊

Object Oriented Technology Applied to IFC Building Information

研究生：沈秉廷
指導教授：林昌佑 博士

Student：Ping-Ting Shen
Advisor：Dr. Chang-Yu Lin

國立交通大學

土木工程學系

碩士論文

A Thesis
Submitted to Department of Civil Engineering
College of Engineering
National Chaio Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Civil Engineering
June 2008
HsinChu, Taiwan, Republic of China

中華民國九十七年六月

運用物件導向技術於 IFC 建築資訊

學生：沈秉廷　　指導教授：林昌佑　博士

國立交通大學土木工程學系(研究所)碩士班

摘要

在建築物生命週期，從規劃、建築設計、工程分析、估價發包、施工乃至營運維護，無時無刻需要進行資訊交換；然而資訊交換卻沒有一套共同標準，使得資訊在交換過程中時常發生錯漏，或是相同的資訊卻要重新建檔，再用性不佳，管理上也相當不易。由 IAI(International Alliance for Interoperability)國際組織提出的 IFC(Industry Foundation Classes)是一種公開的資訊交換標準，目的在使整個建築物生命週期的所有資訊能夠整合在一個建築資訊模型中(BIM, Building Information Model)，讓生命週期中所有軟體能夠共享、交換資訊。

本研究由物件導向的觀點探討 IFC 規格(Specification)設計與實作(Implementation)方式，介紹兩種免費的 IFC 實體檔案存取應用程式介面(API, Application Programming Interface)，並探討其 IFC 資訊存取方式。

本研究並運用物件導向技術，自行撰寫程式將 IFC Entity 規格轉換為.Net Framework 類別(Class)，再以類別庫(Class Library)封裝。此類別庫提供較直觀的 IFC 應用程式開發方式，並能夠運用物件導向於系統開發。本研究最後使用所建置之類別庫開發以 Plug-In 為架構的 IFC BIM 系統，並開發一些 Plug-In 元件進一步示範如何應用類別庫，包含: 3D 建築模型瀏覽、建築物框架建模、樑柱斷面資訊瀏覽 Plug-In 元件。

關鍵字: IFC、BIM、物件導向、Plug-In。

Object Oriented Technology Applied to IFC Building Information

Student：Ping-Ting Shen　　Advisor：Dr. Chang-Yu Lin

Institute of Civil Engineering

National Chiao Tung University

ABSTRACT

Information exchange is an important issue during the life circle of a building. However, the lack of exchange standard usually results information leakage and low efficiency. IFC (Industry Foundation Classes) which introduced by IAI (International Alliance for Interoperability) is an open standard for information exchange. IFC aims to describe all the information relating to a building's life circle and integrates them into a Building Information Model (BIM). Consequently, information can be exchanged and shared via BIM.

This research discusses the specification and implementation of IFC in an Object Oriented approach. Two charge-free APIs (Application Programming Interface) are discussed.

Additionally, this research introduces an IFC based .Net Framework class library which provides an easier and safer way to access IFC file. Also, this class library can be easily applied to Object Oriented programming. Finally, this research presents a Plug-In based IFC system, and demonstrates how to use this class library to develop Plug-In component, includes 3D building model viewer, building infrastructure builder, and beam/column profile viewer.

Keywords：IFC、BIM、Object Oriented、Plug-In

誌謝

研究所這兩年是我最快樂的求學生活，很高興能夠到交大唸書，這段期間在專業知識與待人處事上皆有所成長。首先要感謝我的指導教授林昌佑博士，對於學生在修課與論文撰寫，教授提供了許多協助與指導並給予學生相當大的研究自由，感激之情溢於言表。

在研究的過程中，感謝研究室成員：學長奕銘、益世、啟勇、弘毅、志偉及學姊雅晶還有學弟妹們，提供我許多建議與協助。另外感謝，同學承禹、昱德、昊志、怡廷，以及志銘、君暉等 IT 組的學弟妹，因為你們讓我的研究生活更充實。

在論文口試期間，要感謝交通大學洪士林教授、趙文成教授提供了很多寶貴的意見及建議，讓論文疏漏之處得以改進。在此深表感謝。

最後要感謝我的家人，尤其是我的父母，感謝你們的養育與栽培，在求學過程中給予我支持與鼓勵，讓我可以順利獲得碩士學位，希望你們身體健康，平安快樂。

目錄

表目錄

圖目錄

第一章 緒論

1.1 研究背景

AEC/FM(Architecture,Engineering,Construction and Facilities Management)領域中，一個建築物從規劃、建築設計、工程分析、估價發包、施工乃至營運維護，每一階段都牽涉到許多專業分工。各專業之間的互動相當的複雜，大量的資訊必須在不同專業領域間交換傳遞，故資訊如何完整且有效的在各階段傳遞、分享是相當重要的議題。

隨著電腦輔助設計(Computer-Aided Design, CAD)方法的發展及電腦輔助繪製設計系統(Computer-Aided Drafting and Design System, CADD)的普及，傳統的紙本資料(建築圖、工程圖、計算書等)逐漸被電子檔案取代。然而各家軟體公司採用的資訊描述格式不盡相同，導致在資訊交換傳遞過程中經常發生錯誤漏失，或是相同的資訊卻要重新建檔檢核，無法有效的分享資訊，除了浪費時間與人力，管理上也相當不易。

於是有後續研究提出了建築資訊模型(Building Information Model,BIM)的概念，希望將整個建築物生命週期所涵蓋的資訊保存在同一個資訊模型，讓建築物生命週期各階段的軟體能夠直接從這個資訊模型分享、交換資訊。為了能夠讓資訊完整有效的在各階段所使用的軟體間交換，軟體間需要有一個建築資訊交換標準，此標準必須能夠描述建築物生命週期各階段所涵蓋的資訊，並且受到各軟體的支援。

在資訊交換標準的發展方面，國際標準組織(International Organization for Standardization,ISO)提出了 STEP(Standard for the Exchange of Product model data)，工業產品資訊交換標準；NIST(National Institute of Standards and Technology)提出了 CIS/2(CIMsteel Integration Standards Release 2)，鋼結構資訊交換標準；IAI(International Alliance for Interoperability)則提出了 IFC(Industry Foundation Classes)，為 AEC/FM 領域公開資訊交換標準。

1.2 研究動機與目的

資訊的標準化是一股難以忽視的趨勢，IFC 在國外已經有相當多的研究，並有許多軟體公司已經開始遵循此標準。然而國內對於 IFC 的研究應用目前尚不多，為了促進國內對於建築資訊模型及資訊交換標準的研究發展與應用，本研究以 IFC 這個針對 AEC/FM 領域制定的資訊交換標準為對象，進行探討其規格及實作方法。

為了能夠快速的建立、擷取 IFC 資訊，本研究運用物件導向技術，自行撰寫程式將 IFC Entity 規格轉換為.Net Framework 類別(Class)，再以類別庫(Class Library)封裝，讓程式開發人員能夠更快速擷取、建立 IFC Entity 資訊，並運用物件導向於系統開發。

1.3 研究內容

本研究由物件導向的觀點探討 IFC 規格(Specification)設計與實作(Implementation)方式，介紹兩種免費的 IFC 實體檔案存取應用程式介面(Application Programming Interface, API)，並探討其 IFC 資訊存取方式。

本研究並運用物件導向技術，自行撰寫程式將 IFC Entity 規格轉換為.Net Framework 類別，再以類別庫封裝。此類別庫提供較直觀的 IFC 應用程式開發方式，並能夠運用物件導向於系統開發。本研究最後使用所建置之類別庫開發以 Plug-In 為架構的 IFC BIM 系統，並開發一些 Plug-In 元件進一步示範如何應用類別庫，包含: 3D 建築模型瀏覽、建築物框架建模、樑柱斷面資訊瀏覽 Plug-In 元件。

1.4 研究流程

1. 文獻回顧及研究相關知識加強:
   蒐集閱讀關於 IFC 的資訊及文獻，並加強物件導向的基本概念。熟悉.Net Framework 及相關工具。
2. 研究 IFC 規格:
   由 IAI International 官方網站下載最新版的 IFC 規格，了解其架構以及如何實作一個完整且有效的 IFC 檔案。使用幾種支援 IFC 輸入/出軟體，觀察其對 IFC 的支援情形。嘗試使用幾種 IFC 資訊存取應用程式介面(API, Application Programming Interface)，並了解其運作方式。
3. IFC 類別庫設計/實作:
   先根據 IFC 規格設計 IFC 類別庫。再撰寫類別轉換器將 IFC 規格中的 Entity 轉換為類別，最後以類別庫封裝並編譯成 DLL(Dynamic-Link Library)。
4. 類別庫測試/修正:
   使用類別庫實際進行資訊擷取與建立，檢討是否有需要改進及修正的地方。
5. 類別庫應用:
   實際應用類別庫進行 BIM 系統的開發，本研究先建置一個 Plug-In 系統，再利用類別庫開發 Plug-In。

1.5 論文架構

本論文總共分為六章。第一章為緒論，包含研究背景、研究動機與目的、研究內容、研究流程、論文架構，接著第二章為研究相關知識，包含建築資訊模型與建築資訊塑模、介紹相關建築資訊交換標準、物件導向基本概念、.Net Framework 介紹及相關應用研究探討。第三章為 IFC 規格與實作，包含 IFC 資訊描述方式、物件導向於 IFC 規格運用、IFC 規格基礎架構、IFC 建築資訊模型與模型視圖、IFC 實體檔案格式，並介紹兩種 IFC 資訊存取應用程式介面。

本論文第四章則在描述本研究提出之 IFC 類別庫的設計及建立細節，包含類別庫的架構及如何將 IFC 規格中的資料類型映對到程式語言以便進行資訊擷取建立，接著第五章為 IFC 類別庫的使用範例及應用，使用範例包含建立、擷取 IFC 檔案資訊及如何應用 IFC 類別庫於物件導向程式設計，應用部分為示範如何使用類別庫開發 BIM 系統，最後第六章為結論與建議。

第二章 研究相關知識

本章闡述關於本研究的相關知識，包括:建築資訊模型與建築資訊塑模、簡介 STEP、CIS/2、IFC 三種資訊交換標準、介紹物件導向的基本概念、.Net Framework 簡介、相關研究文獻回顧。

2.1 建築資訊模型/建築資訊塑模

建築資訊模型，是建築物在整個生命週期中所有產生與關聯的資訊集合體；建築資訊塑模(Building Information Modeling)則是指建立及管理建築資訊模型的過程。在塑模的過程中應謹慎考量模型資訊交換性(interoperable)及再利用性(reusable)。

一個 BIM 系統泛指能夠在建築物生命週期中建立、整合(integrate)及重複利用建築資訊與專業知識(domain knowledge)的系統[1]。BIM 系統結合 IT(Information Technology)技術、電腦輔助設計方法，以及 AEC/FM 領域的專業知識，以設計的角度建立、整合建築資訊模型。與以往認知的電腦輔助繪製設計系統最主要不同在於電腦輔助繪製設計系統主要以繪製幾何圖形資訊為主，而且從這些輸出的幾何圖形中無法直接取得有意義的資訊(例如門窗的型號、樑柱的長度)，必須經由人員判讀才能轉為有用的資訊，而 BIM 系統能夠由資訊模型中直接取得所需資訊。

一個 BIM 系統具有以下特性:

1. BIM 系統具有輸入/輸出建築資訊模型的能力。BIM 系統能夠輸入建築資訊模型進行資訊處理動作，也能夠將處理後的建築資訊模型輸出。
2. BIM 系統間能夠有效的分享資訊。一個建築物在不同階段的專業領域間必定有許多相關聯的資訊，這些資訊在 BIM 系統間必須能夠同步且有效的分享。例如當建築師使用建築 BIM 系統修改了某根樑的位置，結構技師的 BIM 系統必須能夠直接由建築 BIM 系統輸出的建築資訊模型顯示被改變的資訊。
3. BIM 系統將資訊的處理重心放在物件(Object)本身，每個物件都具有個別身份(Identity)與意義，並且含有關於此物件的相關資訊，這些資訊能夠以參數(Parametric)形式描述這個物件，例如一個樑的物件具有其位置、幾何、數量資訊。

BIM 系統涵蓋的範圍相當廣，包含建築規劃設計、結構分析模擬、建築機電、施工排程，營運管理...等。目前市面上已經有不少 BIM 系統，依照其所屬領域及開發公司整理如表 2-1。

2.2 建築資訊交換標準

為了讓不同 BIM 系統能夠完整有效的進行資訊交換，必須有一個描述建築資訊模型的資訊交換標準，若此系統為 BIM 的血肉，這個資訊交換標準即為 BIM 的靈魂，此標準必須能夠描述建築物生命週期各階段所涵蓋的資訊，並且受到各 BIM 系統的支援。目前較被 BIM 系統開發公司接受的標準有：STEP(Standard for the Exchange of Product model data**)** 、IFC(Industry Foundation Classes)、CIS/2(**C**IMsteel **I**ntegration **S**tandards Release 2)，分別介紹於以下章節，並將三種資訊標準的比較整理於表 2-2。

2.2.1 STEP

STEP(Standard for the Exchange of Product model data**)**為國際標準組織(ISO, International Organization for Standardization)提出的產品資訊交換標準，代號為 ISO10303，目的在於提供一個機制以描述整個產品生命週期的相關資訊，方便資訊交換，STEP 能夠描述的產品範圍很廣，幾乎涵蓋整個製造工業，包含:營建、機電、造船、航空...等。現由 ISO:TC184/SC4(ISO Technical Committee184-Sub-committee 4)[2]負責維護，並定期召開會議修訂標準。STEP 由許多部分標準(part standard)組成，並可分為 9 個區塊(圖 2-1)。敘述如下:

1. 描述方法(Description methods Parts 1-19)
   定義 STEP 的規格，其中 part11 為 EXPRESS 資訊塑模語言(data modeling language)。EXPRESS 能夠運用物件導向的概念於資訊描述，其資訊描述語法很像 PASCAL 程式語言，但基本上 EXPRESS 所描述的資訊是獨立於程式語言的，如同描述程式流程的虛擬碼。此描述語言也被運用於描述 IFC 及 CIS/2 的規格。

2. 實作方法(Implementation methods Parts 20-29)
   說明如何實現 STEP 標準，其中 part21 定義 STEP 實體檔案格式(STEP Physical File)，將資訊模型以實體檔案表示，IFC 及 CIS/2 亦使用此方法表示資訊模型。

3. 一致性測試方法及架構(Conformance testing methodology and framework Parts 30 -39)
   資料在設計完成後必須通過驗證才能符合 STEP 標準，這部份定義了驗證資料的方法及相關測試。

4. 整合性一般資源(Integrated generic resources Parts 40-49)
   定義了各領域的一般性資源，如形狀、尺寸、單位的描述方式。

5. 整合性應用資源(Integrated application resources Parts 100 - 199)
   根據一般性資源定義特定領域的資源，如營建領域定義了樑、柱、牆等元件。

6. 應用協定(Application protocols Parts 200 - 299)
   規定特定領域實現標準的需求資料。

7. 抽象測試套件(Abstract test suites Parts 300 - 399)
   用來測試是否符合應用協定。

8. 應用解釋結構(Application interpreted constructs Parts 500 - 599)
   用於開發新的資料模型。

9. 應用模組(Application Modules Parts 1000 -)
   為小型的資訊模型，用於開發新的應用協定。

2.2.2 IFC

IFC(Industry Foundation Classes) 由 IAI(International Alliance for Interoperability)國際組織[3]提出且維護，針對 AEC/FM 領域設計的公開資訊交換標準。目的在於讓建築物生命週期所有軟體能夠藉由 IFC 描述建築資訊，進而提高資訊的交換性與再用性。IAI 推動的 BuildingSmart[4]計畫即是以 IFC 為基礎實現 BIM 的理念。IFC 規格的發展進度見圖 2-2，IFC 從 1.5 版開始釋出，目前最新版為 2x3 Final[5]，亦為本研究所使用的 IFC 版本。

IFC 以物件導向(Object Oriented)為基本概念，參考引用了 STEP 部分標準，為了實作考量也參考了 C/C++程式語言的一些特性。以架構而言可以分為四個層次[6](圖 2-3)，由下而上分別為:資源層(Resource Layer)、核心層(Core Layer)、資訊交換層(Interoperability Layer)、專業領域層(Domain Layer)，每層分別定義了不同的種類的資料類型(Data Type)與實體(Entity)，分別敘述如下:

資源層:

位於 IFC 架構的最底層，此層定義較一般性的實體，有點像是 STEP 的整合性一般資源，通常它們都是被較高層次的實體參考，例如:幾何形狀

被一個產品參考，也就是一個產品具有幾何形狀這個資訊，當產品存在時它的幾何形狀才能被定義，但也有例外：工具資源(Utility)、量測資源(Measure)部份可以單獨存在。

核心層：

此層定義 IFC 基礎的實體，此層的實體定義了許多共同的介面，它們可被資訊交換層或專業領域層的實體參考或繼承。此層又可細分為兩層:內核(Kernel)層與產品延伸(Product Extension)層。內核層定義最基礎的實體，這些實體不被限定在 AEC/FM 領域，並只能參考資源層的實體，例如定義“產品”具有“位置”與“形狀描述”這兩個屬性。產品延伸層定義較高階的實體，他們皆繼承自內核層的實體，而且都是屬於 AEC/FM 領域，例如:建築物實體。

資訊交換層：

此層定義能夠在 AEC/FM 領域內做資訊交換的共同實體，例如:樑、柱、門、窗、空間…等資訊；並且，各個專業領域可將其資訊附加於此層的實體上，例如:樑柱上可能有材料資訊、施工/完工日期、結構分析結果…等資料。

專業領域層：

此層為定義 AEC/FM 內各專業領域的實體，包含:建築、結構分析、營建管理、設施管理、機電設備、水電空調、配管工程…等專業領域的實體，例如：結構分析的分析模型、結構分析的束制條件、營建管理的人力資源。目前此層定義尚未完備，由於牽涉到許多專業領域，故標準訂定有一定的困難。

IFC 2x 有一個基本的層與層間相互參考(reference)的原則，“重力原則”(Gravity Principle)[7](見圖 2-4)，規定各層級的實體只能與相同層級的實體或低於此層級的實體相互參考，例如資源層的實體無法參考核心層的實體，所有 IFC 的規格設計都緊緊遵循著此原則。目前已有多家軟體廠商認同 IFC 建築資訊交換標準，並努力讓自家軟體支援 IFC 建築資訊模型的輸入/輸出，也有不少軟體已通過 IAI 的 IFC 相容性驗證，在 IAI ISG(Implementer Support Group)網站[8]上有建立相容軟體資料庫，雖然各家軟體廠商支援程度不一，但隨著 IFC 規格的更趨成熟，相信支援度會愈來愈高。本研究在第三章會就 IFC 的規格定義及實作方式做進一步的探討。

2.2.3 CIS/2

CIS/2(CIMsteel Integration Standards Release 2)，為 NIST(National Institute of Standards and Technology)[9]提出，目的在提供鋼結構建築物一套資訊交換的標準。CIS/2 包含了整個鋼結構生命週期的資訊，包括分析、設計、細部設計、製造。CIS/2 與 IFC 同樣使用 EXPRESS 語言描述規格。CIS/2 有三個主要的模型[10]，分別敘述如下:

分析模型(analysis model)

一個分析模型包含許多節點(node)與元素(element)支援幾種靜態(static)與動態(dynamic)分析方法。

設計模型(design model)

一個設計模型包含許多設計組合(design assembly)，每個設計組合又可分解為更多的部分設計(design part)及連結系統設計(design joint system)。

製造模型(manufacturing model)

一個製造模型包含許多細部製造組合(manufacturing assembly)，用於細部設計、規劃、製造。

CIS/2 已被許多軟體支援，NIST 也致力於 CIS/2 與其他資料格式的交換，並開發了一個“CIS/2 to VRML and IFC Translator”軟體，允許將 CIS/2 的檔案轉為 IFC 或是 VRML (Virtual Reality Modeling Language)。

2.3 物件導向

物件導向為一種程式設計的方法論，與結構化程式設計(structured programming)最大的不同在於將程式分解的方法，但其核心想法都為“各個擊破”(divide-and-conquer)[11]。結構化程式設計將程式視為一個程序(process)，再依演算法或功能性分解為許多模組(module)，每個模組都可視為程序的步驟(step)之一。物件導向程式設計將程式視為一個物件的集合體，藉由物件之間的互動建構程式，將程式分解為物件後，再對物件做功能上的分解。使用物件導向分解程式，最主要的優點讓整個程式更有彈性，並讓程式更容易再利用(reuse)。

2.3.1 物件導向基本概念

物件導向具有幾個基本概念，分別敘述如下：

類別(Class)

類別為物件的規格，定義物件的屬性(attribute)、方法(method)。類別可視為物件的模子，例如:“生物”類別，定義了名稱、種類…等屬性及生長、繁殖、死亡…等方法，所有屬於生物的物件都具有這些成員(member)，這些成員也可視為這些物件的介面(interface)。

物件(Object)

物件為類別程式執行時期(runtime)的實例(instance)，能夠藉由呼叫物件的方法進行運算及擁有自己的狀態(state)，例如: 你跟我都是屬於“人”這個類別，你我都具有“姓名”這個屬性，但很有可能我們的姓名是不一樣的，也就是我們各自擁有不同的狀態。

繼承(Inheritance)

子類別(subclass)可藉由繼承父類別(super class)擁有父類別的所有成員，並可額外擁有自己的成員或是覆寫(override)父類別的方法，子類別是父類別的一種，所以父類別也能夠視為所有子類別的介面，例如:“貓”類別與“狗”類別都是繼承自“生物”類別，他們都具有生物的成員，也各自擁有自己其他的成員，貓可能有“抓老鼠”這個方法，狗有“看門”這個方法。

封裝(Encapsulation)

封裝或是資訊隱藏(Information Hiding)目的在將實作的細節與私有成員隱藏起來，對於該物件的操作只能根據其公開成員，就像我們沒辦法由晶片的成品直接了解其電路設計一樣。

資料抽象化(Data Abstraction)

資料抽象化目的在將物件的介面與實作分開來，例如:對於“汽車”物件，我們只要知道它能駕駛就夠了，至於怎麼製造生產對我們不是太重要。抽象資料類型(ADT, Abstract data type)為一種特別的資料型態，它用以表示類別的介面，一個類別能夠實現許多不同介面，例如:“狗”類別實現了“寵物”跟“朋友”的介面，你可以把牠當作寵物也可以當做朋友。

多型(Polymorphism)

多型是指不同的物件能夠藉由相同的介面加以操作，結果會因為實作的不同而有異。例如:“狗”類別與“貓”類別都有實作“寵物”這個介面，你可以將狗跟貓都當作寵物看待，但身為寵物，狗跟貓表現出的行為是很不一樣的。

當我們說一個程式語言支援物件導向，那它必須具有下列特性:

1. 它必需支援物件。
2. 物件必須屬於某一類別。
3. 必需支援繼承。

一個程式語言支援前兩項特性，但不支援繼承時只能稱為物件基礎語言(object-based language)。目前常見支援物件導向程式語言有:C++、Java、C#、VB.net。雖然 C 語言不支援物件導向，但是由於語言本身彈性很大，能夠直接操控記憶體，要實現物件導向也是可能的，如 Linux 核心主要以 C 語言撰寫卻能活用許多物件導向的概念。

2.3.2 物件導向於建築資訊交換標準的應用

物件導向以物件為基礎的思考方式，非常適合用於表達真實世界的事物，STEP、IFC、CIS/2 三種資訊交換標準皆參考了物件導向的概念，用以描述資訊。AEC/FM 的產品是由建築元件組成，例如一棟建築物含有樑、柱、牆、版、門窗、樓梯...等建築元件，每一個元件皆可視為物件，其各自包含自己的屬性，例如一根樑有斷面、位置、長度、材料...等屬性。利用物件導向繼承的概念能夠很容易的擴展物件的資訊，例如制定“建築元件”這個物件可能定義了較一般性的資料像是使用材料、是否為受力元件，“樑”物件跟 “樓梯”物件，繼承自“建築元件”他們除了擁有自己的屬性外同時也擁有“建築元件”的屬性。

許多 BIM 系統在處理建築資訊上也運用了物件導向的概念，以建築設計為例，傳統的 CADD 系統(如 AutoCAD, MicroStation, QuickCad)，對於真實世界的建築元件，只是由點、線、弧形等幾何訊組成；而建築設計 BIM 系統(如 Revit、ArchiCad、Bentley Architecture)對於建築元件則是對應到一個物件，並藉由參數化的方式進行設計。

2.4 .Net Framework

.Net Framework 是微軟(Microsoft)提出於 Windows 平台的軟體開發、建置、執行方案。它包含了兩個主要的部份[12]：

1. 基礎類別庫(Base class library)，包含使用者介面、資料存取、資料庫、演算法、網路通訊...等許多作業系統的功能，此類別庫運用許多物件導向概念建置，讓程式開發者快速的使用這些類別，降低程式開發者的負擔。

2. CLI(Common Language Infrastructure)架構(圖 2-5)，CLI 又分為兩個部份 CIL(Common Intermediate Language)以及 CLR(Common Language Runtime)。

CIL 為一種中繼語言，如同 Java 的位元組碼(Byte Code)，使用微軟提供的不同程式語言的編譯器能夠將不同程式語言的程式碼編譯成 CIL，並且這些程式語言都能夠使用基礎類別庫進行開發程式，目前支援的程式語言有:Visual C++、Visual Basic.Net、C#、J#，雖然使用的語言不同但最終都會編譯成 CIL。CLR 扮演一個虛擬機器的角色，如同 Java 的 JVM(Java Virtual Machine)，它負責在執行時將 CIL 轉換為該平台的原生碼(Native Code)，屬於 JIT 編譯器(Just In Time Compiler)。

.Net Framework 具有幾個特點[12]:

1. 安全：由於 CIL 含有整個程式的所有資訊，故能夠在程式載入時做型別和程式碼驗證。
2. 跨平台：由於所有的程式碼最終都編譯成 CIL，故只要在不同平台實現 CLR 即可達到跨平台的效果。
3. 互動性高：能夠與微軟的過去的一些技術結合，像是 COM(Component Object Model)技術、OLE(Object Linking and Embedding)技術。

本研究主要使用.Net Framework 1.5 版及 Visual Basic .Net 程式語言，雖然目前最新版為 3.5，但 1.5 的功能對於本研究已經足夠。

2.5 IFC 相關應用研究

對於 IFC 建築資訊交換標準已經有不少研究與應用，在 IFC 應用與研究的過程中為了從 IFC 實體檔案建立或擷取建築資訊模型的資訊，必須有一個存取實體檔案的方法，當然我們也可以開啟 IFC 檔案直接從文字檔案抓取資訊，但若是能夠藉由一個應用程式介面存取資訊，我們可以在不同的程式中使用相同的方法存取資訊，讓程式更容易重複使用。

近年來國外對於 IFC 應用之研究有：

Po-Han Chen 等人(2004)以 IFC 為基礎並使用 JavaEE 技術及 XML 技術建立一個資訊交換伺服器，讓建築師與結構技師能夠藉由 IFC 建築資訊模型與網路技術進行協同設計。M.Hassanien Serror 等人(2007) 從結構分析的角度探討如何運用 IFC 建築資訊模型實現建築物結構分析模型，並實際運用於結構分析，此研究部分被 IAI 採用以進行 IFC 的設計。

Ali Murat Tanyer 等人(2004) 建立一個以 IFC 為基礎的資料庫，並開發結合 4D 建築工程規劃及估價的系統，能夠從這個資料庫取得建築資訊模型。Changfeng Fu 等人(2005) 使用 IFCsvr API 開發一個能夠瀏覽 IFC 建築資訊模型內容的系統，由三維圖形出發到多維的資訊，如時間、成本等專案相關資訊。

國內對於 IFC 近年來有愈來愈多的相關研究，最近幾年對於 IFC 的相關研究有：

蘇建豪(2000)嘗試將 STEP、IFC 等資訊標準運用於工程資訊管理，藉由建立標準資訊模型進行資料交換並將所有電子資訊存入物件導向資料庫中，形成一虛擬中心資料庫，達成資訊共享之目的。潘稟嘉(2000)將 IFC 建立的營建圖檔資訊以網路方式共享，讓 IFC 圖形資訊能夠透過網路進行解讀、存取及展示，並應用互動式網頁技術，將 IFC 檔案動態轉換為 VRML，並建立 IFC 共享資料庫，讓 IFC 圖檔資訊能夠過資料庫進行存取，在 IFC 資訊的存取上為使用自行撰寫的 VB 程式處理。

蔡志偉(2007)針對結構分析所需資料進行擷取 IFC 建築資訊，並建立結構分析部分資訊，以節省在結構分析時重複建置模型資訊的時間，在 IFC 資訊的存取上為使用自行撰寫的 JAVA 程式進行處理。樊啟勇(2007) 對幾種支援 IFC 的 BIM 系統進行探討與比較並針對結構設計所需結構元件如樑柱等之組成資料進行擷取，並建立結構分析相關部分資訊，在 IFC 資訊存取上為使用自行撰寫的 FORTRAN 程式處理。

曾國峰(2007)以 IFC 為基礎建置一個視覺化 3D 工序模擬系統 CS3D(3D Construction Sequence Simulation System)。CS3D 使用 IFC Engine OCX 呈現 3D 模型及運用 IFCsvr API 擷取幾何物件屬性資料。能夠將建築 BIM 系統所匯出之 IFC 檔案載入系統，讓使用者利用便捷的編製、儲存與施工順序模擬功能。吳翌禎(2007) 提出了一套工程專案資訊整合管理架構，以解決工程監管單位在工程專案中所遭遇之資訊介面及資料整合管理上的問題。此研究根據所提出架構建立一視覺化專案管理系統，此系統在資訊交換上以 IFC 為基礎，並且使用 IFCsvr API 進行資料存取。

第三章 IFC 規格與實作

本章就 IFC 的規格(Specification)及實作(Implementaion)進行探討，規格部份包含 IFC 資訊的描述方式、物件導向於 IFC 規格運用及 IFC 規格的基礎架構；實作部份包含 IFC 建築資訊模型與模型視圖、IFC 實體檔案格式及 IFC 資訊存取應用程式介面。

3.1 IFC 資訊描述方式

IFC 的規格以物件導向為基礎進行設計，運用了類別、繼承、抽象資料類型與多型等概念定義資訊，故必須有一個適合的資訊描述方式才能表達這些資訊。IAI 使用 EXPRESS 資訊塑模語言(data modeling language)作為 IFC 資訊的描述方式並定義了四種資料型態[23]，分別敘述如下:

Defined Type

用於表示基礎資料型態，包含：REAL(實數)、INTEGER(整數)、NUMBER(數字)、LOGICAL(邏輯符號)、BOOLEAN(布林代數)、BINARY(位元)、STRING(字串)。Defined Type 很像使用 C/C++的 typedef，目的都在給予基礎資料型態定義一個更語意的別名(alias)。以“IfcMassMeasure”及“IfcLabel”為例，其 EXPRESS 描述如下:

```
TYPE IfcMassMeasure = REAL;
END_TYPE;
```

IfcMassMeasure 為表示物品重量的 Defined Type，其基礎型態為實數(REAL)。

```
TYPE IfcLabel = STRING;
END_TYPE;
```

IfcLabel 為描述文字標籤的 Defined Type，其基礎型態為字串(STRING)。

Enumeration

用於表示狀態或類型，它的值只能屬於定義的內容之一，所定義的內容將以文字型式存在。Enumeration 很像 C/C++的列舉(enum)，目的都在表示一個狀態，而且這個狀態必須為列出來的狀態之一。以“IfcBeamTypeEnum”為例，其 EXPRESS 描述如下:

TYPE IfcBeamTypeEnum = **ENUMERATION OF**
(BEAM,
JOIST,
LINTEL,
T_BEAM,
USERDEFINED,
NOTDEFINED);
END_TYPE;

IfcBeamTypeEnum 定義樑的類型，能夠為 BEAM(一般水平結構樑)、JOIST(用以支撐樓地板的小樑)、LINTEL(水平橫楣樑)、T_BEAM(包含樓地板的 T 型樑)、USERDEFINED(自行定義)、NOTDEFINED(未定義)。

Select Type

這是一種比較特殊的資料型態，它必須屬於內容定義的資料型態之一，與 Enumeration 不同的是，他的內容就是一種資料型態，此資料型態可以為 Defined Type 或是 Entity，而不是單純表示狀態的文字。Select Type 很像 C/C++ 的 union，表示一個存放不同的資料型態的記憶體位置。以 “IfcSimpleValue”為例，其 EXPRESS 描述如下:

TYPE IfcSimpleValue = **SELECT**
(IfcInteger,
IfcReal,
IfcBoolean,
IfcIdentifier,
IfcText,
IfcLabel,
IfcLogical);
END_TYPE;

IfcSimpleValue 表示一個簡單的數值，可以是 IfcInteger(整數)、IfcReal(實數)、IfcBoolean(布林值)、IfcIdentifier(字串)、IfcText(字串)、IfcLabel(字串)、IfcLogical(邏輯符號)這幾種 Defined Type 其中之一。

Entity

它是 IFC 規格中最複雜也最重要的資料型態，其可以視為物件導向的類別(Class)，它能包含各種屬性、或是繼承其他 Entity、以及定義成抽象資料類型，並可以描述一些法則(rule)。它是 IFC 表達資訊最主要的方式，以

“IfcOrganizationRelationship”為例，其 EXPRESS 描述如下:

```
ENTITY IfcOrganizationRelationship;
    Name                 : IfcLabel;
    Description          : OPTIONAL IfcText;
    RelatingOrganization : IfcOrganization;
    RelatedOrganizations : SET [1:?] OF IfcOrganization;
END_ENTITY;
```

上例中 IfcOrganizationRelationship 用於描述組織與組織間的關係，其定義了四個屬性:Name 表示此關係的名稱，資料型態為 IfcLabel；Description 表示此關係的描述，資料型態為 IfcText；RelatingOrganization 表示關聯組織，資料型態為 IfcOrganization；RelatedOrganizations 表示與關聯組織具有此關係的其他組織，資料型態為 IfcOrganization。

Entity 屬性的資料型態必須為 IFC 定義的四種資料型態之一，若該屬性為選擇性存在則必須加上“OPTIONAL”關鍵字，如前述 Description 屬性。並且屬性也能夠以集合體存在，IFC 定義兩種集合體 SET 與 LIST 分別表示無序及有序集合體，如前述 RelatedOrganizations 為無序的 IfcOrganization 集合體。

IFC Entity 常常利用關聯屬性(Relating)與被關聯屬性(Related)來表示一對多的關係，如前述的 RelatingOrganization 與 RelatedOrganizations。除了屬性之外，Entity 能夠藉由法則進一步定義，這些法則包含:

1. 逆向法則(Inverse Rule)：用來表示逆向參考的關係，例如 IfcOrganization 定義兩個逆向關係: IsRelatedBy 與 Relates 表示其分別在 IfcOrganizationRelationship 的 RelatedOrganizations 與 RelatingOrganization 屬性被參考。
2. 獨特法則(Unique Rule)：用來定義屬性的唯一性，例如定義 ID 這個屬性的內容在同個資訊模型中不能重複出現。
3. 衍生法則(Derive Rule)：用來表示一個虛擬屬性，此屬性是由同個 Entity 的其他屬性衍生而來，例如一個矩形斷面的面積由其長、寬計算而來。
4. 領域法則(Domain Rule)：定義一個屬性的範圍，例如定義 T 型斷面的翼板寬度必須大於腹板寬度。

3.2 物件導向於 IFC 規格運用

本節介紹 IFC 規格在設計上如何運用物件導向的概念包括繼承、抽象資料類型與多型。

3.2.1 繼承

IFC 定義 Entity 能夠繼承自其他 Entity，但只限於單一繼承(Single Inheritance)，也就是無法同時繼承自多個 Entity，但可同時被其他多個 Entity 繼承，此外 Entity 繼承關係同樣遵循了“重力原則”，IFC 規定各層級的 Entity 只能繼承(被繼承)相同層級的 Entity 或低於此層級的 Entity，例如資源層的 Entity 無法繼承核心層的 Entity。IFC 規格設計上運用了許多繼承，藉由繼承使得規格更容易擴充與維護，以“IfcAddress”、“IfcPostalAddress”、“IfcTelecomAddress”三個 Entity 的繼承關係為例進行說明如下。

IfcAddress 的 EXPRESS 描述如下:

```
ENTITY IfcAddress
ABSTRACT SUPERTYPE OF(ONEOF(IfcPostalAddress,IfcTelecomAddress))
    Purpose      : OPTIONAL IfcAddressTypeEnum;
    Description  : OPTIONAL IfcText;
    UserDefinedPurpose  : OPTIONAL IfcLabel;
END_ENTITY;
```

IfcAddress 定義“位址”的基本資料，它有三個屬性:Purpose 表示位址的意義，可能是住家(HOME)或是工址(SITE)等；Description 描述這個位址的資訊；UserDefinedPurpose 自行定義位置的意義。此 Entity 藉由 ABSTRACT 關鍵字定義為抽象資料類型，並由 SUPERTYPE OF 關鍵字得知為 IfcPostalAddress、IfcTelecomAddress 的父類別(Super Class)，它同時被兩個 Entity 所繼承。

IfcPostalAddress 的 EXPRESS 描述如下：

```
ENTITY IfcPostalAddress
SUBTYPE OF (IfcAddress );
    InternalLocation : OPTIONAL IfcLabel;
    AddressLines : OPTIONAL LIST [1:?] OF IfcLabel;
    PostalBox : OPTIONAL IfcLabel;
    Town : OPTIONAL IfcLabel;
    Region : OPTIONAL IfcLabel;
    PostalCode : OPTIONAL IfcLabel;
    Country  : OPTIONAL IfcLabel;
END_ENTITY;
```

IfcPostalAddress 用 SUBTYPE OF 關鍵字表示其繼承自 IfcAddress，除了定義了自己的屬性外，其額外擁有 IfcAddress 的所有屬性，它進一步擴展“位址”的內容為“郵件位址”。它定義了幾個描述這個 Entity 的屬性，如：InternalLocation 表示其內部位置，可能是某個樓層或是房間編號；AddressLines 表示郵件地址；Town 表示位址所在的城市，可能是某個鄉鎮；Region 表示位址所在的區域，可能是某個省份或是縣市；PostalCode 為該位址的郵遞區號；Country 為該位址所在的國家。

IfcTelecomAddress 的 EXPRESS 描述如下：

ENTITY IfcTelecomAddress
SUBTYPE OF (IfcAddress **);**
TelephoneNumbers **: OPTIONAL LIST [1:?] OF** IfcLabel;
FacsimileNumbers **: OPTIONAL LIST [1:?] OF** IfcLabel;
PagerNumber **: OPTIONAL** IfcLabel;
ElectronicMailAddresses : **OPTIONAL LIST [1:?] OF** IfcLabel;
WWWHomePageURL **: OPTIONAL** IfcLabel;
END_ENTITY;

由上述 SUBTYPE OF 關鍵字得知 IfcTelecomAddress 同樣繼承自 IfcAddress，除了自己定義的屬性外，其也擁有 IfcAddress 的所有屬性，它進一步將“位址”的內容擴展為“通信位址”。它定義幾個描述這個 Entity 的屬性，如：TelephoneNumbers 為該位址的電話號碼；FacsimileNumbers 為該位址的傳真號碼；ElectronicMailAddresses 為電子郵件位址；WWWHomePageURL 為網頁位址。

IFC 規格在設計上運用了許多如上述範例的繼承關係進行擴充 Entity 的內容，當規格需要修正時也能藉由修改部份的 Entity 進而影響其子類別，在規格的維護上較為方便。

3.2.2 抽象資料類型與多型

IFC Entity 可被定義為抽象資料類型，這種類型的 Entity 只能被繼承而無法成為物件，它通常用於描述一個共同的概念(Common Concept)，抽象資料類型可以視為所有子類別(Sub Class)的共通介面，對於參考它的 Entity 而言只要了解它具有這些介面，因此藉由抽象資料類型能夠達到多型的效果，被參考時實際上必須以非抽象資料類型的子類別存在。前面提到的“IfcAddress”即是一種抽象資料類型，它被“IfcOrganization”參考。

IfcOrganization 的 EXPRESS 描述如下：

```
ENTITY IfcOrganization;
   Id    : OPTIONAL IfcIdentifier;
   Name  : IfcLabel;
   Description : OPTIONAL IfcText;
   Roles : OPTIONAL LIST [1:?] OF IfcActorRole;
   Addresses : OPTIONAL LIST [1:?] OF IfcAddress;
END_ENTITY;
```

IfcOrganization 的 Addresses 屬性資料型態為 IfcAddress，但由於其為抽象資料型態，實際上被參考可能是“IfcPostalAddress”或“IfcTelecomAddress”這兩個非抽象子類別。

再以“IfcExtrudedAreaSolid”這個常被用來描述建築元件形狀的 Entity 為例，其 EXPRESS 描述如下:

```
ENTITY IfcExtrudedAreaSolid
SUBTYPE OF ( IfcSweptAreaSolid);
   ExtrudedDirection : IfcDirection;
   Depth : IfcPositiveLengthMeasure;
END_ENTITY;
```

它所代表的形狀為一個二維的斷面往指定的方向伸展固定的長度，如圖 3-1，ExtrudedDirection 表示伸展的方向；Depth 為伸展的長度。由 SUBTYPE OF 關鍵字得知其繼承自 IfcSweptAreaSolid 這個 Entity，其 EXPRESS 描述如下:

```
ENTITY IfcSweptAreaSolid
ABSTRACT SUPERTYPE OF(ONEOF(IfcExtrudedAreaSolid,
                            IfcRevolvedAreaSolid,
                            IfcSurfaceCurveSweptAreaSolid))
SUBTYPE OF ( IfcSolidModel);
   SweptArea : IfcProfileDef;
   Position  : IfcAxis2Placement3D;
END_ENTITY;
```

IfcSweptAreaSolid 的 SweptArea 屬性用來表示這個形狀的斷面，資料型態為 IfcProfileDef(Entity)，其 EXPRESS 描述如下:

```
ENTITY IfcProfileDef
ABSTRACT SUPERTYPE OF  (ONEOF(IfcParameterizedProfileDef,
                        IfcArbitraryOpenProfileDef,
                        IfcArbitraryClosedProfileDef,
                        IfcCompositeProfileDef,
                        IfcDerivedProfileDef));
  ProfileType : IfcProfileTypeEnum;
  ProfileName : OPTIONAL IfcLabel;
END_ENTITY;
```

由 ABSTRACT 關鍵字得知它是一個抽象資料類型，ProfileType 屬性定義斷面類型；ProfileName 屬性表示斷面名稱。由於它是抽象資料類型，當 IfcExtrudedAreaSolid 藉由 SweptArea 屬性參考這個 Entity 時實際上是參考繼承 IfcProfileDef 的非抽象子類別。IfcProfileDef 繼承關係如圖 3-2，實際上被參考的 Entity 可能是繼承它的 IfcIShapeProfileDef(描述 I 型斷面)或是 IfcCircleProfileDef(描述圓形斷面)等非抽象子類別。

3.3 IFC 規格基礎架構

在 IFC 架構的內核層(Kernel Layer)與產品延伸層(Product Extension)定義了一些基礎 Entity，它們對於 IFC 規格有相當重要的意義，所有此層以上的 Entity(除了資源層外)都是繼承自這些關鍵的 Entity。整理這些基礎 Entity 繼承關係及所屬層級如圖 3-3，以 IfcRoot 為頂點，藉由繼承擴展 Entity 的內容，以下由繼承關係介紹這些基礎 Entity 制定的意義以及其重要的子類別。

IfcRoot

它是一個抽象資料類型，也是繼承的頂點，除了資源層以外的 Entity 都具有 IfcRoot 的身分。就如同 JAVA 或是.Net Framework 類別庫中的 Object 類別，它主要的功能在於提供物件辨識、物件擁有權、變更歷史，以及名稱與敘述。GlobalId 這個屬性在 IFC 資訊模型範圍內必需是唯一，其可當作 Entity 物件的索引。

IfcObjectDefinition

它繼承 IfcRoot，同樣是抽象資料類型，所有物體及事件都具有這個身份。它定義了四個逆向關係:

1. HasAssignments

   表示此 Entity 能夠藉由 IfcRelAssigns 指定給其他 Entity，例如把一些工作指定給某家包商。

2. IsDecomposedBy

表示此 Entity 能夠藉由 IfcRelDecomposes 被分解，例如一個樑柱接頭可以分解成許多結構元件。

3. Decomposes

表示此 Entity 能夠藉由 IfcRelDecomposes 組成其他 Entity，其為組成其他 Entity 的部件。

4. HasAssociations

表示此 Entity 能夠藉由 IfcRelAssociates 與其它 Entity 產生關聯，例如一種材料可能被用在許多構件上。

繼承它的類別有:

IfcResource:用以描述建築生命週期中牽涉到的資源，如機具、人力。

IfcProject:用以描述建築專案相關資訊，如單位、座標系統。

IfcProduct:用以描述一個具有形狀與位置屬性的物件，如樑、柱。

IfcProcess:用以描述一個行程，如規劃行程、設計行程。

IfcGroup:用以描述群組集合，如將產品分為“已完成”與“未完成”。

IfcControl:用以描述管理，如描述一件工作的時間表。

IfcActor:用以描述組織人員的角色，如將一個組織設定為業主。

IfcRelationship

它繼承自 IfcRoot ，用來描述 Entity 與 Entity 之間的關係，只要具有 IfcRelationship 身分的 Entity 命名都是以“IfcRel”做開頭。IfcRelationship 是一個抽象資料類型，本身沒有任何屬性，它分別被 IfcRelAssigns、IfcRelDecomposes、IfcRelAssociates、IfcRelDefines、IfcRelConnects 繼承，這五個 Entity 定義了 IFC Entity 之間五個主要的關係:

1. IfcRelAssigns

用來定義指定的關係，將一組 Entity 指定給一個特定的 Entity。繼承他的 Entity 有: IfcRelAssignsToProcess，指定行程；IfcRelAssignsToProduct，指定產品；IfcRelAssignsToControl，指定管理；IfcRelAssignsToResource，指定資源；IfcRelAssignsToActor，指定人員；IfcRelAssignsToGroup，指定群組。

2. IfcRelDecomposes

用來定義分解、組合的關係，定義一組 Entity 組成一個 Entity。IfcRelAggregates 子類別用來描述物體的分解組合，例如一個工程專案由可以由好幾個工址組成。

3. IfcRelAssociates

用來建立 Entity 間關聯的關係，定義一組 Entity 關聯到一個特定的 Entity，重要的子類別有: IfcRelAssociatesMaterial，材料關聯；

IfcRelAssociatesDocument，文件關聯。

4. IfcRelDefines

用來將一組 Entity 給予一個定義屬性或是類型。重要的子類別有: IfcRelDefinesByProperties，定義屬性；IfcRelDefinesByType，定義類型。

5. IfcRelConnects

用來定義 Entity 間連結的關係，這些連結通常都是用在特定的地方。IfcRelContainedInSpatialStructure 子類別定義一組佔有空間的物體(樑、柱、牆等)連接到一個空間容器 Entity(樓層 、建築物等)。

IfcPropertyDefinition

具 IfcPropertyDefinition 身分的 Entity 用所有的屬性來描述一個特性。而這些特性大多屬於特定領域，例如: IfcDoorPanelPropertiesd 子類別用來描述門開口的方式。其中 IfcPropertySet 是一個能夠藉由一組 IfcProperty 自己定義一個屬性，例如能夠自定義一個描述"舒適度"的 IfcPropertySet，那可能就必須擁有溫度、溼度、風速等 IfcProperty。

3.4 IFC 建築資訊模型與模型視圖

制定 IFC 規格的目的在於實現 BIM 資訊交換的精神。將建築物整個生命週期的資訊利用 IFC 描述成建築資訊模型，並運用於資訊交換稱為 IFC 的實作。本節內容包含基礎 IFC 建築資訊模型與模型視圖。

3.4.1 基礎 IFC 建築資訊模型

一個基礎的 IFC 建築資訊模型有幾個 Entity，包含:IfcProject、IfcSite、IfcBuilding、IfcBuildingStorey 及建築元件，其關係如圖 3-4，分別介紹如下:

IfcProject

表示一個建築專案的相關資訊，它為 IFC 建築資訊模型最上層的 Entity，每個模型必須且只能擁有一個 IfcProject。它定義了幾個重要資訊: 預設單位(例如設定長度的單位為公尺)、世界座標系統(絕對座標)、數值的精確度。

IfcSite

表示一個工址的相關資訊，包含包含其位置(相對於絕對座標)，位址的地形描述、經緯度及高程、郵件地址。IfcProject 與 IfcSite 為組合與分解關係，使用 IfcRelAggregates 描述，一個 IfcProject 能夠由許多個 IfcSite 組成。

IfcBuilding

表示一棟建築物的相關資訊，包含其位置(相對於 IfcSite 的位置)、外觀、建築物高度等資訊。IfcSite 與 IfcBuilding 為組合分解關係，使用 IfcRelAggregates 描述，一個 IfcSite 能夠由許多個 IfcBuilding 組成。

IfcBuildingStorey

表示一個樓層的相關資訊，包含位置(相對於 IfcBuilding 的位置)、外觀、樓層高(相對於 IfcBuilding 的高度)。IfcBuilding 與 IfcBuildingStorey 為組合分解關係，使用 IfcRelAggregates 描述，一個 IfcBuilding 能夠由許多個 IfcBuildingStorey 組成。

IfcBuildingElement

定義一個具有實體建築元件，它是一個抽象資料類型，所有 Share Building Elements(資訊交換層)的 Entity 都是他的子類別，包含: IfcBeam(樑)、IfcColumn(柱)、IfcMember(桿件)、IfcDoor(門)、IfcWall(牆)、IfcSlab(板)、IfcStair(樓梯)、IfcRoof(屋頂)、IfcPile(椿)、IfcFooting(基礎)等。IfcBuildingElement 與 IfcSite、IfcBuilding、IfcBuildingStorey 為空間連結關係，使用 IfcRelContainedInSpatialStructure 描述其關係。

IfcSite、IfcBuilding、IfcBuildingStorey 都繼承自 IfcSpatialStructureElement，它們可以用 CompositionType 屬性將其進一步分解，這個屬性有三個狀態“COMPLEX”、“ELEMENT”、“PARTIAL”，狀態為 COMPLEX 的 Entity 能夠分解成狀態為 ELEMENT 的 Entities；狀態為 ELEMENT 的 Entity 能夠分解成成狀態為 PARTIAL 的 Entities，如圖 3-5，ID 為#4 的 IfcBuilding 其 CompositionType 屬性為 ELEMENT，ID 為#6、#7 的 IfcBuilding 其 CompositionType 屬性為 PARTIAL，故 ID 為#4 的 IfcBuilding 可以藉由 IfcRelAggregates 表示其由 ID 為#4 及#6 的 IfcBuilding 所組成。

3.4.2 模型視圖

IFC 的規格涵蓋了 AEC/FM 領域，若對建築資訊模型沒有一個共同的視圖(view)很難達到資料交換的目的，以交換圖形資料為例: 一個軟體可以用點與線的 Entity 描述一個物品的外觀，而另一個軟體則是用一個參數化的 Entity 來描述，他們要進行資訊交換時還必需做轉換，所以定義共同的模型視圖(Model View)是資訊交換的第一步。IAI 提供基本驗證的方式即是判斷軟體的輸入及輸出是否符合軟體廠商提供的模型視圖，然而一個軟體的輸入或輸出完成驗證，但資訊不能在其它軟體分享則沒有多大意義，樊啟勇(2007)做過相關的 IFC 資訊交換的研究，即使通過 IFC 認證的軟體資訊交換的效率也是

很差。故 IAI 提出 IDM(Information Delivery Manual)[24]與 MVD(Model View Definition)[25]供軟體廠商實作 IFC 時參考。

IDM

目的在將建築專案分解為許多流程(process)，定義該流程所需的資訊與輸出的資訊，以及定義流程與流程間之的資料流以建立流程地圖(process map)，以釐清資訊交換的模型視圖。例如:結構分析這個流程需要桿件、節點、載重資料、材料性質，輸出節點位移的資料；進行估價的資料需要建築物設計與材料價單。目前 IDM 還屬於制定階段，許多流程都尚未完備，這部份可能要讓各大軟體廠商共同協商制定應是比較可行的。

MVD

提供一套定義模型視圖的方法論，包含模型視圖的描述格式、描述程序、及描述工具，IAI 官方網站有提供一些範本。MVD 目的在讓大家共同一種方式描述 IFC 建築資訊模型，如同使用 UML 進行物件導向設計。

3.5 IFC 實體檔案格式

為了讓使用 IFC 描述的資訊模型能夠在軟體間進行資訊交換，故需要一個電子檔案格式用於描述資訊模型。目前 IFC 的建築資訊模型有兩種主要的電子實體檔案格式: STEP 實體檔案(STEP Physical File)及 ifcXML 格式，市面上支援 IFC 的軟體主要以 STEP 實體檔案為主，亦為本研究使用的格式。實體檔案副檔名為 ifc，是以文字形式存在，主要分為兩個部份:標頭部份(Header)與資料部份(Data)。標頭部份包含一些檔案的基本資料，有檔案敘述、作者、IFC 版本等資料；資料部份則是建築資訊模型的實體。雖然檔案可以直接用文字編輯軟體開啟，但是當檔案很大時物件關係會相對的複雜，能用 G.E.M. Team Solutions[26]開發的 IfcQuickBrowser 方便查找檔案。以 IfcProject 為例，其實體檔案格式描述如下:

```
#97=IFCPROJECT('XYqLQNXnM0Wv+J5Xw/NdbQ',#107,$,$,$,$,$,(#96),#101);
#107=IFCOWNERHISTORY(#106,#102,.READWRITE.,.NOCHANGE.,0,$,$,0);
#106=IFCPERSONANDORGANIZATION(#105,#104,$);
#105=IFCPERSON($,$,'Pingting', $,$,$,$,$);
#104=IFCORGANIZATION($,$,$,(#103), $);
#103=IFCACTORROLE(.ENGINEER.,$,$);
#102=IFCAPPLICATION(#104,'0.1','project','54967');
#96=IFCGEOMETRICREPRESENTATIONCONTEXT('Pingting','Null',3,0.,#82,$);
#82=IFCAXIS2PLACEMENT3D(#71,#63,#64);
```

#71=IFCCARTESIANPOINT((0.,0.,0.));
#63=IFCDIRECTION((0.,0.,1.));
#64=IFCDIRECTION((1.,0.,0.));
#101=IFCUNITASSIGNMENT((#98,#99,#100));
#98=IFCSIUNIT(.LENGTHUNIT.,$,.METRE.);
#99=IFCSIUNIT(.AREAUNIT.,$,.SQUARE_METRE.);
#100=IFCSIUNIT(.PLANEANGLEUNIT.,$,.RADIAN.);

Entity 的表示法為 “#Number = EntityName(Attribute1, Attribure2, …);”，每個數字代表 Entity 物件的 ID，當 Entity 之間互相參考時即是使用這個符號，例如上述#97(IfcProject)的 OwnerHistory 屬性的內容為#107(IfcOwnerHistory)。Defined Type 與 Enumeration 資料型態的屬性內容都直接以文字表示，例如上述#97(IfcProject)的 GlobalId(IfcGloballyUniqueId – DefinedType)屬性的內容為'XYqLQNXnM0Wv+J5Xw/NdbQ'；#103(IfcActorRole)的 Role(IfcRoleEnum)屬性為內容為.ENGINEER.。對於 OPTIONAL 屬性，若其內容不存在，則使用“$”符號代替。

3.6 IFC 資訊存取應用程式介面

目前已經有不少 IFC 資訊存取的應用程式介面(API)，能夠使用程式語言呼叫這些 API 進行擷取及建立 IFC 檔案的資訊。在 IAI 官方網站[3]有列出許多開發工具，但大部分皆為商業軟體。為了方便進行學術研究 IFC 應用，本研究嘗試使用兩種兩種免費且容易取得的 API，IFCsvr 與 IFCEngineDLL，了解其對 IFC 檔案資訊存取的方式，各分別介紹於下面章節。

3.6.1 IFCsvr

IFCsvr 由 SECOM[27]公司開發，其核心為商業軟體 ST-Developer[28]，目前最新版本為 R300 支援到 IFC2x3。IFCsvr 提供一個機制存取/建立 IFC 實體檔案與 ifcXML。其為 ActiveX DLL 元件，能在 Windows 平台藉由 COM(Component Object Model)介面使用此軟體元件，使用此元件前必須先向作業系統登錄，理論上支援 COM 技術的程式語言皆可使用(Visual Basic 6.0、Visual C++ 6.0、VBA、.NetFramework 程式語言等)。IFCsvr 架構如圖 3-6，其分為幾個物件，分別介紹如下：

IFCsvr.R300

此物件控制 IFC 實體檔案的開啟與新增，使用 OpenDesign 方法開啟 IFC 檔案並產生 Design 物件；使用 NewDesign 方法建立 IFC 檔案並產生 Design 物件。

Design

此物件代表整個 IFC 實體檔案的建築資訊模型，包含所有的 Entity，其只能由 IFCsvr.R300 物件產生，Design 提供方法尋找 Entity 及 Entities 物件 :FindObject 、 FindObjects 、 FindObjectsByValue 、 FindObjectByP21Id 、 FindObjectByName；使用 Add 方法能夠新增 Entity 到 Design；用 Save 方法將 Design 的資料寫入 IFC 實體檔案。

Entities

此物件為 Entity 物件的集合體，必須由 Design 尋找 Entity 的方法取得，由 Count 屬性能夠得知其大小；使用 Item 方法可取得 Entity 物件。

Entity

此物件代表 IFC 建築資訊模型中的一個 Entity，它必須由 Design 物件尋找或新增 Entity 的方法、Entities 物件的 Item 方法或從 Attribute 物件取得。使用 Delete 方法可以將此 Entity 自 Design 中刪除；藉由 GetUsedIn 這個方法能夠找到參考它的 Entity; 其 Attributes 屬性可以取得 Attributes 物件。

Attributes

此物件為一個 Entity 所有屬性的集合，Count 屬性可以得知屬性的數量；使用 Item 方法能夠取得某 Attribute。

Attribute

此物件為 Entity 的某一個屬性，必須由 Attributes 的 Item 方法取得。當此屬性非集合體(SET、LIST)時用 Value 屬性取得或設定其值；若為集合體則用 GetItem 方法取得其內容，用 SetItem 方法設定其內容。

以下使用 VB.net 示範如何使用 IFCsvr 建立一個 IFC 檔案與 IfcPerson Entity。

```
Private svr As New IFCsvr.R300
Private design As IFCsvr.Design
design = svr.NewDesign("C:\sample.ifc", "IFC2x3")
Dim Person As IFCsvr.Entity
Person = design.Add("IfcPerson")
Person.Attributes("ID").Value = "001"
Person.Attributes("GivenName").Value = "Pingting"
Person.Attributes("FamilyName").Value = "Shen"
design.Save()
```

3.6.2 IFCEngineDLL & IFCEngineOCX

IFCEngineDLL 為 TNO[29]公司開發，目前最新版本為 1.01，提供學術研究及非營利用途免費，若要使用在商業軟體則必需付一些費用。其為 Windows 平台下的 DLL 元件，使用微軟 C++編譯器建置，使用此 API 最好的程式語言是 C++，因為其用到許多指標(pointer)來操作資料。IFCEngineDLL 根據 Standard Data Access Interface (SDAI, ISO 10303-22)設計，SDAI 定義存取 STEP 資料之 API 的格式，這些資訊存取的方式相當低階，使用上比較煩雜，但是其效率高，彈性也較大。以下依照功能介紹一些比較重要的函式。

檔案存取

sdaiOpenModelBN

用於開啟 IFC 檔，需指定檔案的 IFC EXPRESS schema，回傳 ModelID。

sdaiCreateModelBN

用於建立 IFC 檔案，需指定檔案的 IFC EXPRESS schema，回傳 ModelID。

sdaiSaveModelBN

將 Model 的資料寫入 IFC 檔案。

Entity 存取

sdaiCreateInstanceBN

在 Model 中建立一個 Entity，回傳 InstanceID。

sdaiDeleteInstance

刪除 Model 中的 Entity 物件。

sdaiGetEntity

從 Model 中藉由 Entity 名稱取得 Entity 物件，回傳 InstanceID。

sdaiGetEntityExtentBN

從 Model 中藉由 Entity 名稱取得一組 Entity 物件，回傳 aggregateID，再藉由 engiGetAggrElement 函式去取得 InstanceID。

屬性存取

sdaiGetAttrBN

藉由 InstanceID、屬性名稱及類型取得該屬性的內容。

sdaiPutAttrBN

藉由 InstanceID、屬性名稱及類型設定該屬性的內容。

以下用 C++示範如何使用 IFCEngineDLL 建立一個 IFC 實體檔案與 IfcPerson Entity。

```
int ModelID = sdaiCreateModelBN(0, "C:\sample.ifc", "IFC2X3_Final.exp");
int Person = sdaiCreateInstanceBN(ModelID, "IFCPERSON");
char Id[] = "001";
```

```
sdaiPutAttrBN(Person, "Id", sdaiSTRING, Id);
char given_name[] = "Pingting";
sdaiPutAttrBN(Person, "GivenName", sdaiSTRING, given_name);
char family_name[] = "Shen";
sdaiPutAttrBN(Person, "FamilyName", sdaiSTRING, family_name);
sdaiSaveModelBN(0, "C:\sample.ifc");
```

IFCEngineOCX 為根據 IFCEngineDLL 發展出來的 OLE 控制項 (Object Linking and Embedding controls)，能夠嵌在應用程式上，目的在提供顯示 IFC 3D 模型及較高階的 API(例如:點選一個元件取得 InstanceID)，TNO 公司的 IFCEngineView 軟體也是根據此 OCX 發展出來，由於使用 OLE 技術所以只能運用於微軟的開發平台。目前 IFCEngineOCX 是以公開部分原始碼的方式散佈，使用者能夠新增或是更改其功能。

IFCsvr 與 IFCEngineDLL 這兩種 API 對於 IFC 檔案資訊擷取建立的方式都相當接近，先藉由呼叫開啟檔案的方法將 IFC 檔案的資訊模型用一個資料結構保存(IFCsvr 為 Design 物件，IFCEngineDLL 以 ModelID 為索引指向那個資訊模型資料結構)，再使用一些方法從這個資料結構擷取或建立 Entity 資訊，最後將資料結構的內容以實體檔案格式輸出。

經本研究實際使用這些 API 後發現它們最主要的缺點在於將所有的 Entity 在程式設計階段都視為相同的資料類型(IFCsvr 為 Entity 物件，IFCEngineDLL 以 InstanceID 為索引指向那個 Entity 資料結構)，存取屬性時必須提供屬性的名稱才能取得該屬性的資料，並且所有屬性的資料也是使用相同的資料類型表示(IFCsvr 為 Object,，IFCEngineDLL 為 void*)，在程式設計階段無法做型別檢查，且不方便運用於物件導向程式設計。有鑑於此，本研究嘗試建立一套 Entity 資訊建立及擷取的方式，以便在程式設計階段能夠快速存取 Entity 屬性資料、並做型別的檢查以及運用物件導向程式設計於 BIM 系統開發，相關內容將於第四章說明。

第四章 IFC 類別庫

由研究探討，能了解 IFC 的規格與物件導向密不可分。本研究嘗試使用過前述 3.5 節的兩種免費 API 後發現在實作上並不適合用於物件導向程式設計。由於其對所有的 Entity 與屬性在程式設計階段都視為相同的資料類型，不方便從 IFC 規格的角度來建立建築資訊模型，並且在執行階段很可能發生資料型態的錯誤，例如誤將一個整數資料指定給某個字串屬性，在編譯時期並沒有辦法做檢核。

為了能夠更容易的對 Entity 資訊做擷取與建立，本研究以 IFCsvr 作為基礎並運用物件導向技術，自行撰寫程式將 IFC Entity 規格轉換為.Net Framework 類別(Class)，再以類別庫(Class Library)封裝。此類別庫提供一個接近 IFC 規格的 Entity 資訊存取方式，能夠從 IFC 規格的角度進行建立建築資訊塑模。並能夠運用物件導向於系統開發。本章內容包括:類別庫架構、如何將以 EXPRSS 描述的 IFC 資料類型映對到.Net Framework 資料類型。

4.1 類別庫架構

本研究根據 IFC 規格及 IFCsvr 為基礎進行類別庫設計，類別庫架構如圖 4-1。類別庫的設計遵循著 IFC 的規格，每個 IFC Entity 都映對到一個相同名稱的類別，並提供資訊存取介面。類別庫中每個類別可能繼承於其他類別，或是定義為抽象類別，當 IFC 規格改變時，類別庫只要做相對應類別的修正即可。由於處理 IFC 實體檔案的存取就必須耗費大量時間精力，故本研究仍利用 IFCsvr 進行基本的資訊存取，包含檔案開啟、檔案建立、搜尋 Entity 物件等。本研究以.Net Framework 為平台進行建置此類別庫，因此.Net Framework 上的程式語言皆能夠引用此類別庫。並使用 VB.net 物件導向程式語言進行描述類別，主要是語法精簡容易維護，且物件導向的功能完備，支援繼承、抽象資料類型、方法覆寫(Override)。

本研究所建立之類別庫中的類別都實作了 IfcEntity 介面。此介面為類別與 IFCsvr 溝通的橋樑，其定義 Entity 物件的建立、刪除與更新方法。此介面定義如下:

```
Public Interface IfcEntity
    Property Entity() As IFCsvr.Entity
    Function CreateEntity(ByRef design As IFCsvr.Design) As IFCsvr.Entity
    Function DeleteEntity() As Boolean
    Function UpdateEntity() As IFCsvr.Entity
End Interface
```

此介面定義了幾個成員: Entity 屬性，存放此類別的 IFCsvr.Entity 物件，每個 IFCsvr.Entity 物件保存著各自的資訊；CreateEntity 方法，藉由呼叫此方法於資訊模型(IFCsvr.Design)建立新的 Entity 物件(IFCsvr.Entity)，若此物件已經存在於模型時則對該物件進行更新；DeleteEntity 方法，將此 Entity 物件自資訊模型中移除；UpdateEntity 方法，將此 Entity 物件的資訊更新到模型。

4.2 IFC 資料類型映對

為了能夠在程式設計階段根據 IFC 規格進行資訊存取，必須將 EXPRESS 描述的 IFC 資料類型轉換為.Net Framework 的資料型態，本章節介紹如何將 Defined Type、Enumeration、SelectType 及 Entity 四種 IFC 資料類型映對(mapping)到.Net Framework 的資料型態，並由這些映對的規則建立類別庫。

4.2.1 Defined Type

Defined Type 的映對相當容易，它只是對一個基礎的資料型態提供一個別名，我們只要直接對那個基礎的資料型態做映對就好，例如，IfcText、IfcLabel 這兩個 Defined Type 的基礎的資料型態是 STRING 能夠直接映對到 VB.net 的 String 類別。將各種 IFC 的基礎資料型態與 VB.net 類別的資料型態映對整理如表 4-1，其中 REAL 及 NUMBER 都用於表示數字，差別是 NUMBER 可能代表整數或實數，本研究都將其映對到 Double；LOGICAL 的值可以是 TRUE、FALSE 或 UNKNOWN，本研究將其與 BOOLEAN 都映對到 Boolean；BINARY 的值可能是 0 或 1，本研究將其映對到 Byte。

4.2.2 Enumeration

.Net Framework 允許定義 Enum 資料結構，故 Enumeration 能夠直接映對道相對應的 Enum 資料結構。從 Enumeration 的 EXPRESS 描述直接轉換為 VB.net 的語法的 Enum 資料結構只有少部份語法不同，利用正則表示式(Regular expression)便能夠快速的進行語法轉換，最後將所有轉換好的 Enum 放置於 Enumerations 類別。轉換範例如圖 4-2，左半邊為 Enumeration 的 EXPRESS 描述，右半部為轉換為 Vb.net 語法的 Enum 資料結構，在 Vb.net 的每個 Enum 資料結構額外增加了一個 EMPTY 狀態，表示此 Enum 屬性的狀態未設定。在轉換過程中最主要的問題在於 EXPRESS 定義的 Enumeration 內容與 Vb.net 程式語言的關鍵字有衝突，例如: IfcStateEnum 其中有一個內容為 READONLY，而它剛好是 Vb.net 語言的關鍵字，這時我們在內容的結尾加上“_”避免衝突，在做資訊擷取及建立時再做判斷，把“_”移除。

4.2.3 Select Type

Select Type 是一種比較特殊的資料型態，它屬於已定義的資料型態之一，在擷取資訊時因為沒有固定的資料型態，很容易發生執行時期的錯誤，例如前述 3.1 節提到的 IfcSimpleValue 這個 Select Type，假設一個軟體將它用於保存字串資料，而另一個軟體在讀取時並沒有辦法確定它是字串或是其他資料類型。 Select Type 主要參考 C/C++的 Union 資料類型設計，.Net Framework 沒有辦法建立這種資料結構，對於這種類型的映對方法根據其內容定義的資料型態分為兩種:

1. Defined Type
若定義的內容為 Defined Type 則選擇其中一種 Defined Type 進行映對，例如，IfcSimpleValue 可以是 IfcInteger(整數)、IfcReal(實數)、IfcBoolean(布林值)、IfcIdentifier(字串)、IfcText(字串)、IfcLabel(字串)、IfcLogical(邏輯符號)其中之一。

2. Entity
若定義的內容為 Entity，則建立一個與此 Select Type 相同名稱的的介面，裡面不實作任何內容，而它內容所定義的 Entity (映對到 Class，於下節介紹)必須實作這個介面。以 IfcAxis2Placement 為範例，如圖 4-3，這個 Select Type 具有兩種資料類型(都是 Entity): IfcAxis2Placement2D 與 IfcAxis2Placement3D，這兩種都必須實作 IfcAxis2Placement 介面，如此一來便可藉由參考這個介面達到多型，也就是 IfcAxis2Placement 能夠指向 IfcAxis2Placement2D 或 IfcAxis2Placement3D。

4.2.4 Entity

Entity 為 IFC 最複雜的資料型態，其也是用以表示資訊最主要的方式。Entity 很明顯必須映對到類別(Class)，但除了將 EXPRESS 定義的 Entity 轉換為 Class，也要考慮到跟 IFCsvr 的繫結，才能建立資訊於實體檔案。類別的映對比較複雜，其中有幾個原則:

1. 每個 Entity 都映對到一個相同名稱的 Class，並根據 IFC 定義的繼承關係繼承父類別，若 Entity 為抽象資料型態 Class 則必須加上 MustInherit 關鍵字，定義為抽象類別。

2. 每個類別都有一個 IFCsvr.Entity 物件屬性，利用這個物件將類別的資訊保存到資訊模型中，或是從資訊模型擷取資料。

3. 所有的類別都必須具有 IfcEntity 介面，藉此提供建立、刪除、更新 Entity 物件的能力。抽象類別由於無法建立實體，故不需要實作內容，但這些方法的修飾子(modifier)必須設為“MustOverride”，表示這個方法必須由子類別實現；非抽象的類別必須實作內容，用“Overrides”修飾子表示其覆寫父類別的方法，而實作內容會隨著類別而異，例如 IfcProject 與 IfcBuilding 各擁有不同屬性，建立的物件時所填入的資訊也不同。

4. 所有類別都必須具有三個方法:DataAbstract，負責抽取 Entity 物件的資料並儲存於類別；DataStuff，負責將類別的資料填入 Entity 物件；DataUpdate，負責將類別的資料更新到 Entity 物件。這些方法的修飾子該為“Overrides”，表示其覆寫父類別的方法，這三個方法與 IfcEntity 介面及建構子有相當重要的關係，不同於 IfcEntity 介面的方法，就算是非抽象類別也必須實現這些方法的內容。

5. 每個 Class 提供兩個建構子(Constructor)，一個以 IFCsvr.Entity 為參數，目的在將 Entity 物件的資料儲存於類別；另一個不需要任何參數，可藉此建立新的 Entity 物件。

6. 根據 Entity 的每個屬性名稱及其資料型態一對一映對到類別，若資料型態為集合體則利用陣列表示。

7. 若 EXPRESS 定義屬性為 OPTIAONL，則在映對 Class 的屬性字尾加上“_Optional”。

9. Entity 的法則(Rule)不進行轉換。

以 IfcProject 這個 Entity 的映對為例，根據上述原則之編號對照如圖 4-4，並分別依照順序闡述如下：

1. Entity IfcProject 映對到相同名稱的類別“IfcProject”。由於 Entity IfcProject 繼承自 Entity IfcObject，故映對之類別同樣繼承 IfcObject 類別。

2. IfcProject 類別具有 IFCsvr.Entity 物件。

3. IfcProject 為非抽象類別故必須實作 IfcEntity 介面的內容，由於每個類別實作的內容都不同，故這邊必須使用覆寫(Overrides)的方式實作這些方法的內容。

4. 實作資料處理的方法，這些方法的權限為 “Protected”表示其只能為同類別的方法或是子類別的方法呼叫。

5. 定義 IfcProject 類別的兩個建構子。

6. 將 IfcProject Entity 的所有屬性，按照其資料型態映對到 IfcProject 的類別中。LongName 屬性資料型態為 IfcLabel(Defined Type)映對到 String；Phase 屬性資料型態為 IfcLabel(Defined Type)映對到 String；RepresentationContexts 屬性資料型態為 IfcRepresentationContext (Entity)集合體，映對到 IfcRepresentationContext 類別陣列；UnitsInContext 屬性為 IfcUnitAssignment(Entity)映對到 IfcUnitAssignment 類別。

7. LongName 及 Phase 這兩個屬性在 EXPRESS 定義為 OPTIONAL，在轉換到類別後在其名稱後面加上“_OPTIONAL”。

IfcProject Class 藉由帶參數的建構子擷取 IFCsvr.Entity 物件的資訊，這個建構子實作內容如下:

```
Public Sub New(ByRef _Entity As IFCsvr.Entity)
    If _Entity Is Nothing Then
        Exit Sub
    ElseIf _Entity.Type = "IfcProject" Then
        myEntity = _Entity
        DataAbstract(myEntity)
    End If
End Sub
```

建構子先判斷 Entity 是否存在，再檢核 Entity 名稱是否為 IfcProject，若名稱正確再呼叫 DataAbstract 方法進行 Entity 資料擷取，DataAbstract 實作內容如下:

```
Protected Overrides Sub DataAbstract(ByRef _Entity As IFCsvr.Entity)
    MyBase.DataAbstract(_Entity)
    LongName_Optional= _Entity.Attributes("LongName").value
    Phase_Optional= _Entity.Attributes("Phase").value
    …
End Sub
```

IfcProject 的 DataAbstract 方法先藉由呼叫父類別的 DataAbstract 方法(MyBase.DataAbstract)進行父類別屬性的擷取，再將將本身的屬性從 Entity 物件擷取出來(LongName、Phase 等屬性)。以這個例子而言其父類別為 IfcObject，藉由呼叫 IfcObject 的 DataAbstract 進行 IfcObject 所有屬性的資訊擷取。IfcObject 的 DataAbstract 方法如下:

```
Protected Overrides Sub DataAbstract(ByRef _Entity As IFCsvr.Entity)
        MyBase.DataAbstract(_Entity)
        ObjectType_Optional= _Entity.Attributes("ObjectType").value
End Sub
```

IfcObject 只有一個 ObjectType 屬性，在擷取這個屬性的資訊之前一樣先呼叫父類別的 DataAbstract 方法，依此類推直到繼承的頂端為止，如此一來就可以將 IfcProject 所有屬性資料擷取出來，由父類別的 DataAbstract 方法將繼承而來屬性資訊擷取出來。呼叫順序如圖 4-5，從 IfcProject 開始一直到 IfcRoot 再遞迴到 IfcProject。

IfcEntity 界面的 CreateEntity 方法用於在資訊模型建立 Entity 物件，其實作方式與建構子相當類似，藉由呼叫 DataStuff 將資料填入 Entity 物件。IfcProject 的 CreateEntity 方法如下:

```
Public Overrides Function CreateEntity(ByRef design As IFCsvr.Design) As IFCsvr.Entity
    If myEntity Is Nothing Then
        myEntity = design.Add("IfcProject")
        DataStuff(myEntity, design)
        Return myEntity
    ElseIf (design.FindObject(myEntity.oid) Is Nothing) Then
        myEntity = design.Add("IfcProject")
        DataStuff(myEntity, design)
        Return myEntity
    Else
        Return UpdateEntity()
    End If
End Function
```

CreateEntity 在呼叫時必須傳入 IFCsvr.Design 當作參數，表示此欲建立 Entity 物件的資訊模型。若 myEntity 屬性沒有內容，表示此類別尚未建立 Entity 物件，則直接於資訊模型新增 Entity 物件(design.Add)，若 Entity 物件存在，則

會檢查此 Entity 物件是否存在於欲建立此物件的資訊模型，若不存在則建立，存在則對資料進行更新。藉由 IFCsvr.Design 建立 Entity 物件後，呼叫 DataStuff 方法將類別的屬性填入 Entity 中。IfcProject 的 DataStuff 方法內容如下:

```
Protected Overrides Sub DataStuff(ByRef _Entity As IFCsvr.Entity, ByRef design As IFCsvr.Design)
    MyBase.DataStuff(_Entity, design)
    _Entity.Attributes("LongName").vlue = LongName_OptionalŠ
    _Entity.Attributes("Phase").value = Phase_Optional
    …
End Sub
```

IfcProject 的 DataStuff 方法先呼叫父類別的 DataStuff 方法，再將自己的屬性填入 Entity 物件，藉由父類別的 DataStuff 方法將繼承而來屬性資訊填入 Entity 物件中，IfcProject 父類別 IfcObject 的 DataStuff 方法內容如下：

```
Protected Overrides Sub DataStuff(ByRef _Entity As IFCsvr.Entity,ByRef design As IFCsvr.Design)
        MyBase.DataStuff(_Entity,design)
        _Entity.Attributes("ObjectType").value=ObjectType_Optional
End Sub
```

IfcObject 的 DataStuff 方法一樣先呼叫父類別的 DataStuff，再將自己的屬性資料填入 Entity 物件，藉此就可以將 IfcProject 所有屬性資料填入 Entity 物件。IfcProject 的 DataStuff 呼叫順序與 DataAbstract 方法一樣從 IfcProject 開始一直到 IfcRoot 再遞迴到 IfcProject。

為了更有效率的將 IFC Entity 轉換為類別，本研究撰寫一個類別轉換器將 EXPRESS 的 Entity 轉換為類別。最後將轉換好的類別以類別庫方式封裝成.Net Framework 組件(Assembly)，以 DLL 形式存在，讓其他程式專案能夠參考引用此類別庫，部份已轉換的類別見圖 4-6，類別庫的使用範例及應用將於第五章進行介紹。

第五章 IFC 類別庫使用範例與應用

為顯示本研究所建立 IFC 類別庫在 IFC 資訊存取上更為便利並容易運用物件導向於系統開發之特色。本章介紹 IFC 類別庫之使用範例及應用。範例部份包含如何使用類別庫進行 IFC Entity 資訊的建立及擷取，並另外使用 IFCsvr 建立及擷取同樣資訊以作為類別庫的對照，並介紹如何運用類別庫於物件導向程式設計。應用部份，本研究先建立一個以 IFC 為基礎的 Plug-In 架構 BIM 系統，讓不同功能的軟體能夠以 Plug-In 元件方式動態載入系統執行，並共享同一個資訊模型，再運用本研究所建立的 IFC 類別庫進行 Plug-In 元件開發，更進一步示範如何運用 IFC 類別庫於軟體開發。

5.1 IFC 類別庫使用範例及與 IFCsvr 之比較

本節介紹如何使用本研究建立的 IFC 類別庫，示範如何利用類別庫進行建立、擷取 IFC 資訊，另外使用 IFCsvr 擷取同樣資訊以作為類別庫的對照，對 IFC Entity 資訊存取的方式做比較。範例以 VB.net 進行撰寫，經過實際測試 IFC 類別庫也適用於其他.Net Framework 程式語言。

5.1.1 IFC 資訊建立

使用類別庫建立 Entity 物件步驟如下:

1. 使用不帶參數的建構子，建立該類別的物件
2. 將資料填入該物件的屬性中
3. 呼叫該物件的 CreateEntity 方法

以下範例使用類別庫建立 IfcOwnerHistory Entity 物件並輸出為 IFC 實體檔案。

```
design = svr.NewDesign("C:\sample.ifc", "IFC2x3")
Dim actor As New IfcActorRole
actor.Role = Enumerations.IfcRoleEnum.ENGINEER
Dim person As New IfcPerson
person.ID_Optional = "54967"
person.FamilyName_Optional = "Shen"
person.GivenName_Optional = "Pingting"
Dim org As New IfcOrganization
org.Name = "Lin Lab"
ReDim org.Roles_Optional(1)
org.Roles_Optional(1) = actor
```

```
Dim app As New IfcApplication
app.ApplicationDeveloper = org
app.ApplicationFullName = "IFCool"
app.ApplicationIdentifier = "IFCool"
app.Version = "v4"
Dim person_org As New IfcPersonAndOrganization
person_org.TheOrganization = org
person_org.ThePerson = person
Dim OwnerHistory As New IfcOwnerHistory
OwnerHistory.ChangeAction = Enumerations.IfcChangeActionEnum.NOCHANGE
OwnerHistory.CreationDate = 0
OwnerHistory.State_Optional = Enumerations.IfcStateEnum.READWRITE
OwnerHistory.OwningApplication = app
OwnerHistory.OwningUser = person_org
OwnerHistory.CreateEntity(design)
design.Save()
```

使用 IFCsvr 為對照，建立與上述相同的 Entity 資訊並輸出為實體檔案，範例如下:

```
Dim Actor As IFCsvr.Entity
Actor = design.Add("IfcActorRole")
Actor.Attributes("Role").Value = "Engineer"
Dim Person As IFCsvr.Entity
Person = design.Add("IfcPerson")
Person.Attributes("ID").Value = "001"
Person.Attributes("GivenName").Value = "Pingting"
Person.Attributes("FamilyName").Value = "Shen"
Dim Organization As IFCsvr.Entity
Organization = design.Add("IfcOrganization")
Organization.Attributes("Name").value = " Lin Lab"
Organization.Attributes("Roles").SetItem(Actor, 1)
Dim Application As IFCsvr.Entity
Application = design.Add("IfcApplication")
Application.Attributes("Version").Value = "v4"
Application.Attributes("ApplicationDeveloper").value = Organization
Application.Attributes("ApplicationFullName").value = "IFCool"
Application.Attributes("ApplicationIdentifier").value = "IFCool"
```

```
Dim Org_Person As IFCsvr.Entity
Org_Person = design.Add("IfcPersonAndOrganization")
Org_Person.Attributes("ThePerson").value = Person
Org_Person.Attributes("TheOrganization").value = Organization
Dim OwnerHistory As IFCsvr.Entity
OwnerHistory = design.Add("IfcOwnerhistory")
OwnerHistory.Attributes("ChangeAction").value = "nochange"
OwnerHistory.Attributes("CreationDate").value = 0
OwnerHistory.Attributes("State").value = "readwrite"
OwnerHistory.Attributes("OwningApplication").value = Application
OwnerHistory.Attributes("OwningUser").value = Org_Person
design.Save()
```

IfcOwnerHistory 用於描述一個 Entity 的物主資訊，包含是否可被修改、擁有此物件的組織及人員、建立物件的應用程式及物件建立/修改日期。上述類別庫與 IFCsvr 範例都先建立 IfcOwnerHistory 所參考的相關資訊，包括: IfcActorRole、IfcPerson、IfcOrganization、IfcApplication、IfcPersonAndOrganization，依序將資訊填入 Entity 物件中，最後再指定給 IfcOwnerHistory。上述使用類別庫與 IFCsvr 輸出相同的 IFC 實體檔案，如圖 5-1，DATA 部份為此 Entity 物件的實體。

由此實作範例，探討 IFC 類別庫與 IFCsvr 的資訊建立方式如下：

1. 屬性存取

   比起直接使用 IFCsvr 存取 Entity 資訊，使用類別庫於程式設計階段可以配合微軟的 IntelliSense 技術，使用 “.” 運算子直接瀏覽 Entity 的屬性並得知其資料型態，若為資料型態屬於 Enumeration 更能夠直接從選單中選擇內容，如圖 5-2，上半部為類別庫，下半部為 IFCsvr。IFCsvr 必須手動輸入屬性的名稱才能取得該屬性，若文字輸入錯誤則無法取得屬性，使用上相當不方便。此外使用類別庫能夠直接從類別屬性名稱得知該屬性是否為選擇性屬性(OPTIONAL)，使用 IFCsvr 必須查找 IFC 規格才能得知。

2. 資料型態

   IFCsvr 的 Entity 物件及屬性在程式設計階段各視為相同的資料型態(Entity 為 IFCsvr.Entity, 屬性為 Object)，故在程式設計階段並沒有辦法做型別的判別，將整數資料指定給字串屬性對 IFCsvr 而言是合法的，因為它們都具有 Object 的身份；而類別庫對於每一個 Entity 都映對到一個相同名稱的類別，並且類別的屬性必須合乎所屬的資料型態，多了這一層型別檢查，在程式設計階段就可以避免型別的錯誤，例如無法將整數資料指定給字串屬性，或是將 IfcApplication 物件指定給資料型態為

IfcPerson 的屬性。

為了更進一步示範使用 IFC 類別庫建立較複雜的資訊，以結構物常見的元件“樑”為例，首先必須知道如何使用 IFC 描述樑的資訊，我們建立一個樑的模型視圖如圖 5-3，這個模型視圖幫助釐清描述這個樑所要用到的 Entity，這些 Entity 闡述如下：

IfcBeam:用來描述樑的主體，包含樑的形狀、位置等資料。

IfcProductDefinitionShape:用來描述樑的各種形狀，IFC 定義一個產品能夠具有多種形狀，以樑為例，我們在建築設計時將樑視為具有空間的實體，而在結構分析時將樑視為節點與直線。

IfcShapeRepresentation:用來描述樑的形狀，定義其形狀類型。

IfcBoundingBox:用來描述一個矩形立方體形狀。

IfcLocalPlacement:用來描述樑的位置。

IfcAxis2Placement3D:描述樑的三維座標系統，亦為樑的區域座標系統(Local Coordinate System)。

IfcCartesianPoint:描述座標系統的原點。

IfcDirection:用來描述座標系統的 X 方向與 Z 方向。

IfcOwnerHistory:描述樑的擁有者資訊。

使用類別庫中跟上述 Entity 相同名稱的類別進行建立這個樑的實體，其範例如下：

```
Dim theBeam As New IfcBeam
Dim theShapes As New IfcProductDefinitionShape
Dim theShape As New IfcShapeRepresentation
Dim theBox As New IfcBoundingBox
Dim thePlace As New IfcLocalPlacement
Dim theCoord As New IfcAxis2Placement3D
theBeam.GlobalId = "001"
theBeam.OwnerHistory = OwnerHistory
theBeam.Representation_Optional = theShapes
theBeam.ObjectPlacement_Optional = thePlace
ReDim theShapes.Representations(1)
theShapes.Representations(1) = theShape
ReDim theShape.Items(1)
theShape.Items(1) = theBox
theBox.Corner = New IfcCartesianPoint
theBox.XDim = 15
theBox.YDim = 20
theBox.ZDim = 300
```

```
thePlace.RelativePlacement = theCoord
theCoord.Location = New IfcCartesianPoint
theCoord.Axis_Optional = New IfcDirection(0, 0, 1)
theCoord.RefDirection_Optional = New IfcDirection(1, 0, 0)
theBeam.CreateEntity(design)
```

其 IFC 實體檔案見圖 5-4，在上面範例當中根據前述的模型視圖建立一個樑的實體，其具有自己的位置、形狀等資料，使用類別庫可以很容易並以相當直覺的方式建立這些資訊。

5.1.2 IFC 資訊擷取

擷取 Entity 物件資訊的步驟如下

1. 取得該 Entity 物件
2. 呼叫所屬類別的建構子，並以該 Entity 物件為參數
3. 從類別產生的物件進行擷取資訊

以下示範運用類別庫於 Entity 物件資訊的擷取，以擷取 IfcOwnerHistory 的部分資訊資訊為例。

```
design = svr.OpenDesign("C:\sample.ifc")
Dim own As New IfcOwnerHistory(design.FindObjects("IfcOwnerHistory").Item(1))
Console.WriteLine("Person ID:" + own.OwningUser.ThePerson.ID_Optional)
Console.WriteLine("Person Full Name:" + own.OwningUser.ThePerson.FamilyName_Optional +
   own.OwningUser.ThePerson.GivenName_Optional)
Console.WriteLine("Organization Name:" + own.OwningUser.TheOrganization.Name)
Console.WriteLine("Organization Actor:" + own.OwningUser.TheOrganization.Roles_Optional(1).Role.ToString)
Console.WriteLine("Application:" + own.OwningApplication.ApplicationFullName + own.OwningApplication.Version)
Console.WriteLine("State:" + own.State_Optional.ToString)
```

完全使用 IFCsvr 截取相同的資訊，對照如下:

```
Dim own As IFCsvr.Entity
own = design.FindObjects("IfcOwnerHistory").Item(1)
Dim person_org As IFCsvr.Entity = own.Attributes("OwningUser").Value
Dim app As IFCsvr.Entity = own.Attributes("OwningApplication").Value
Dim org As IFCsvr.Entity = person_org.Attributes("TheOrganization").Value
Dim person As IFCsvr.Entity = person_org.Attributes("ThePerson").Value
Dim actor As IFCsvr.Entity = org.Attributes("Roles").GetItem(1)
```

```
Console.WriteLine("Person ID:" + person.Attributes("ID").Value)
Console.WriteLine("Person Full Name:" + person.Attributes("FamilyName").Value +
    person.Attributes("GivenName").Value)
Console.WriteLine("Organization Name:" + org.Attributes("Name").Value)
Console.WriteLine("Organization Actor:" + actor.Attributes("Role").Value)
Console.WriteLine("Application:" + app.Attributes("ApplicationFullName").Value +
    app.Attributes("Version").Value)
Console.WriteLine("State:" + own.Attributes("State").Value)
```

上述範例使用類別庫與 IFCsvr 輸出相同的資訊，如圖 5-5，顯示 IfcOwnerHistory 的擁有人 ID、姓名、組織名稱、組織的角色、應用程式名稱、以及讀寫狀態。在使用類別庫的類別擷取資訊前，如前面提到的第一個步驟，必須使用 IFCsvr 提供的方法在資訊模型中找到該 Entity 物件，如上述例子我們使用 FindObjects 方法尋找所有為 “IfcOwnerHistory”的 Entity 物件，再利用 IfcOwnerHistory 類別的建構子產生 IfcOwnerHistory 物件。

比起 IFCsvr 必須先分別取得 IfcOwnerHistory 所參考的 Entity 屬性，才能進一步擷取資訊，類別庫在物件初始化時即將其所參考的 Entity 使用類別庫的類別初始化，故能直接從物件的成員擷取資訊，如圖 5-6，可以直接用“.”運算子先取得 IfcOwnerHistory 的 OwningUser 屬性資料(資料型態為 IfcPersonAndOrganization)，並繼續取得 ThePerson 屬性(資料型態為 IfcPerson)，最後取得其 ID 屬性(資料型態為 String)，如前一節所述，類別庫在屬性存取上更方便快速，並且能夠做資料型別的檢查。

綜合前兩節所做的類別庫與 IFCsvr 在使用上的比較整理如表 5-1，類別庫提供了一套更快速、安全、容易的 Entity 資訊存取方式，並且 IFC 類別庫更容易運用物件導向於程式設計，此部份將於下面章節介紹。

5.2 IFC 類別庫於物件導向程式設計之運用

本研究建立之類別庫另一個優點為適合應用於物件導向程式設計，由於其根據 IFC 的規格設計，在實作上面能夠以更貼近規格的方式進行建築資訊塑模。本節使用兩個簡單的範例，示範如何使用類別庫運用繼承及多型於 IFC 實作。

5.2.1 繼承

所有類別庫中的類別都能夠被繼承，藉此擴展該類別的能力。以

IfcRectangleProfileDef 這個描述矩形斷面的類別為例，它具有 Xdim 跟 Ydim 兩個屬性用以描述長與寬，可以定義一個新類別“MyRectangleProfile”繼承它並新增兩個屬性:斷面積與慣性矩,當擷取資訊時能夠直接從該物件取得斷面積與慣性矩。並新增一個建構子，讓類別在實體化時能夠直接設定長寬，範例如下:

```
Class MyRectangleProfile
    Inherits IfcRectangleProfileDef

    Sub New(ByVal Xdim As Double, ByVal Ydim As Double)
        Me.XDim = Xdim
        Me.YDim = Ydim
        Me.ProfileType = Enumerations.IfcProfileTypeEnum.AREA
        Me.Position = New IfcAxis2Placement2D
    End Sub

    Public ReadOnly Property Area()
        Get
            Return Me.XDim * Me.YDim
        End Get
    End Property

    Public ReadOnly Property Moment_Of_Inertia()
        Get
            Return (1 / 12) * Me.XDim * Me.YDim ^ 3
        End Get
    End Property
End Class
```

當進行資訊建立時可以直接使用 MyRectangleProfile 類別的建構子設定其父類別(IfcRectangleProfileDef)的 XDim 及 Ydim 屬性，並定義斷面類型為實心斷面，及其區域二維座標系統。當這個類別被實體化為物件後能夠藉由 Area 及 Moment_Of_Inertia 這兩個屬性取得這個斷面的面積與慣性矩。

IFCsvr 由於將所有的 Entity 都視為相同的物件，在程式設計時無法如上例使用繼承擴充其內容。 使用 IFC 類別庫可以將問題的處理集中在該類別本身，並且可以重複使用這些繼承類別，無論在程式的分解或是維護上都較為便利。

5.2.2 多型

IFC 規格以物件導向為基礎，在設計上運用了不少多型的概念，本研究所提出 IFC 類別庫在使用上能夠遵循 IFC 規格的設計實現多型。例如前述 3.2.2 節有提到 IfcExtrudedAreaSolid，其 SweptArea 屬性的資料型態為 IfcProfileDef 抽象類別，所有具有 IfcProfileDef 身份的非抽象類別皆能夠被參考，類別庫在使用上同樣具有這個規則，以使用類別庫建立 IfcExtrudedAreaSolid 這個 Entity 為例進行說明如下：

```
Dim extruded_area As New IfcExtrudedAreaSolid
extruded_area.SweptArea = New IfcCircleProfileDef
extruded_area.SweptArea = New IfcTShapeProfileDef
extruded_area.SweptArea = New IfcIShapeProfileDef
extruded_area.SweptArea = New MyRectangleProfile(10, 15)
extruded_area.CreateEntity(design)
```

當指定一個物件給其 SweptArea 屬性時，這個物件必須具有 IfcProfileDef 身份，上述將 IfcCircleProfileDef(圓形斷面)、IfcTShapeProfileDef(T 型斷面)、IfcIShapeProfileDef(I 型斷面)，甚至是上一節自己定義的 MyRectangleProfile 指定給 SweptArea 都是可行的，由於它們都有直接或間接繼承 IfcProfileDef。當我們對 IfcExtrudedAreaSolid 物件呼叫 CreateEntity 方法建立 Entity 物件，其在將資料填入 Entity 物件時會呼叫 SweptArea 屬性的 CreateEntity 方法，這個方法由於被子類別覆寫(Override)，它實際上呼叫的是繼承 IfcProfileDef 之子類別的 CreateEntity 方法，可能是圓形、T 型、I 型或其他具有 IfcProfileDef 身份的斷面。

在程式設計階段無法將不具有 IfcProfileDef 身份的物件指定給 SweptArea，例如將一個 IfcBoundingBox 物件指定給 SweptArea，程式即無法通過編譯，並會出現如圖 5-7 之型別轉換錯誤訊息，因此在程式設計時期便能夠基於 IFC 的規格作多型的檢核，而 IFCsvr 由於其屬性的值都是 Object 故沒有辦法直接檢核。

由以上兩個範例得知，本研究所提出之 IFC 類別庫在物件導向程式設計的支援度上比 IFCsvr 高。藉由繼承、多型，程式開發成員能夠以更直覺的方式分解程式，讓程式更容易擴充、維護。並且，由於類別庫屬於的類別屬於.Net Framework 的類別，其能夠利用.Net Framework 所提供的一些機制進行程式設計，例如:使用 Reflection 動態取得類別的資訊，或是實作介面，此部份將會在下章節做介紹。

5.3 IFC 類別庫應用

為了更進一步應用本研究建置的類別庫進行建築資訊系統的開發，本研究先開發一個 Plug-In 架構系統。Plug-In 架構系統與一般的系統最大的差別在於其能夠藉由 Plug-In 元件擴充系統的功能，並且不同 Plug-In 元件之間能夠藉由系統進行互動、影響，不同功能的 Plug-In 元件能夠動態載入系統執行，並共享資訊。接著，再運用本研究所建立的類別庫進行 Plug-In 元件開發，示範如何運用前述章節提到之 IFC 類別庫的特色進行開發 IFC 應用程式。

本節先介紹 Plug-In 系統的架構，再介紹一些已開發的 Plug-In 元件及類別庫的應用方式。

5.3.1 Plug-In 架構 IFC 資訊系統

本研究開發的 Plug-In 架構系統以 IFC 建築資訊模型為基礎，其提供了 IFC 檔案的輸入與輸出，以及基本的 IFC 資訊瀏覽。開發人員能夠將不同資訊需求的軟體建置為 Plug-In 元件，使用者再依照不同需求分別載入系統執行，並共享一個建築資訊模型。例如結構工程師可能需要結構分析、模型瀏覽元件而估價人員需要數量計算、材料檢視元件，我們只要針對各種功能元件進行開發，主要系統並不需要做更動，藉由 Plug-In 的架構更容易簡化開發的流程。

本系統的架構如圖 5-8，包含四個部份:主應用程式、Plug-In Manager、Plug-In 介面、Plug-In 元件，分別敘述如下：

主應用程式

主應用程式提供一個圖形使用者介面(圖 5-9)，讓使用者建立、開啟、儲存 IFC 檔案，及執行 Plug-In 元件。視窗左半部顯示 IFC 檔案的樹狀結構視窗，當開啟一個 IFC 檔案，如圖 5-10 會產生一個以 IfcProject 為樹根節點的樹狀結構，並根據 IfcRelAggrates(組合分解)及 IfcRelContainedInSpatialStructure(空間連結)兩個 Entity 將建築資訊模型展開來。此樹狀結構為系統與 Plug-In 元件最重要的溝通方式，當某個樹狀節點被點選，系統會通知所有執行中的 Plug-In 元件這個事件以及被點選的 Entity 物件。右半部為 Plug-In 元件的工作視窗。

Plug-In Manager

Plug-In Manager 是 Plug-In 元件的容器，它藉由 Plug-In 介面來管理這些 Plug-In 元件，當它接收主應用程式發出的事件，會通知容器內所有執行狀態的 Plug-In 做相對應的動作 ，這些事件包括:檔案開啟、新開檔案、檔案儲存、

建築資訊模型改變以及樹狀節點被選取。當 Plug-In 元件狀態發生改變時，Plug-In Manager 會通知主應用程式這個訊息。

Plug-In 介面

Plug-In 介面定義 Plug-In 元件的資訊及針對各種事件的反應，所有 Plug-In 都必須實作這個介面才能夠載入系統中執行。它定義了幾個方法：RunPlugIn，當主應用程式執行此 Plug-In 元件時這個方法會被呼叫；StopPlugIn，當主應用程式停止此 Plug-In 元件時這個方法會被呼叫；OpenIfc，當主應用程式開啟 IFC 檔案時此方法會被呼叫；NewIfc，當主應用程式建立新 IFC 檔案時此方法會被呼叫；SaveIfc，當主應用程式儲存 IFC 檔案時此方法會被呼叫；ReactDesignChanged，當建築資訊模型被修改時此方法會被呼叫；ReactSelectTreeNode，當主應用程式的樹狀節點被點選時此方法會被呼叫。

Plug-In 元件

Plug-In 元件必須實作 Plug-In 介面才能載入 Plug-In 系統中執行，可以將不同功能的軟體開發成 Plug-In 元件，再按照不同需求載入執行。每個 Plug-In 可能有幾種狀態：Ready，表示此 Plug-In 元件已成功載入，隨時可以啟動；Running，表示此 Plug-In 元件為執行狀態；Stopped，表示此 Plug-In 元件被使用者停止，目前為停止狀態，能夠重新啟動；Error，表示此 Plug-In 元件在執行中發生錯誤，這類 Plug-In 無法在重新啟動。最後開發好的 Plug-In 元件必須編譯為.Net Framework DLL 組件，才能被 Plug-In 系統載入。

5.3.2 3D 建築模型瀏覽 Plug-In 元件

為了能夠瀏覽 IFC 檔案的 3D 模型，本研究使用 IFCEngineOCX[29]開發了一個能夠瀏覽 3D 建築模型的 Plug-In 元件，並提供使用者基本的瀏覽功能，包含：拉近、拉遠、旋轉、平移視角等功能。此元件不需要使用到類別庫，因為主要的工作都是藉由 IFCEngineOCX 處理。

這個 Plug-In 元件主要處理幾個事件，當接收到啟動 Plug-In、開啟 IFC 檔案、儲存 IFC 檔案的事件時更新該 IFC 檔案的 3D 模型；當接收到點選主系統的樹狀節點事件時，會根據所點選節點的 GlobalId 尋找 Entity，若該 Entity 存在於 3D 模型當中，則將該 3D Entity 元件以顯著顏色標示；當使用者對 3D Entity 元件點兩下，則會藉由尋找該 Entity 是否在主應用程式的樹狀結構中，若在該樹狀結構中，該樹狀節點將會被選取，這時其他執行中的 Plug-In 元件會收到樹狀節點選取的事件，藉由這個動作可以利用 3D 視覺化方式跟其他執行中的 Plug-In 元件產生互動。

此 Plug-In 元件執行畫面如圖 5-11 與 5-12，預設的瀏覽方式是藉由滑鼠以拖曳的方式旋轉視點，點選 Plug-In 元件視窗上的 “Zoom”按鈕可將瀏覽模式改為縮放模式，藉由滑鼠拖曳拉近或拉遠視角；“Rotate”按鈕為旋轉模式；“Pan”按鈕為平移模式；“Front View”按鈕將視角還原；“Set InVisible”按鈕能夠將點選的 3D Entity 元件隱藏起來；“Show All”按鈕以顯示所有 3D Entity 元件。

5.3.3 建築物框架建模 Plug-In 元件

本節示範用本研究建立的 IFC 類別庫建立一個建築物框架建模 Plug-In 元件，提供建立樑、柱的功能並能夠以視覺化的方式設定其位置、用參數化的方式設定其斷面。依照元件開發流程、使用者介面、元件操作範例分別說明如下。

元件開發流程

此元件的開發流程如下：

1. 建立建築資訊模型的模型視圖

本元件的目的在於能夠從無到有建立一個建築物的框架，並且此建築物是以 IFC 進行描述其建築資訊模型，故首先要建立這個建築資訊模型的模型視圖。為了能夠建立一個完整有效的 IFC 建築資訊模型，本研究以 3.4.1 節提到的基礎 IFC 建築資訊模型為架構並參考 ArchiCad 10 輸出的 IFC 檔案，設計一個屬於本元件的建築物模型視圖，其架構如圖 5-13，在樑柱的形狀描述上使用 3.2.2 節提到的“IfcExtrudedAreaSolid” Entity，且使用參數化的方式描述其斷面。本元件在進行塑模的過程皆以這個模型視圖為基礎，此模型視圖幫助我們了解在塑模過程中需要用到哪些 Entity(類別)，並可以根據這些類別更進行思考如何設計這個元件的架構。

2. 元件的架構規劃

此元件的架構如圖 5-14 可以分為以下三個部份：

事件處理

此部份目的在處理 Plug-In 系統所送出的事件，此元件必須處理幾個事件：

a. 啟動 Plug-In 元件事件

當接收到這個事件，此元件必須根據前面提到的模型視圖，建立一個建築資訊模型，以這個模型為基礎，讓使用者新增樓層、樑、柱於此模型。

b. 新建 IFC 檔案事件

當接收到新建 IFC 檔案事件，必須如啟動事件新增一個基本的建築資訊模型。

c.點選樹狀結構節點事件

本元件在建模的過程是以樓層為基礎，當接收到點選樹狀結構節點的事件時必須判斷是否為樓層節點(IfcBuildingStorey)，再進行相對應的處理。

圖形使用者介面

此部份讓使用者能夠以視覺化的方式建立建築物框架，這部份主要包含兩個視窗：畫布視窗與斷面選擇視窗，畫布視窗目的在展現某一層的樓的樑、柱方位，並讓使用者能夠直接從畫布視窗上新增樑柱；斷面選擇視窗目的在於讓使用者能夠選擇該樑柱的斷面及設定相關屬性。

建築資訊模型管理

此部份目的在於管理此元件建立的建築資訊模型，當使用者藉由畫布視窗新增樑柱或是新增一個樓層，都是由這個部份管理，它負責新增資訊到建築資訊模型中、紀錄目前的樓層與樑柱數目、將資訊模型初始化等工作。

3. 元件實作

根據前述的架構使用 VB.net 程式語言實作這個元件：

事件處理:由一個實作 Plug-In 介面的類別處理，並實作 RunPlugIn、NewIfc、ReactSelectTreeNode 這三個方法的內容，分別處理執行 Plug-In、開啟 IFC 檔案、點選樹狀節點的事件。

圖形使用者介面:使用兩個表單類別設計，分別為畫布視窗與斷面選擇視窗。

建築資訊模型管理:這個部份建立一個"IfcCreator"類別進行處理，這個類別根據此元件的模型視圖進行設計，它負責建立、管理建築資訊模型，其完整實作內容參考附件一，我們使用本研究建立的類別庫的類別進行建立 IFC Entity 資訊，其中樑柱 Entity 物件建立上運用了前述 5.2.2 節類別庫多型範例提到的 IfcProfileDef 進行斷面多型的處理。樑柱的形狀都是用 IfcExtrudedAreaSolid 描述，斷面選擇確定後會產生一個所選擇斷面類型的物件(選擇矩形則為 IfcRectangleProfileDef，選擇圓形為 IfcCircleProfileDef)當於畫布新增樑或是柱時，會將此斷面物件當做樑柱斷面傳送給新增樑柱的函式。新增樑的函式如下：

```
Public Function AddBeam(ByVal idxStorey As Integer, ByVal X1 As Double, ByVal Y1 As Double, ByVal X2 As Double, ByVal Y2 As Double, ByVal Height As Double, ByRef Profile As IfcProfileDef) As IFCool.IfcBeam
```

上述建立樑的函式傳入樓層的索引、起點、終點、樑的高度、最後一個參數 Profile 以 IfcProfileDef 當作參考，當選擇完斷面後建立的物件可能是 IfcRectangleProfileDef 或是 IfcCircleProfileDef 但由於其都具有 IfcProfileDef 介面故都能被參考，如此一來在斷面的擴充上就顯得相當容

易，我們可以新增更多的斷面，只要他們都具有 IfcProfileDef 的身份。

元件操作範例

此元件的操作步驟如下:

1. 點選 Plug-In 系統的[檔案] -> [新建 IFC] ，如圖 5-15，這時 Plug-In 系統會建立一個空的 IFC 建築資訊模型。

2. 點選 Plug-In 系統的[模組]->[建築物框架建模 Plug-In]->[啟動]，如圖 5-16，這時 Plug-In 系統會進入啟動狀態，此 Plug-In 會在上個步驟建立的資訊模型中建立基本的資訊，包含：IfcProject、IfcSite、IfcBuilding 以及一個預設的 IfcBuildingStorey，其高度為 0。啟動此 Plug-In 後隨即可以見到畫布視窗如圖 5-17。

3. 點選 Plug-In 元件的“樓層”按鈕後會出現新增樓層對話視窗，如圖 5-18 所示，輸入樓層高度(並非樓層的位置高度)按下確定後即可在建築資訊模型中增加一個 IfcBuildingStorey，也能夠直接由 Plug-In 視窗的樹狀結構視窗見到該樓層，如圖 5-19。

4. 接下來選擇樓層以建立樑柱，藉由點選樹狀結構的 IfcBuildingStorey 節點，選擇目前畫布視窗所在的樓層，此 Plug-In 會將該樓層的樑柱繪於畫布視窗。點選本元件建立的預設樓層節點，由於目前尚未新增任何樑柱，畫布視窗上會顯示二維的座標平面，表示可以開始於此畫布上新增樑柱，參考圖 5-20。

5. 樑柱的斷面選擇方式相同，按下畫布視窗的“樑”或“柱”按鈕會出現斷面選擇視窗，如圖 5-21 所示，點選視窗上方的標籤能夠切換所選擇的斷面，內定可以選擇矩形(IfcRectangleProfileDef)、圓形(IfcCircleProfileDef)、I 型(IfcIShapeProfileDef)、T 型(IfcTShapeProfileDef)四種斷面，並以參數化的方式設定尺寸。接下來設定樑高度或是柱的長度(相對於該層樓)，按下確定按鈕完成選擇後可以開始於畫布視窗上新增樑柱。

6. 柱的位置設定直接於畫布視窗上按下滑鼠左鍵即可，畫布視窗右上文字方塊會即時顯示目前的位置，按下左鍵後會顯示一個正方形方塊以表示該柱位置。藉由步驟 5 的斷面選擇視窗，設定柱的斷面形狀為圓形斷面，半徑 20 公分，輸入柱的高度為 2.8 公尺，於畫布上建立柱的實體如圖 5-22，柱之間的水平間隔為 3.5 公尺與 2.5 公尺，垂直間隔皆為 2 公尺。

7. 樑的位置設定必須指定起點與終點，藉由在畫布視窗上按下左鍵決定起點再拖曳滑鼠放開決定終點，樑在畫布上會以一黑色直線表示。藉由步驟5的斷面選擇視窗，設定樑的斷面為矩形斷面，X方向長度為20公分, Y方向長度為 25 公分，樑的高度為 2.8 公尺，新增樑於該樓層如圖 5-23 所示，樑的起點與終點皆與柱相連結。樑柱也能夠用手動的方式設定位置，於畫布視窗上方顯示位置的文字方塊輸入數值，再按下 “ADD”按鈕新增樑柱。
8. 選擇樓層 2，同 6、7 步驟方式新增樑柱於畫布上，如圖 5-24 所示，樑柱的斷面與第一樓層使用相同設定，柱之間的水平間隔為 3.5 尺，垂直間隔為 2 公尺。
9. 點選 Plug-In 系統的[檔案] -> [儲存]，Plug-In 會將目前的 IFC 建築資訊模型寫入 IFC 檔案，完成框架建模。當儲存完畢 Plug-In 系統會更新樹狀結構，此時便可以看見樹狀節點的參考編號，如圖 5-25，前幾個步驟的樹狀結構之節點編號都為 0 表示其尚未寫入 IFC 實體檔案，當模型的資料由記憶體寫入實體檔案後每個 Entity 物件都會具有其 ID。

該範例輸出檔參考附件二。可以啟動 “3D 建築模型瀏覽 Plug-In 元件”來觀看這個建築模型的 3D 外觀，如圖 5-26，藉由載入不同的 Plug-In 元件，擴充 Plug-In 系統的功能。由於此輸出檔根據 ArchiCad 10 的模型視圖進行設計因此也能夠載入 ArchiCad 10 中編修如圖 5-27。

5.3.4 樑柱斷面資訊瀏覽 Plug-In 元件

本節示範應用類別庫建立一個瀏覽樑柱斷面資訊的 Plug-In 元件，此元件能夠瀏覽樑柱斷面的斷面類型、斷面積、斷面周長、慣性矩。

元件架構

此 Plug-In 元件架構如圖 5-28，其分為三個部份：

1. 事件處理

   此 Plug-In 元件只需要處理點選樹狀結構節點的事件，判斷該節點是否為樑(IfcBeam)或柱(IfcColumn)再將 Entity 資訊交給斷面處理部份。這部份使用一個實作 Plug-In 的類別來處理。
2. 斷面資訊處理

   這個部份要先取得該樑或柱描述形狀的 Entity，本 Plug-In 元件只處理以 “IfcExtrudedAreaSolid”作為形狀描述的樑柱，並進一步擷取其描述斷面的 Entity，計算其斷面積、周長、慣性矩。
3. 資訊展示介面

   將處理完的斷面資訊到以視窗介面顯示。

斷面資訊處理實作

在這部份的實作上運用本研究提出的類別庫及.Net Framework 的 Reflection 與介面功能於斷面資訊處理。首先定義一個斷面屬性介面如下：

```
Public Interface IProfileProperty
    ReadOnly Property Profile_Type() As String
    ReadOnly Property Profile_Area() As Double
    ReadOnly Property Profile_Perimeter() As Double
    ReadOnly Property Profile_I() As Double
End Interface
```

此介面定義了所需要擷取的資料，包含斷面類型、斷面積、斷面周長、慣性矩，由於每個斷面都具有這些屬性，但是這些屬性的計算方式或內容會隨著不同斷面而異，故可以將這些屬性獨立出來成為一個介面，無論是什麼斷面只要實作這個介面，便可以用這個介面當作參考取得其屬性。

接下來自定義斷面類別並實作上述介面，而這個斷面類別必須是從類別庫繼承而來，並且名稱定義為 “My + 所繼承的斷面名稱”，例如定義 “MyIfcCircleProfileDef”類別其繼承(inherits)自“IfcCircleProfileDef” ，並且實作(implements) “IProfileProperty”這個介面，其部份實作內容如下:

```
Public Class MyIfcCircleProfileDef
    Inherits IfcCircleProfileDef
    Implements IProfileProperty

    Public ReadOnly Property Profile_Area() As Double Implements IProfileProperty.Profile_Area
      Get
          Return Math.PI * Me.Radius ^ 2
      End Get
    End Property

    Public ReadOnly Property Profile_I() As Double Implements IProfileProperty.Profile_I
      Get
          Return Math.PI * .Radius ^ 4 / 4
      End Get
    End Property
    …
End Class
```

當 Plug-In 元件接收到選取樹狀節點的事件時，會先判斷是否為樑柱節點，再根據描述斷面的 Entity 物件，利用 Reflection 機制動態取得類別的資料，若斷面的 Entity 物件為 IfcCircleProfileDef 則要用 MyIfcCircleProfileDef 類別將其實體化。實作如下：

```
Dim profileProperty As IProfileProperty
Dim profileType As Type = Reflection.Assembly.GetAssembly(Me.GetType).GetType("My" + ProfileEntity.Type)
Dim args(0) As IFCsvr.Entity
args(0) = ProfileEntity
profileProperty = Activator.CreateInstance(profileType, args)
```

上述 ProfileEntity 為斷面的 Entity 物件，利用 Reflection 機制尋找名稱為“My+斷面名稱”的類別，再以這個 Entity 物件為參數實體化這個類別，由於這個類別有實作 IProfileProperty 介面，故可以直接使用這個介面取得該斷面的資料，如此一來只要專心在斷面類別的實作就可以了，可以陸續建立 MyIfcRectangleProfileDef、MyIfcTShapeProfileDef 等類別而不用修改資料擷取部份的程式碼。

元件操作

此 Plug-In 元件啟動後只要點選 Plug-In 系統之樹狀結構的樑(IfcBeam)或柱(IfcColumn)節點便可以瀏覽其斷面資訊，如圖 5-29 所示，選取 IfcBeam(#22)後，能夠在斷面瀏覽視窗直接顯示該樑的斷面資訊。也能夠開啟 3D 模型瀏覽 Plug-In 直接點取 3D 物件，藉此選取樹狀節點，顯示該樑或柱的斷面資訊，如圖 5-30 所示，先點選一個 3D 物件，被點選的 3D 物件會以醒目的金黃色顯示，並且自動選取表示該物件的樹狀節點 IfcColumn(#93)，接著斷面瀏覽視窗會顯示該柱的斷面資訊，能夠藉由 3D 視覺化的方式讓 Plug-In 之間進行互動。

5.3.5 小結

經過實際使用本研究提出的 IFC 類別庫進行 IFC 應用程式開發，證實該類別庫確實能夠提供一個更安全與更容易的資訊存取方式並能夠運用物件導向於程式開發。然而，IFC 類別庫在存取 IFC 檔案時尚須依賴 IFCsvr 做基本的處理，若能夠自行發展一個較低階的 API 跟類別庫相互搭配，相信在效能與彈性都能更加提昇。此外，本研究提出的 Plug-In 架構系統運用在 IFC 應用程式的開發上，能夠讓不同 Plug-In 元件之間共享一個建築資訊模型。可以將不同功能的軟體開發為 Plug-In 元件，藉由互動完成工作，在軟體的維護及開發上都更為便利。

第六章 結論與建議

6.1 結論

1. 本研究根據 IFC 規格與 IFCsvr，設計一套接近規格的類別庫，方便程式設計人員開發 IFC 應用程式，此類別庫能夠更快速的建立 Entity 物件、擷取 Entity 物件的資料，類別庫能夠以更高階的方式實作 IFC，並更容易將物件導向應用於程式開發，讓程式開發人員能專注於問題的解決。

2. 經過實際比較，本研究所提出的類別庫確實在 IFC Entity 資訊擷取與建立上比 IFCsvr 更直覺，也更容易使用，並且更容易運用於物件導向程式設計。

3. 本研究開發一個 Plug-In 架構 IFC BIM 系統，使用者能依照不同需求及目的將 Plug-In 元件動態載入系統中執行，並共享同一份 IFC 檔案。運用此 Plug-In 架構能夠簡化 IFC 應用程式開發流程，只要實作 Plug-In 介面的方法並專注於實現 Plug-In 功能即可很容易的開發 Plug-In 元件。

4. 本研究進一步的示範如何運用所建置的類別庫於開發 Plug-In 元件。使用類別庫及運用物件導向的繼承、多型、Reflection 等機制開發了建築物框架建模與樑柱斷面資訊瀏覽 Plug-In 元件。

5. 本研究建置的類別庫並未完全遵循 IFC 規格，主要是 IFC 規格中的 Select Type 資料類型具有多重的身分，很可能這筆資料一開始是數字，執行到一半變成字串，在執行過程非常容易發生錯誤，目前類別庫的處理方式為:內容為 Entity 的 Select Type 利用介面達到多型；內容是 Defined Type 則是映對到其中一種資料型態。

6. 本研究以物件導向觀點探討 IFC 規格與實作方法，IFC 的規格在設計上引用許多物件導向的概念(Entity、繼承、多型)，及 C/C++程式語言的特性(Union 對應到 Select Type, Enum 對應到 Enumerations 資料類型）。

6.2 建議

1. 本研究建置的 IFC 類別庫還有改善的空間，由於 IFC 規格的限制沒有辦法完全遵循，這邊值得再深入的研究是否有更好的解決辦法。以及 Entity 法則的部份，理論上應該能於類別中描述。

2. 本研究的類別庫使用 IFCsvr 作為基礎的 IO，在使用過程中由於需要呼叫其他 DLL，效率不是很好，最好能夠自行發展一套低階的 IFC 應用程介面式與 IFC 類別庫緊密結合，這部份需要花許多時間處理。

3. 本研究提出的 Plug-In 架構在許多地方仍然需要加強及改進，Plug-In 介面應能夠建立更多 Plug-In 間溝通的機制，這值得再進一步研究。

4. 期望本研究建置的 IFC 類別庫能被應用在其他研究或 IFC 應用程式開發上。

5. IFC 是一個公開的標準，不該被限定用特定的程式語言實作，應該發展成為更一般化的標準，讓支援物件導向的程式語言能夠以更貼近 IFC 規格的方式實作 IFC。IAI 對於 IFC 規格還有非常大的改進空間，例如:Select Type 這種資料類型應該謹慎使用，因為在實作上同一塊記憶體空間保存不同類型的資料，或是一個記憶體位址指向不同類型的資料型態都是不安全的作法。

6. 為了能夠更有效率的交換資料，IAI 應儘快與軟體廠商及各領域專家討論制定各領域資料交換的 IDM，而各軟體廠商應達成資訊交換的共識，如此一來才能夠實現 BIM 的精神。

7. 由於各國法規及施工習慣方法的不同，IFC 應該要考慮區域化的問題。各 IFC 應用程式介面的開發也應要能夠處理中文的資訊，目前發現使用 IFCsvr 寫入中文字會有亂碼的情形。

8. 台灣對 IFC 研究方興未艾，雖其他國家在 IFC 的運用有相當不錯的成果，建議對 IFC 的導入應該循序漸進，IFC 是一個好方案，但不一定是最好的。

參考文獻

[1] Ghang Lee, Rafael Sacks, Charles M. Eastman, “Specifying parametric building object behavior (BOB) for a building information modeling system”, Automation in Construction 15 758–776, 2006

[2] ISO/TC 184/SC4, http://www.tc184-sc4.org, 2001

[3] IAI International , http://www.iai-international.org/, 1995

[4] Building Smart, http://buildingsmart.org.au/, 1997

[5] IAI, “IFC Specification 2x3 Final”, http://www.iai-international.org/Model/files/IFC2x3_Final_HTML_distribution.zip , 2004

[6] IAI, “IFC2x Technical Guide”, http://www.iai-international.org/Model/documentation/IFC_2x_Technical_Guide.pdf , 2000

[7] IAI, “IFC Implementation Guide”, http://www.iai-international.org/Model/files/20040318_Ifc2x_ModelImplGuide_V1-7.pdf, 2004

[8] IAI ISG, http://www.iai.fhm.edu/, 2006

[9] NIST, http://cic.nist.gov/, 1996

[10] Vineet R. Kamat, Robert R. Lipman, “Evaluation of standard product models for supporting automated erection of structural steelwork”, Automation in Construction 16 232–241, 2007

[11] Ellis Horowitz, Sartaj Sahni, Dinesh Metha, Fundmentals of Data Structures in C++, 1994

[12] Wikipedia, “.Net Framework”, 2008

[13] Po-Han Chen, Caiyun Wan, Qizhen Yang, Seng Kiong Ting ,Lu Cui,Robert L.K. Tiong, “Implementation of IFC-based web server for collaborative building design between architects and structural engineers”, Automation in Construction 14 115–128, 2005

[14] Changfeng Fu , Ghassan Aouad, Angela Lee, Amanda Mashall-Ponting, Song Wu, “IFC model viewer to support nD model application”, Automation in Construction 15 178– 185, 2006

[15] M.Hassanien Serror, Junya Inoue, Yoshinobu Adachi, YozoFujino, “Shared computer-aided structural design model for construction industry”, Computer-Aided Design, 2007

[16] Ali Murat Tanyer, Ghassan Aouad, “Moving beyond the fourth dimension with an IFC-based single project database”, Automation in Construction 14 15–32, 2005

[17] 蘇建豪, “營建工程資訊管理之標準資訊模型初步研究”, 碩士論文, 國立台灣大學, 台北, 2000
[18] 潘稟嘉, “建築圖形資訊標準於營建業電子商務之應用研究”, 碩士論文, 國立台灣大學, 台北, 2000
[19] 蔡志偉, “IFC 建築資訊內容應用於結構分析資料擷取”, 碩士論文, 國立交通大學, 新竹, 2007
[20] 樊啟勇, “IFC 資料標準之結構物資訊擷取與建立”, 碩士論文, 國立交通大學, 新竹, 2007
[21] 曾國峰, “建築工程 3D 施工順序模擬系統之研究”, 碩士論文, 國立台灣大學, 台北, 2007
[22] 吳翌禎, “多維度工程專案資訊整合管理與視覺化之研究”, 博士論文. 國立台灣大學, 台北, 2007
[23] IAI, “The EXPRESS Definition Language for IFC Development”, http://www.iai-international.org/Model/documentation/The_EXPRESS_Definition_Language_for_IFC_Development.pdf, 2001
[24] IAI, “The Information Delivery Manual – IDM”, http://idm.buildingsmart.no/confluence/display/IDM/Home,2006
[25] IAI, “Model View Definition”, http://www.blis-project.org/IAI-MVD/, 2005
[26] G.E.M. Team Solutions, “IFC Quick Browser”, http://www.team-solutions.de/?page_id=19, 1999
[27] SECOM, “IFCsvr ActiveX Component”, http://tech.groups.yahoo.com/group/ifcsvr-users/, 2000
[28] STEP Tools inc. , “ST-Developer”, http://www.steptools.com/products/stdev/, 2008
[29] TNO, “IFC Engine DLL”, http://www.ifcbrowser.com/ifcenginedll.html, 2008

表 2-1 BIM 系統整理

所屬領域	BIM 系統名稱	開發公司
建築設計	Revit Architecture	Autodesk
建築設計	ArchiCad	GraphiSoft
建築設計	Bentley Architecture	Bentley
建築設計	DDS-CAD Building	DDS-CAD
建築設計	EliteCAD	Messerli Informatic GmbH
建築設計	MagiCAD	Progman
建築設計	Active3D	Active3D-Lab
建築設計	Allplan	Nemetschek
結構設計/分析	Bentley Structural	Bentley
結構設計/分析	Tekla Structures	Tekla
結構設計/分析	SCIA-ESAPT	SCIA
建築能源	RIUSKA	Olof Granlund Oy
設施管理	Bentley Building Mechanical Systems	Bentley
設施管理	Bentley Facilities Manager	Bentley
設施管理	DDS-CAD HVAC	DDS-CAD
設施管理	DDS-CAD Electrical	DDS-CAD
設施管理	DDS-CAD Plumbing	DDS-CAD
設施管理	Facility Online	Vizelia
建築資訊模型管理	EDMServer	EPM Technology

表 2-2 STEP/IFC/CIS2 比較

比較項目	STEP	IFC	CIS/2
制定組織	ISO	IAI	NIST
制定範圍	工業界(大範圍)	AEC/FM(中範圍)	鋼結構(小範圍)
是否公開標準	否	是	是
規格發展速度	較慢	較快	普通
規格描述語言	EXPRESS	EXPRESS	EXPRESS
資訊模型檔案格式	STEP 實體檔案	STEP 實體檔案	STEP 實體檔案
AEC/FM 軟體支援情形	沒有支援的軟體	許多軟體支援	許多軟體支援

表 4-1 IFC 基礎資料型態映對到 VB.net 資料型態

IFC Type	VB.net Type
REAL	Double
INTEGER	Integer
NUMBER	Double
STRING	String
BOOLEAN	Boolean
LOGICAL	Boolean
BINARY	Byte

表 5-1 IFCsvr 與 IFC 類別庫資訊存取比較

	IFCsvr	IFC 類別庫
能否快速存取成員	否	是
是否有作型別檢查	否	是
能否判斷 OPTIONAL 成員	否	是
物件導向支援能力	差	佳
使用容易度	中	佳

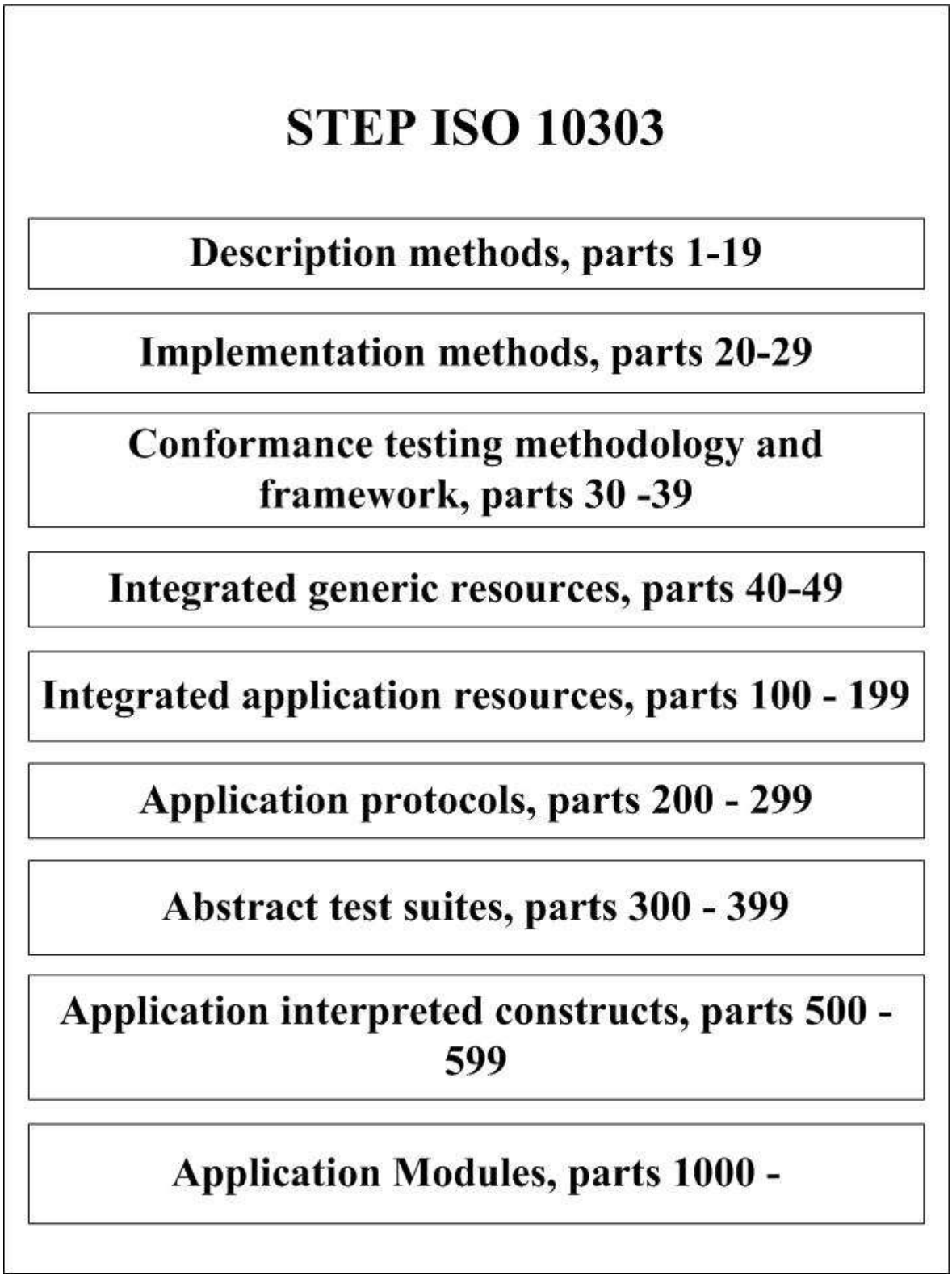
STEP ISO 10303
Description methods, parts 1-19
Implementation methods, parts 20-29
Conformance testing methodology and framework, parts 30 -39
Integrated generic resources, parts 40-49
Integrated application resources, parts 100 - 199
Application protocols, parts 200 - 299
Abstract test suites, parts 300 - 399
Application interpreted constructs, parts 500 - 599
Application Modules, parts 1000 -

圖 2-1 STEP 架構

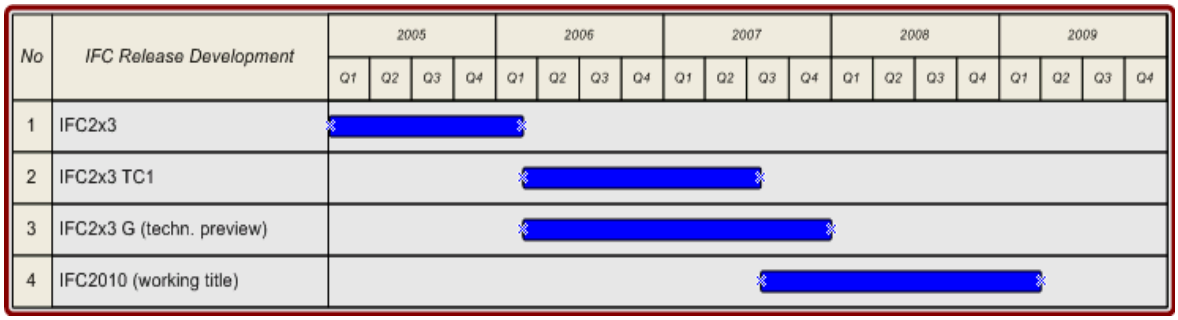

圖 2-2 IFC 發展進度[3]

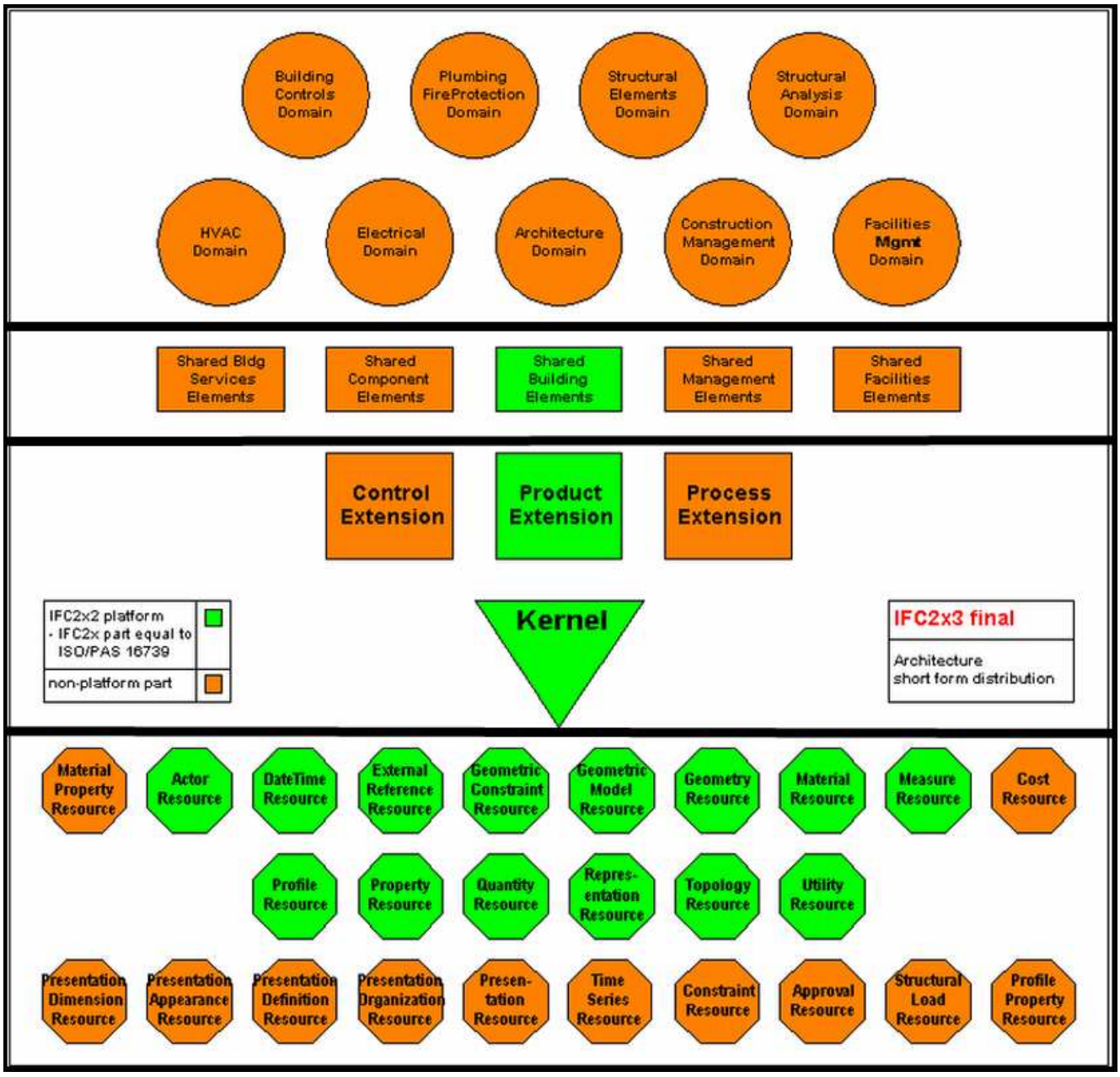

圖 2-3 IFC2x3 Final 架構[6]

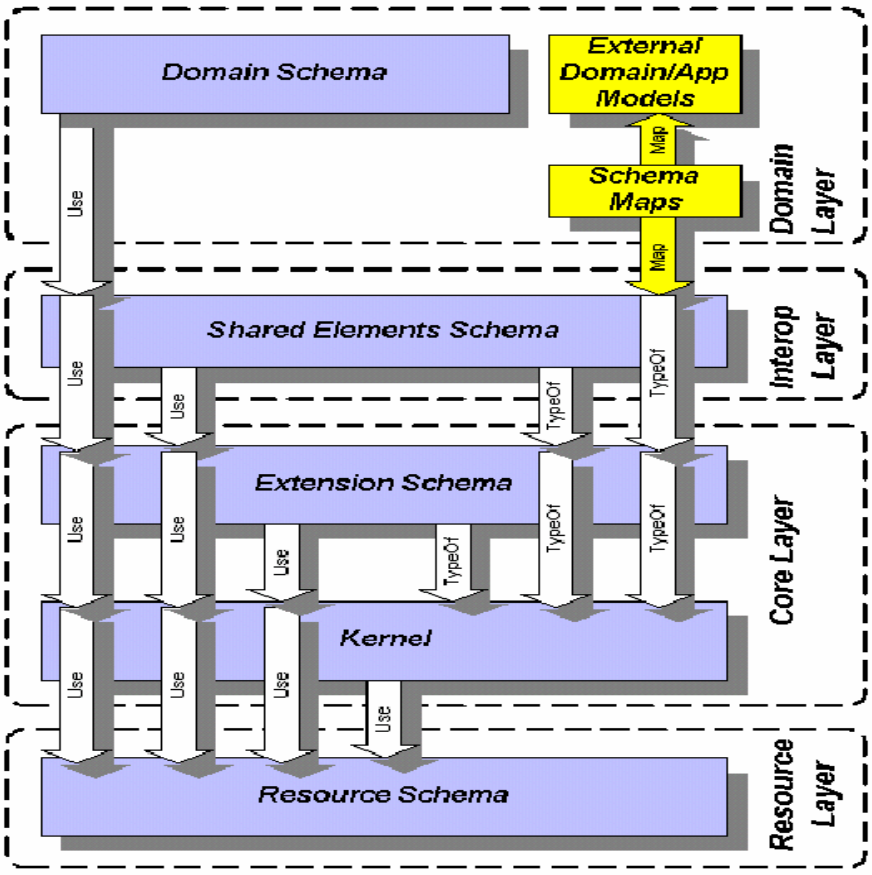

圖 2-4 IFC 參考原則[7]

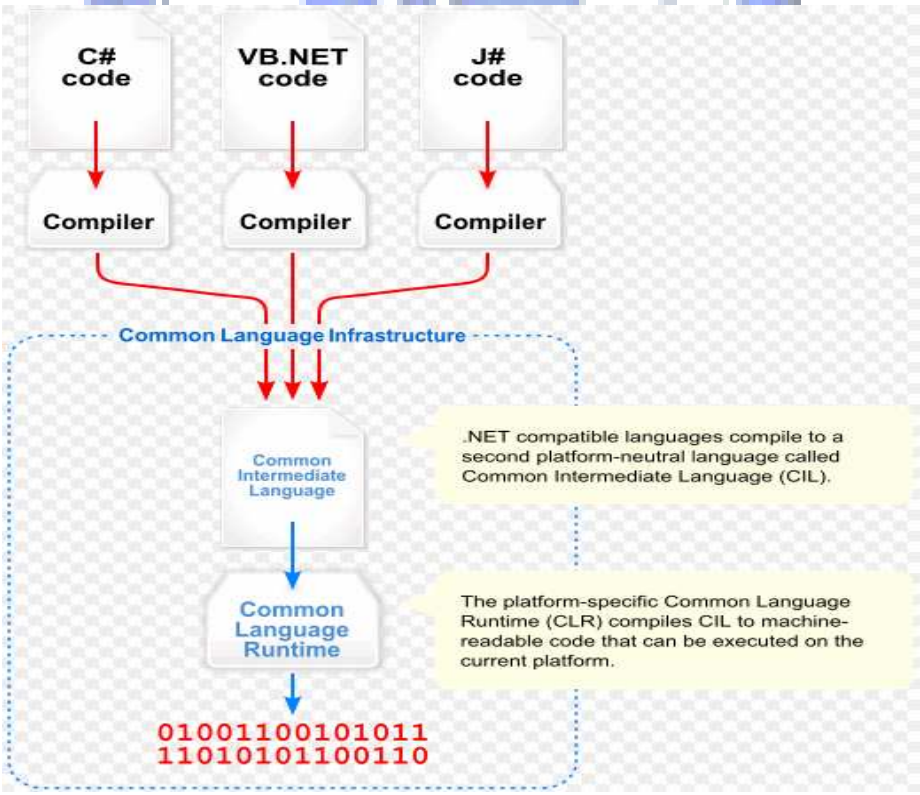

圖 2-5 .Net Framework CLI 架構[12]

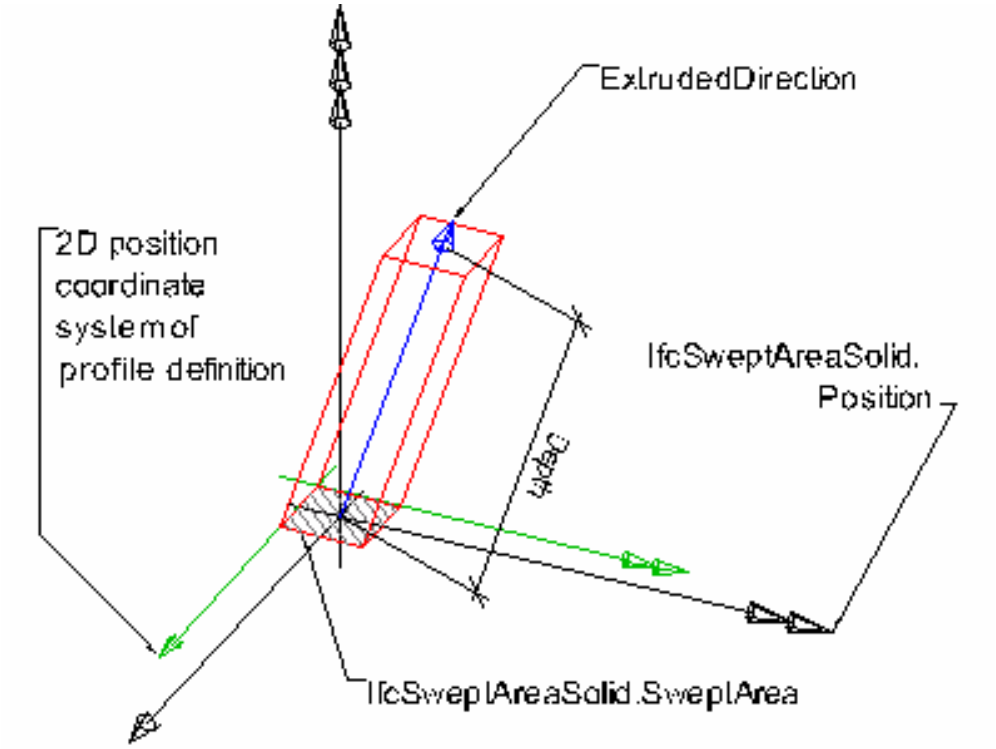

圖 3-1 IfcExtrudedAreaSolid 形狀[5]

圖 3-2 IfcProfileDef 繼承關係

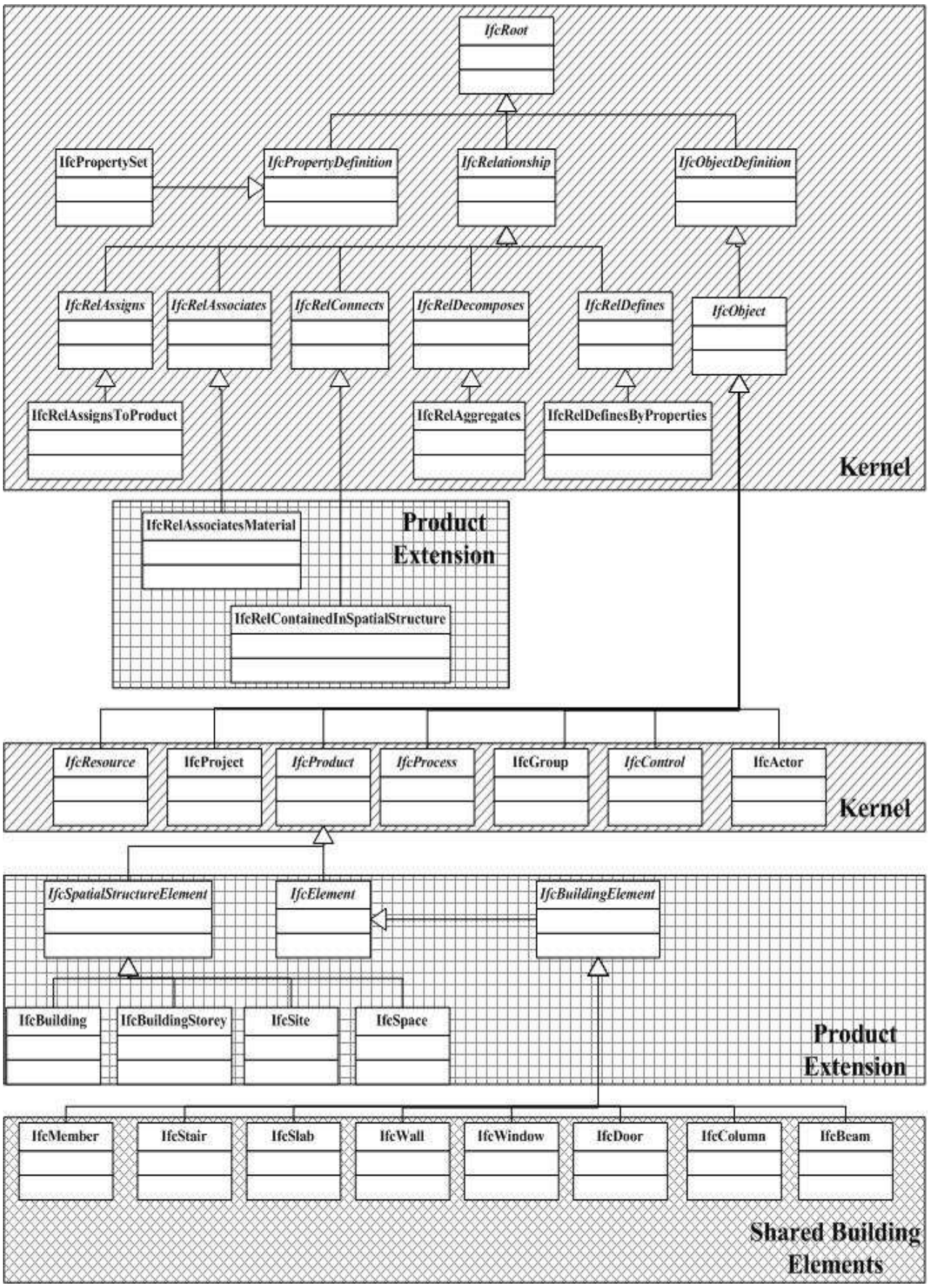

圖 3-3 IFC 基礎 Entity 繼承關係

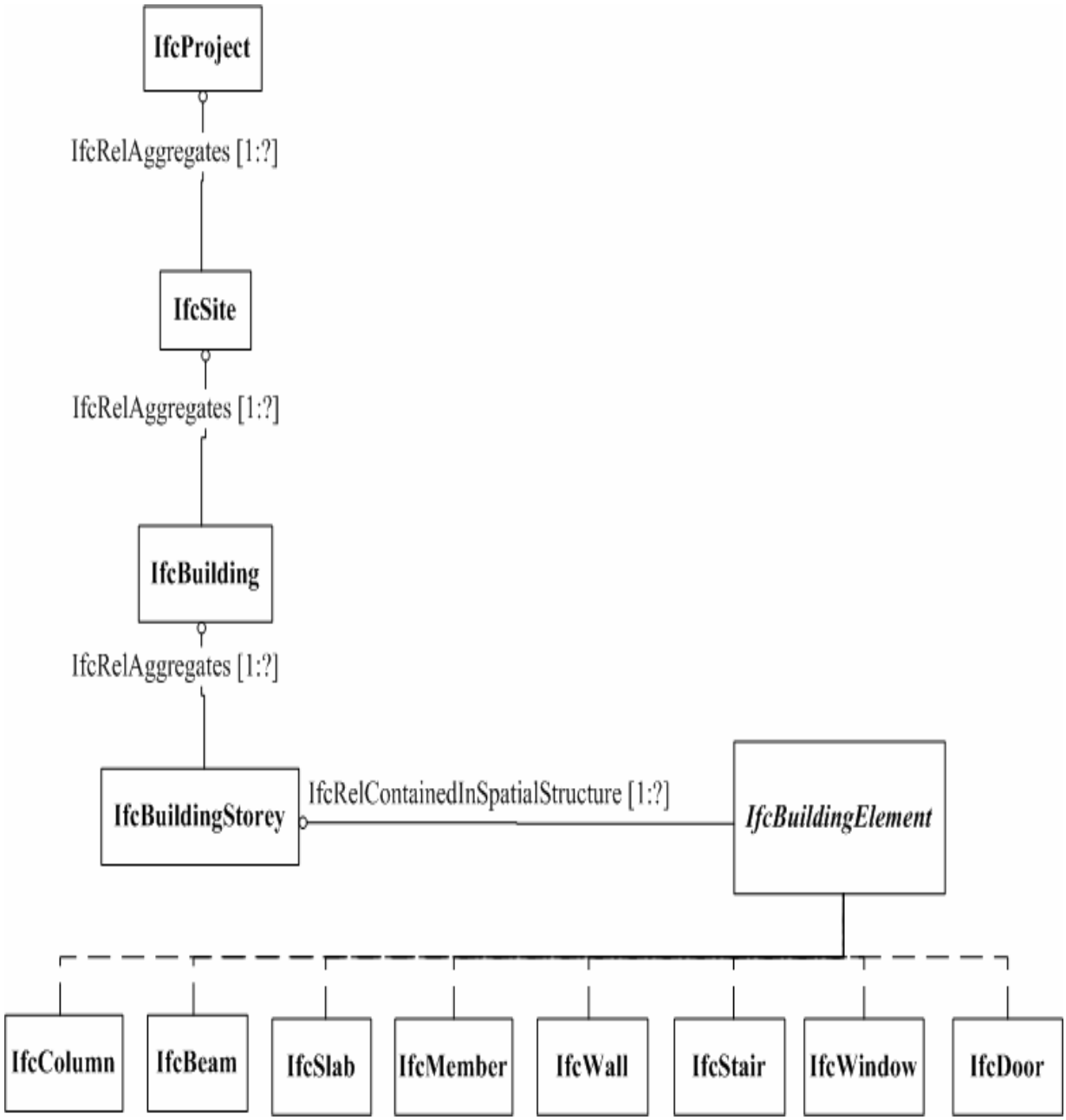

圖 3-4 IFC 建築資訊模型基本 Entity 關係

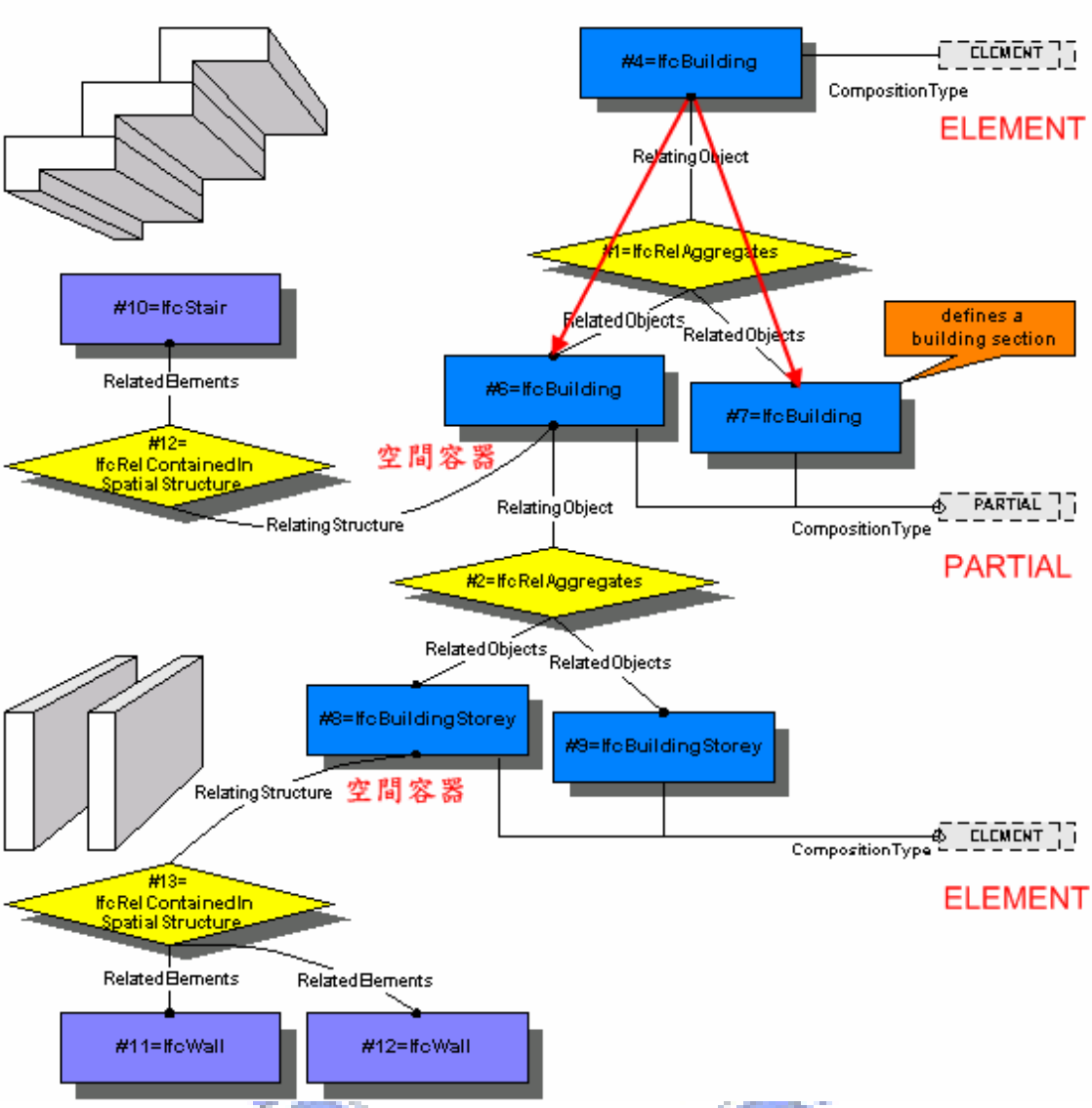

圖 3-5 IfcSpatialStructureElement 分解圖[5]

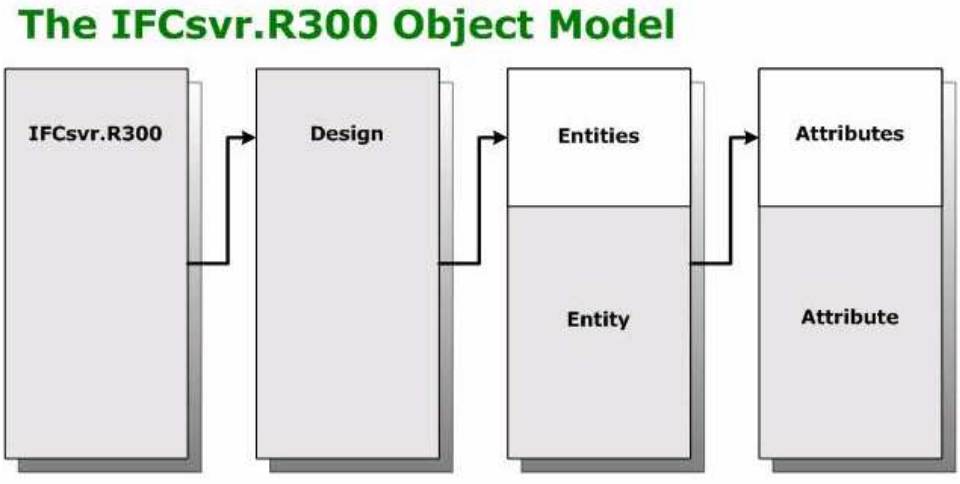

圖 3-6 IFCsvr 架構

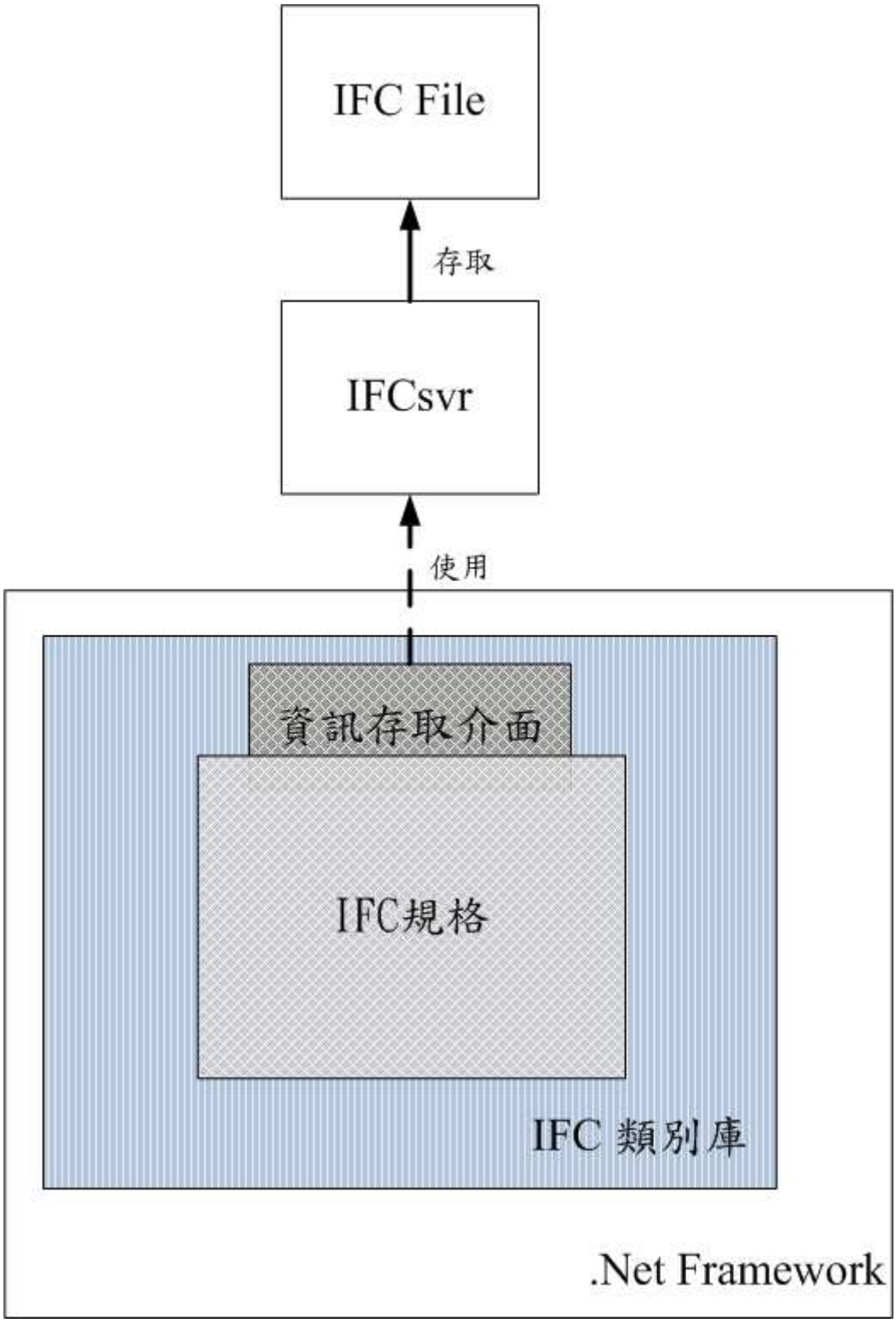

圖 4-1 本研究建立之 IFC 類別庫架構

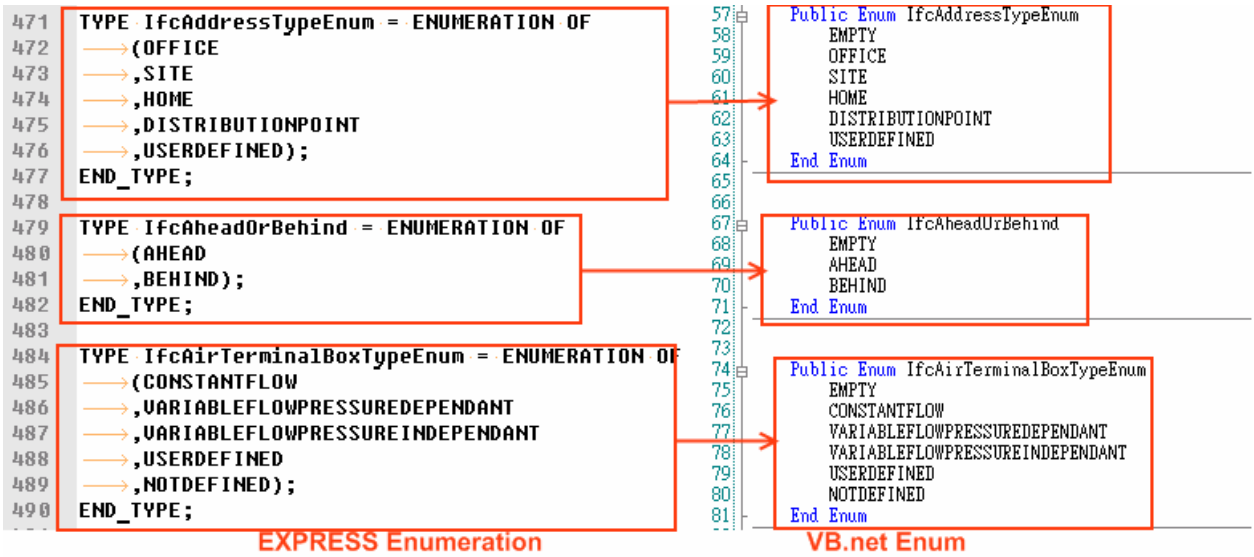

圖 4-2 Enumeration 轉換範例

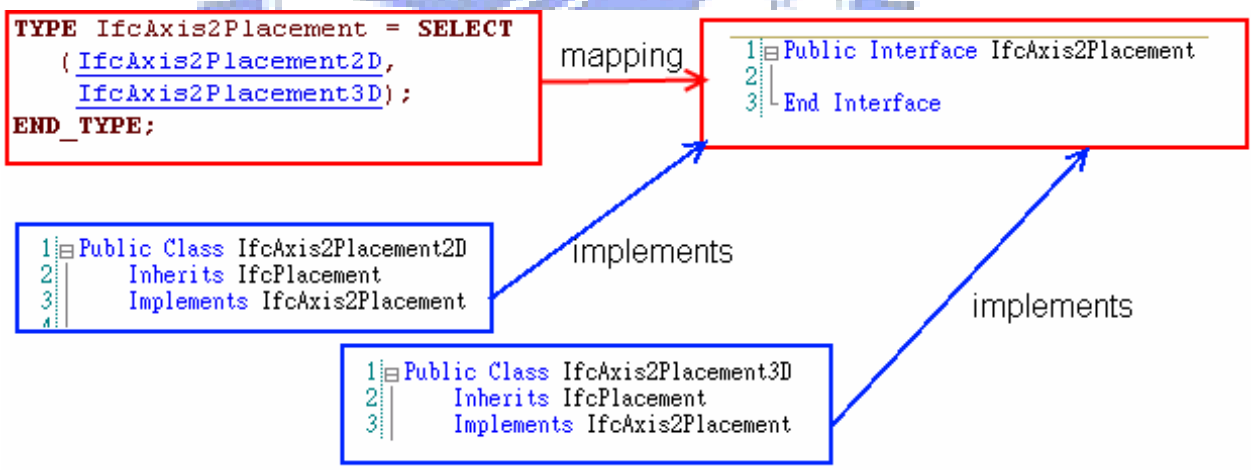

圖 4-3 Select Type 轉換範例

圖 4-4 IfcProject 類別映對

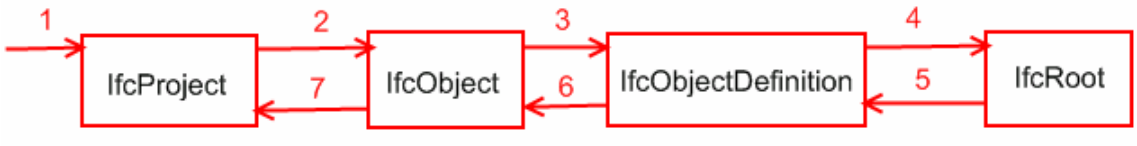

圖 4-5 資料擷取呼叫順序

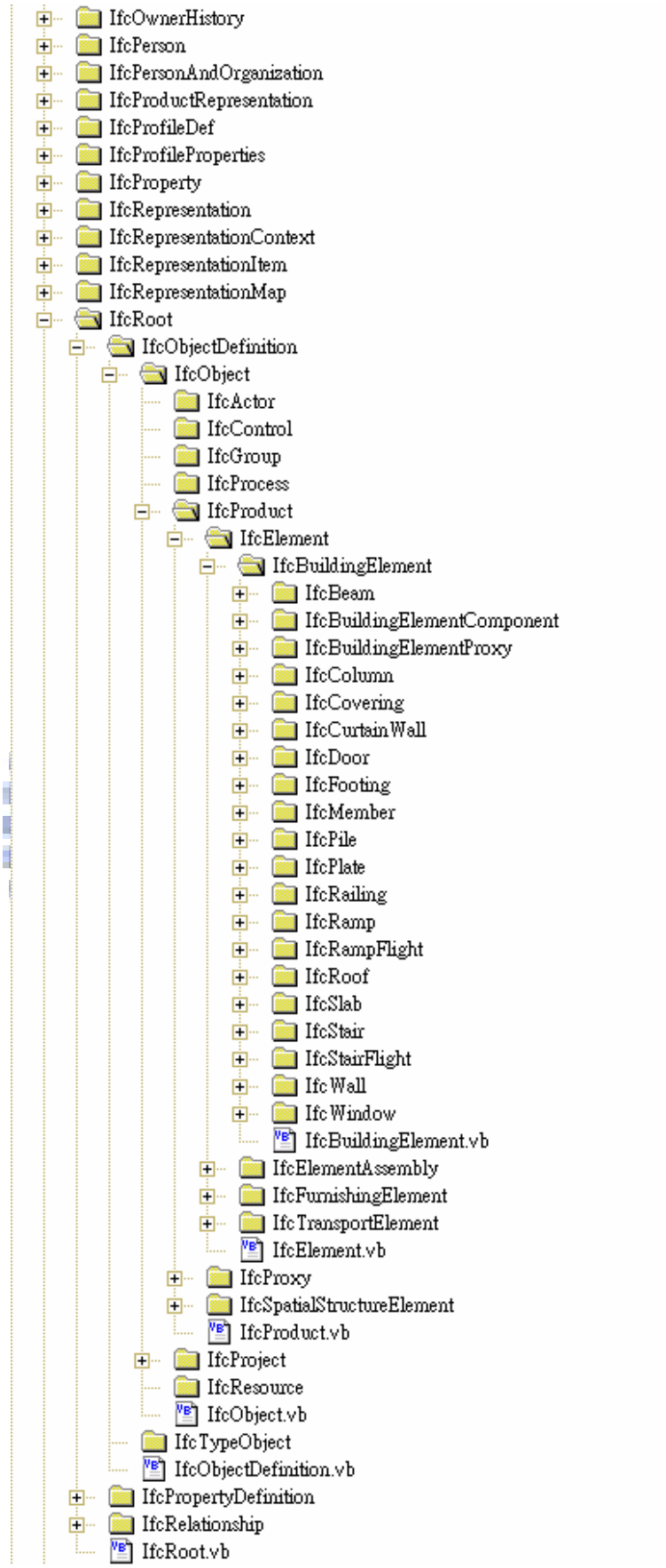
IfcOwnerHistory
IfcPerson
IfcPersonAndOrganization
IfcProductRepresentation
IfcProfileDef
IfcProfileProperties
IfcProperty
IfcRepresentation
IfcRepresentationContext
IfcRepresentationItem
IfcRepresentationMap
IfcRoot
IfcObjectDefinition
IfcObject
IfcActor
IfcControl
IfcGroup
IfcProcess
IfcProduct
IfcElement
IfcBuildingElement
IfcBeam
IfcBuildingElementComponent
IfcBuildingElementProxy
IfcColumn
IfcCovering
IfcCurtainWall
IfcDoor
IfcFooting
IfcMember
IfcPile
IfcPlate
IfcRailing
IfcRamp
IfcRampFlight
IfcRoof
IfcSlab
IfcStair
IfcStairFlight
IfcWall
IfcWindow
IfcBuildingElement.vb
IfcElementAssembly
IfcFurnishingElement
IfcTransportElement
IfcElement.vb
IfcProxy
IfcSpatialStructureElement
IfcProduct.vb
IfcProject
IfcResource
IfcObject.vb
IfcTypeObject
IfcObjectDefinition.vb
IfcPropertyDefinition
IfcRelationship
IfcRoot.vb

圖 4-6 部分已轉換類別

```
ISO-10303-21;
HEADER;
/* Generated by software containing ST-Developer
 * from STEP Tools, Inc. (www.steptools.com)
 */

FILE_DESCRIPTION(
/* description */ (''),
/* implementation_level */ '2;1');

FILE_NAME(
/* name */ 'sample',
/* time_stamp */ '2008-04-09T10:49:30+08:00',
/* author */ (''),
/* organization */ (''),
/* preprocessor_version */ 'ST-DEVELOPER v10',
/* originating_system */ '',
/* authorisation */ '');

FILE_SCHEMA (('IFC2X3'));
ENDSEC;

DATA;
#10=IFCAPPLICATION(#12,'v4','IFCool','IFCool');
#11=IFCACTORROLE(.ENGINEER.,$,$);
#12=IFCORGANIZATION($,'Lin Lab',$,(#11),());
#13=IFCPERSON('001','Shen','Pingting',(),(),(),(),());
#14=IFCPERSONANDORGANIZATION(#13,#12,());
#15=IFCOWNERHISTORY(#14,#10,.READWRITE.,.NOCHANGE.,0,$,$,0);
ENDSEC;
END-ISO-10303-21;
```

圖 5-1 範例輸出檔

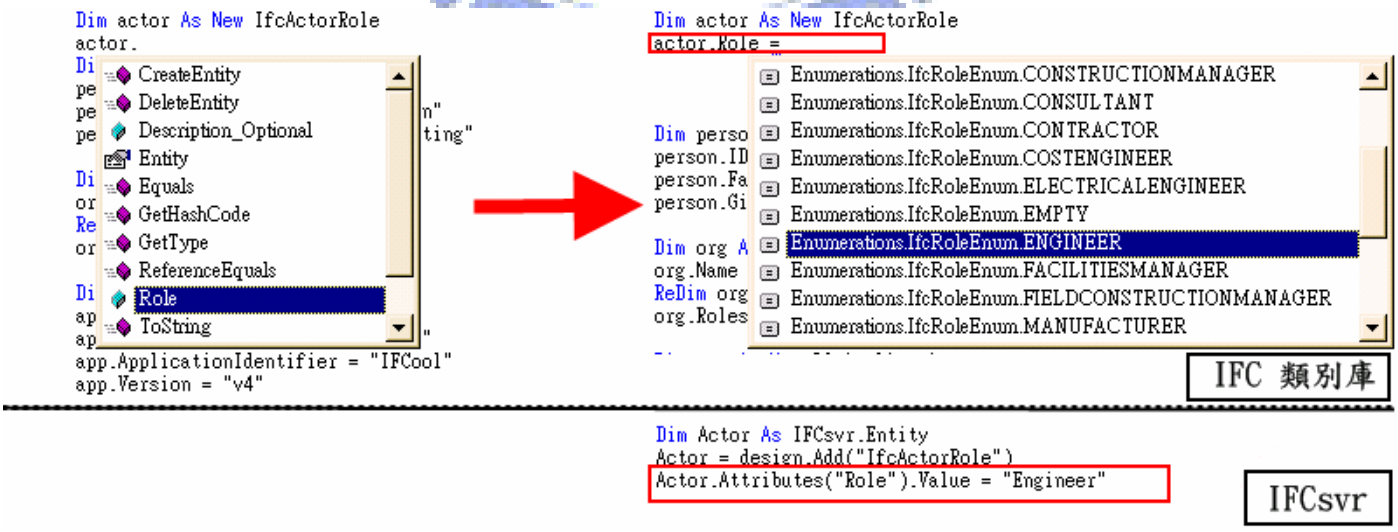

圖 5-2 類別庫與 IFCsvr 屬性存取比較

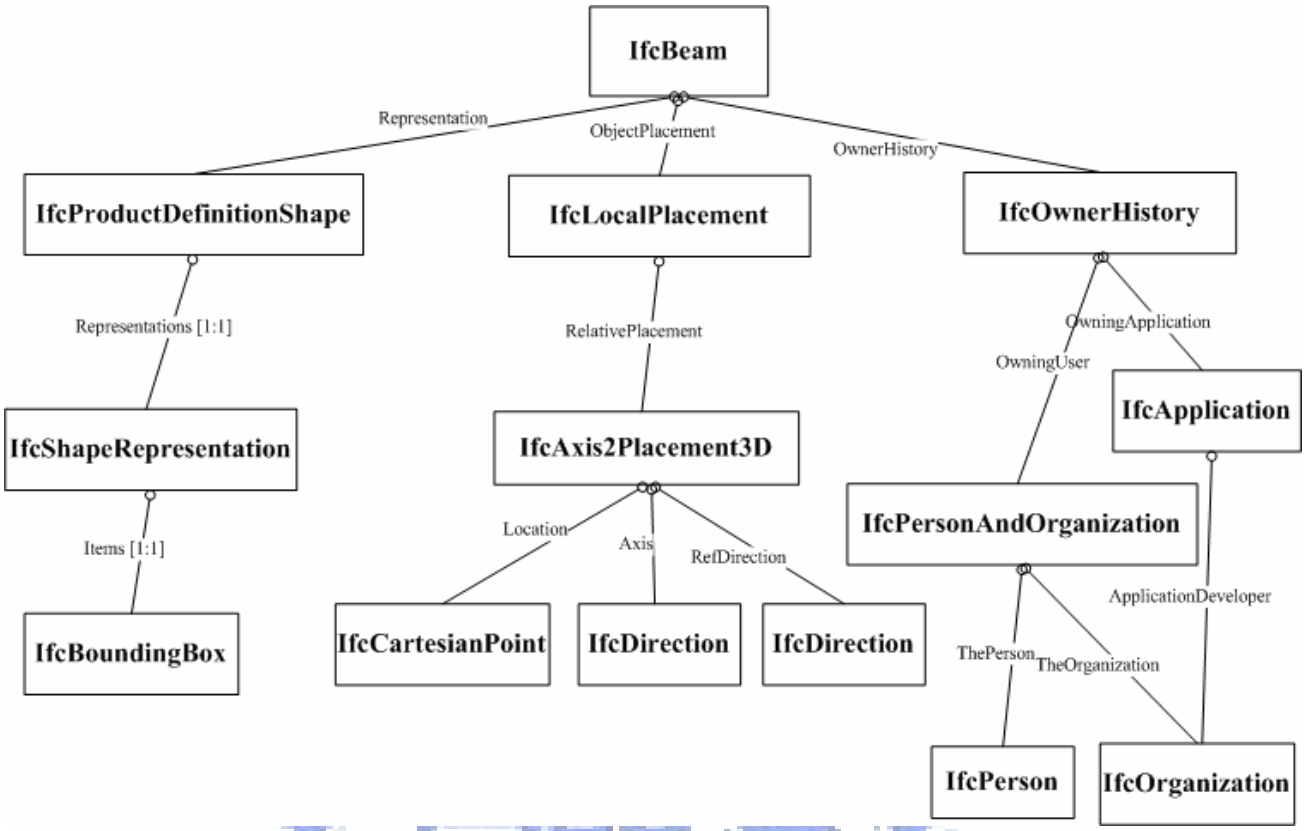

圖 5-3 自定義 IfcBeam 模型視圖

sample.ifc - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
/* description */ (''),
/* implementation_level */ '2;1');

FILE_NAME(
/* name */ 'sample',
/* time_stamp */ '2008-05-23T15:41:59+08:00',
/* author */ (''),
/* organization */ (''),
/* preprocessor_version */ 'ST-DEVELOPER v10',
/* originating_system */ '',
/* authorisation */ '');

FILE_SCHEMA (('IFC2X3'));
ENDSEC;

DATA;
#10=IFCBOUNDINGBOX(#16,15.,20.,300.);
#11=IFCSHAPEREPRESENTATION($,$,$,(#10));
#12=IFCPRODUCTDEFINITIONSHAPE($,$,(#11));
#13=IFCDIRECTION((0.,0.,1.));
#14=IFCDIRECTION((1.,0.,0.));
#15=IFCCARTESIANPOINT((0.,0.,0.));
#16=IFCCARTESIANPOINT((0.,0.,0.));
#17=IFCAXIS2PLACEMENT3D(#15,#13,#14);
#18=IFCLOCALPLACEMENT($,#17);
#19=IFCAPPLICATION(#21,'v4','IFCool','IFCool');
#20=IFCACTORROLE(.ENGINEER.,$,$);
#21=IFCORGANIZATION($,'Lin Lab',$,(#20),());
#22=IFCPERSON('54967','Shen','Pingting',(),(),(),(),());
#23=IFCPERSONANDORGANIZATION(#22,#21,());
#24=IFCOWNERHISTORY(#23,#19,.READWRITE.,.NOCHANGE.,0,$,$,0);
#25=IFCBEAM('001',#24,$,$,$,#18,#12,$);
ENDSEC;
END-ISO-10303-21;
```

圖 5-4 範例輸出檔

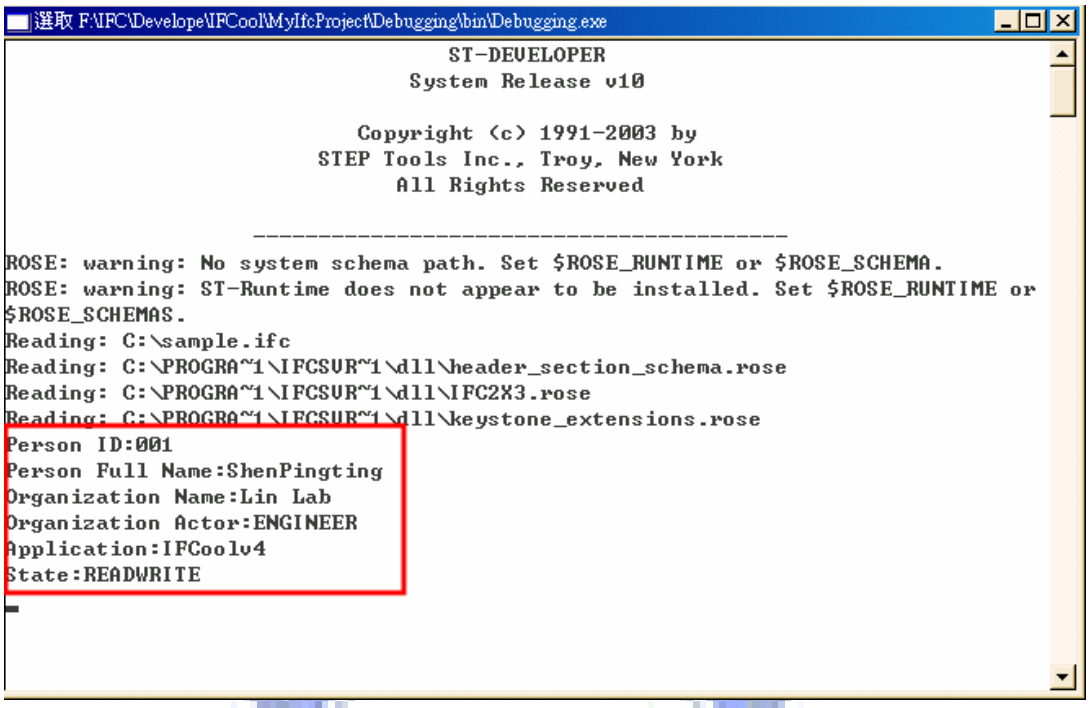

圖 5-5 輸出範例

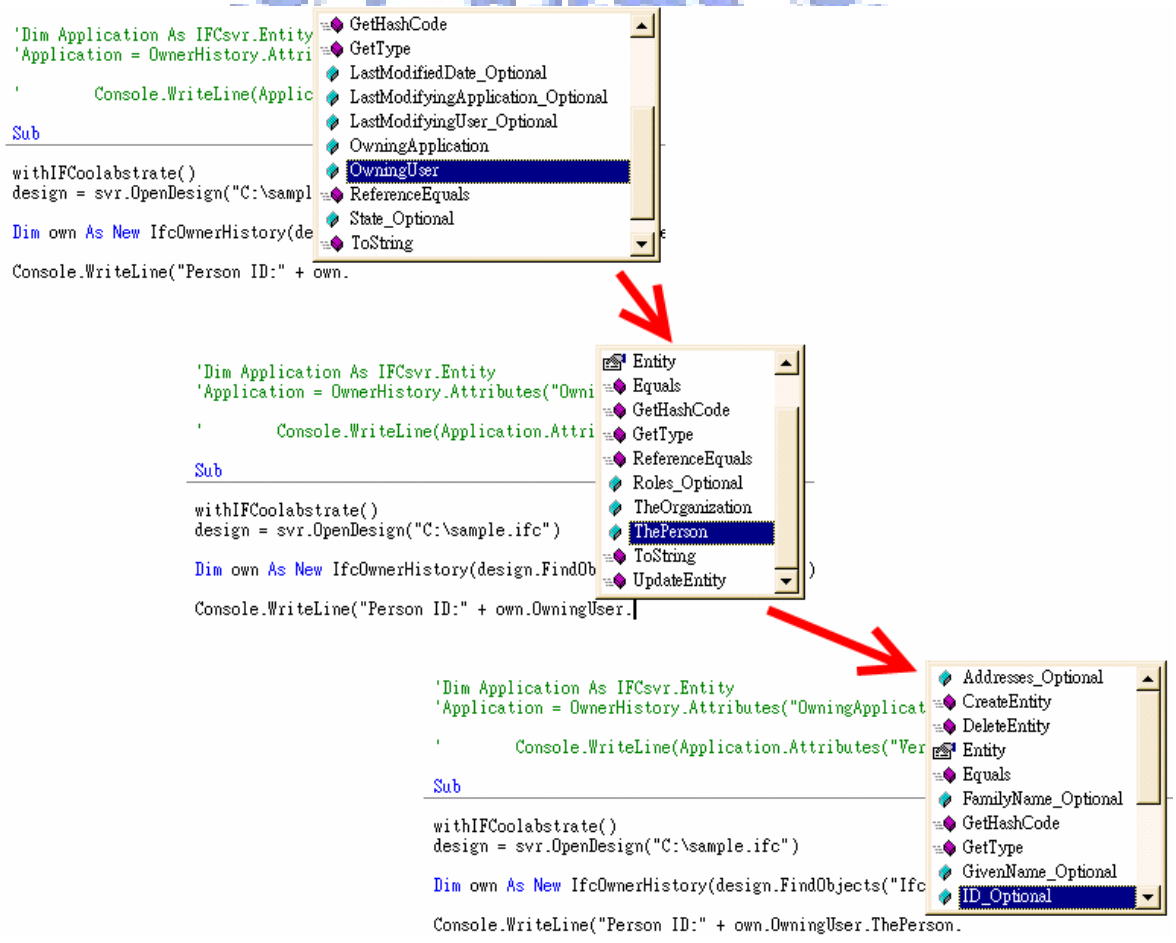

圖 5-6 Entity 物件屬性存取

```
Dim extruded_area As IfcExtrudedAreaSolid
extruded_area.SweptArea = New IfcCircleProfileDef         ' OK 圓形斷面
extruded_area.SweptArea = New IfcTShapeProfileDef         ' OK T型斷面
extruded_area.SweptArea = New IfcIShapeProfileDef         ' OK I型斷面
extruded_area.SweptArea = New MyRectangleProfile(10, 15) ' OK 自定義矩形斷面類別
extruded_area.SweptArea = New IfcBoundingBox              ' NG 不具IfcProfileDef介面
                          型別 IFCool4.IfcBoundingBox 的值無法轉換成 IFCool4.IfcProfileDef。
```

圖 5-7 型別轉換錯誤

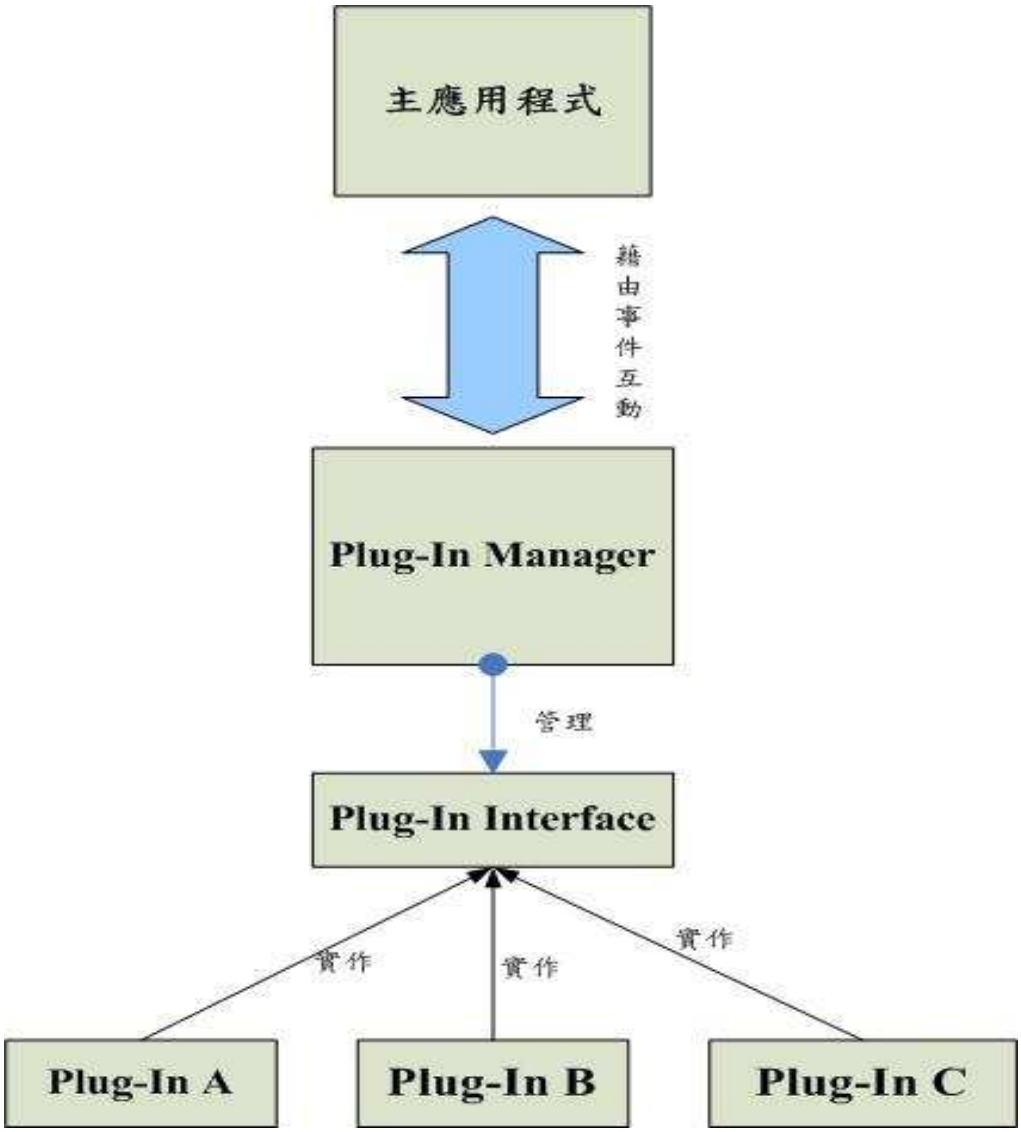

圖 5-8 Plug-In 系統架構

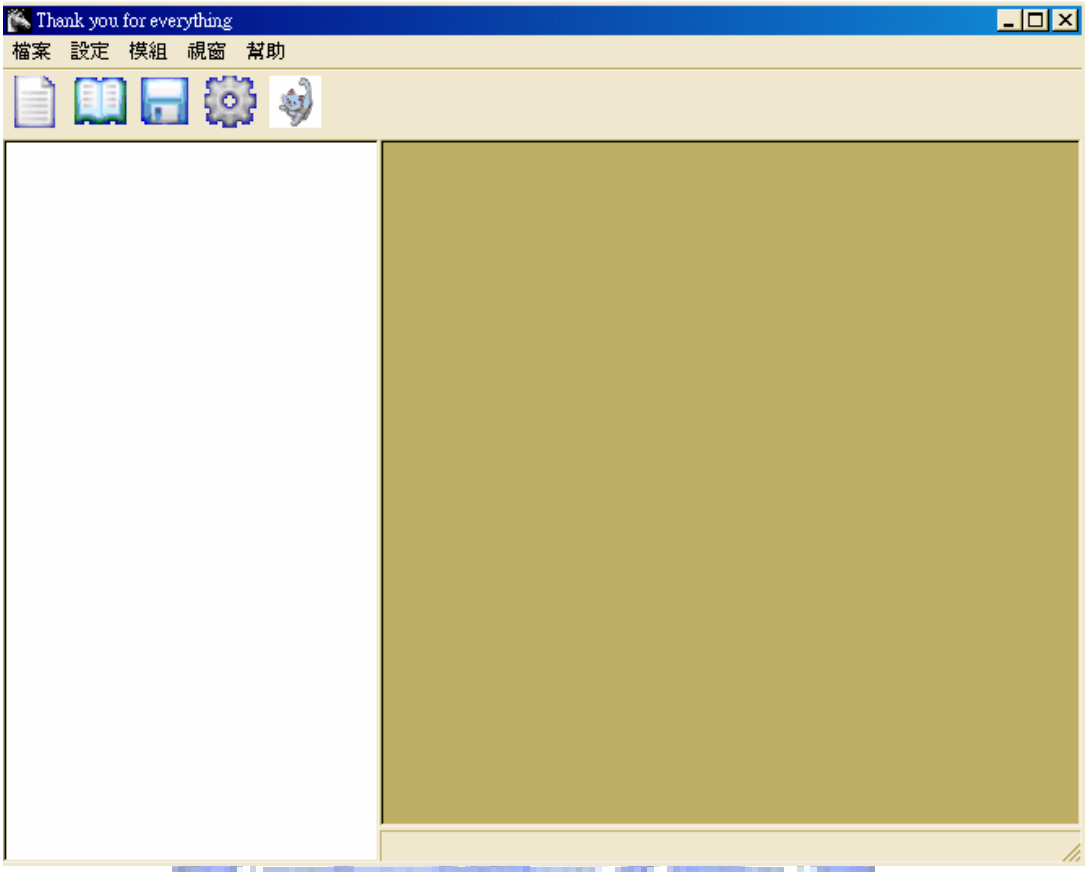

圖 5-9 主應用程式使用者介面

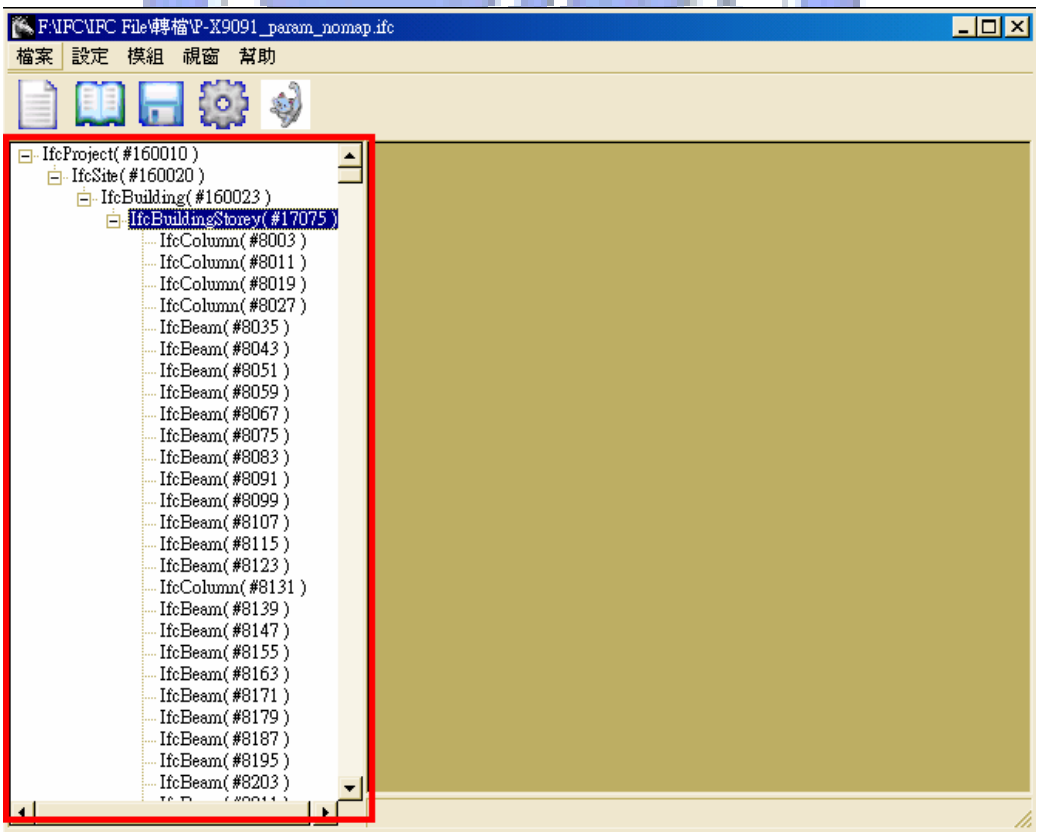

圖 5-10 主應用程式樹狀結構視窗

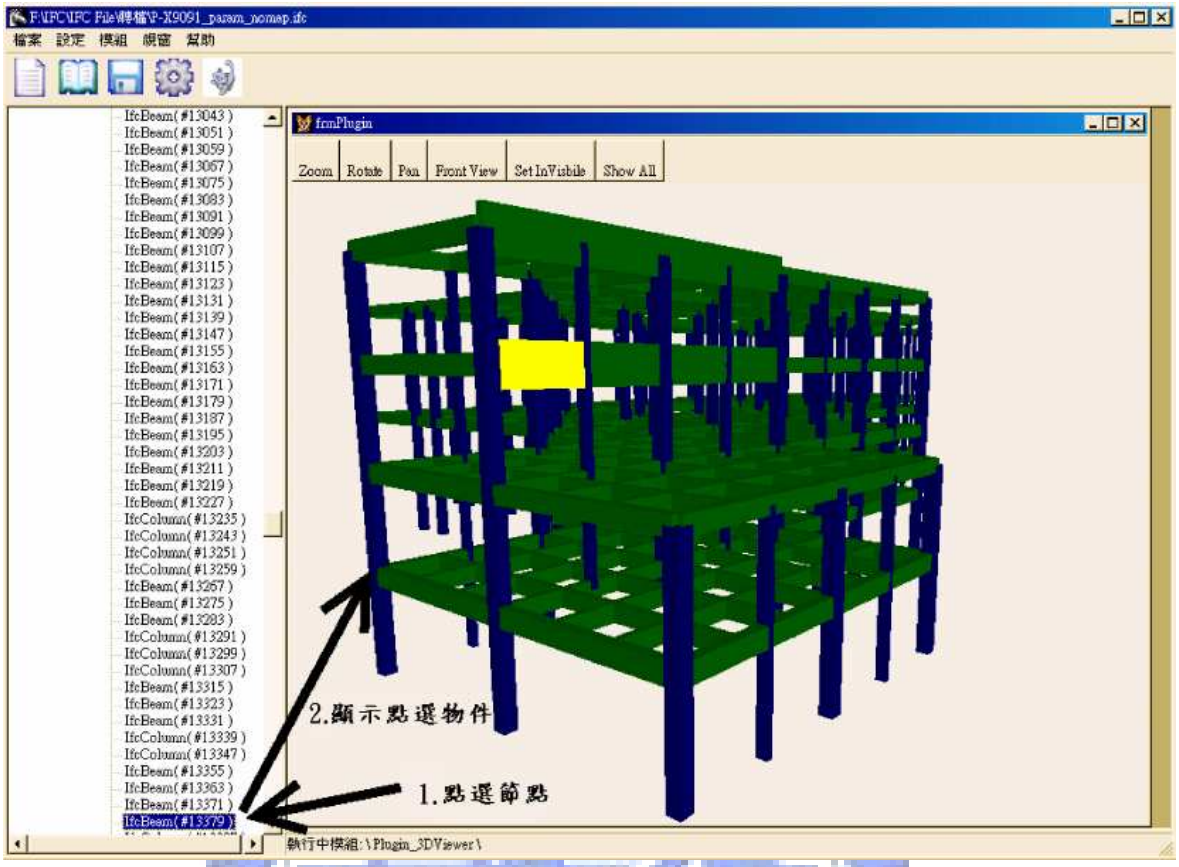

圖 5-11 3D 建築模型瀏覽 Plug-In 元件-1

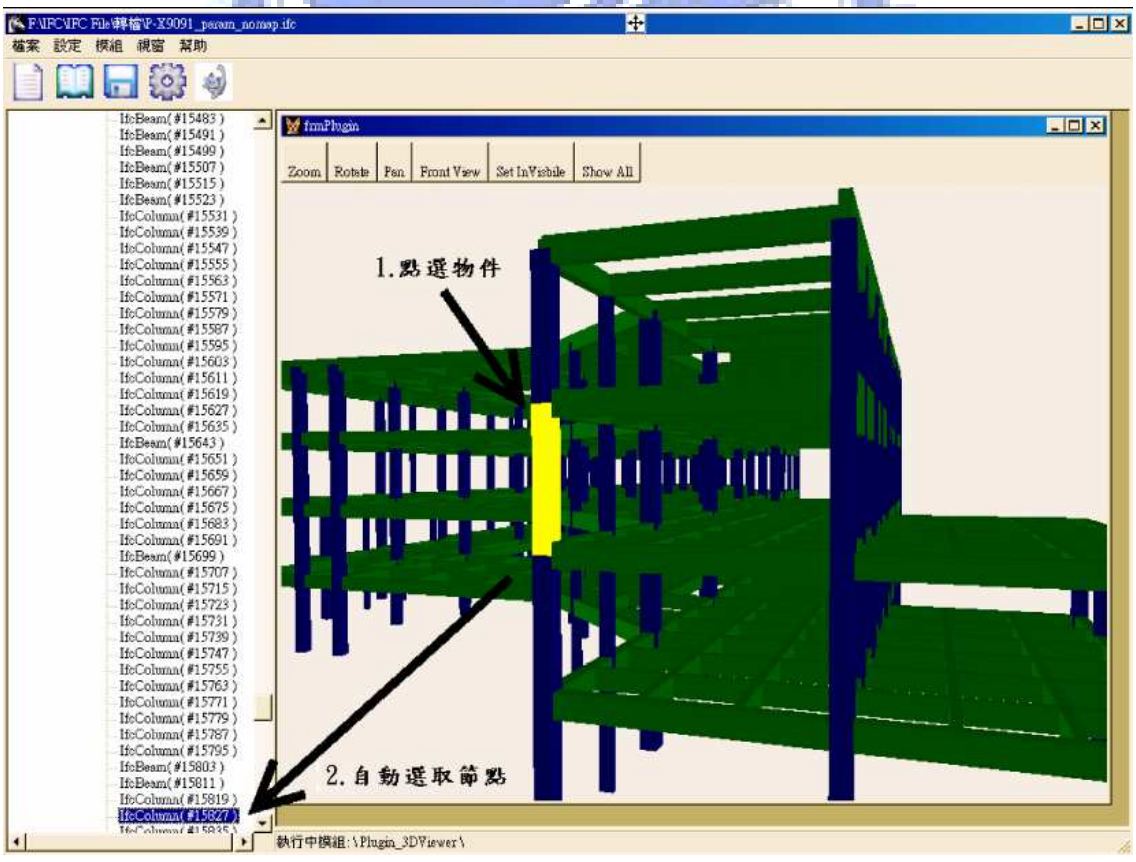

圖 5-12 3D 建築模型瀏覽 Plug-In 元件-2

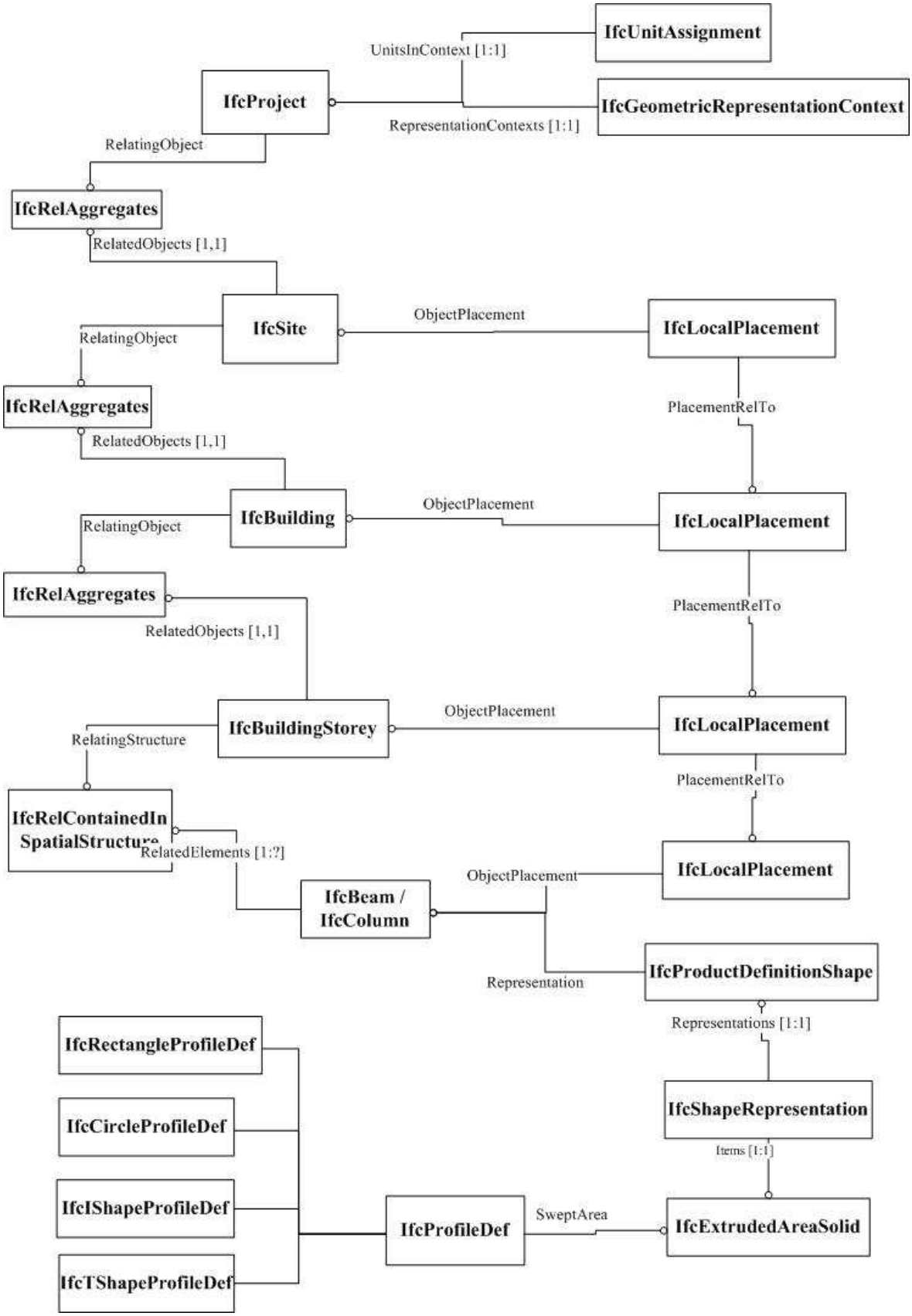

圖 5-13 IFC 建築物模型視圖

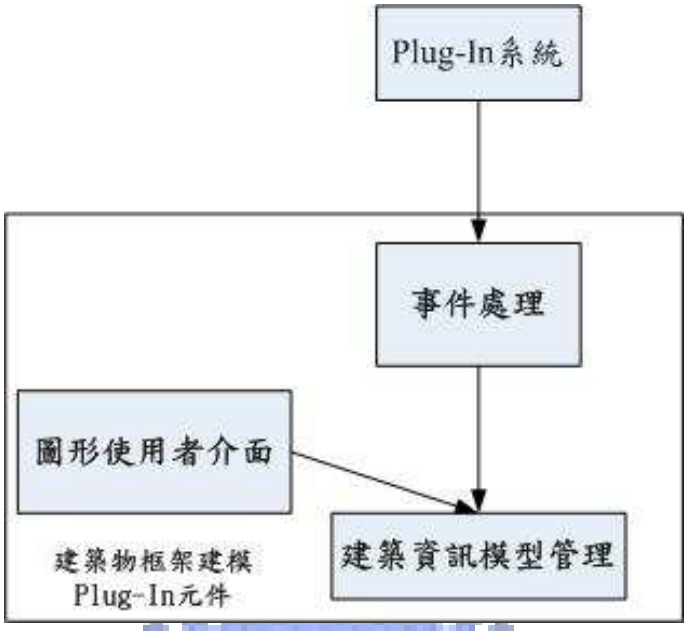

圖 5-14 建築物框架建模 Plug-In 元件架構

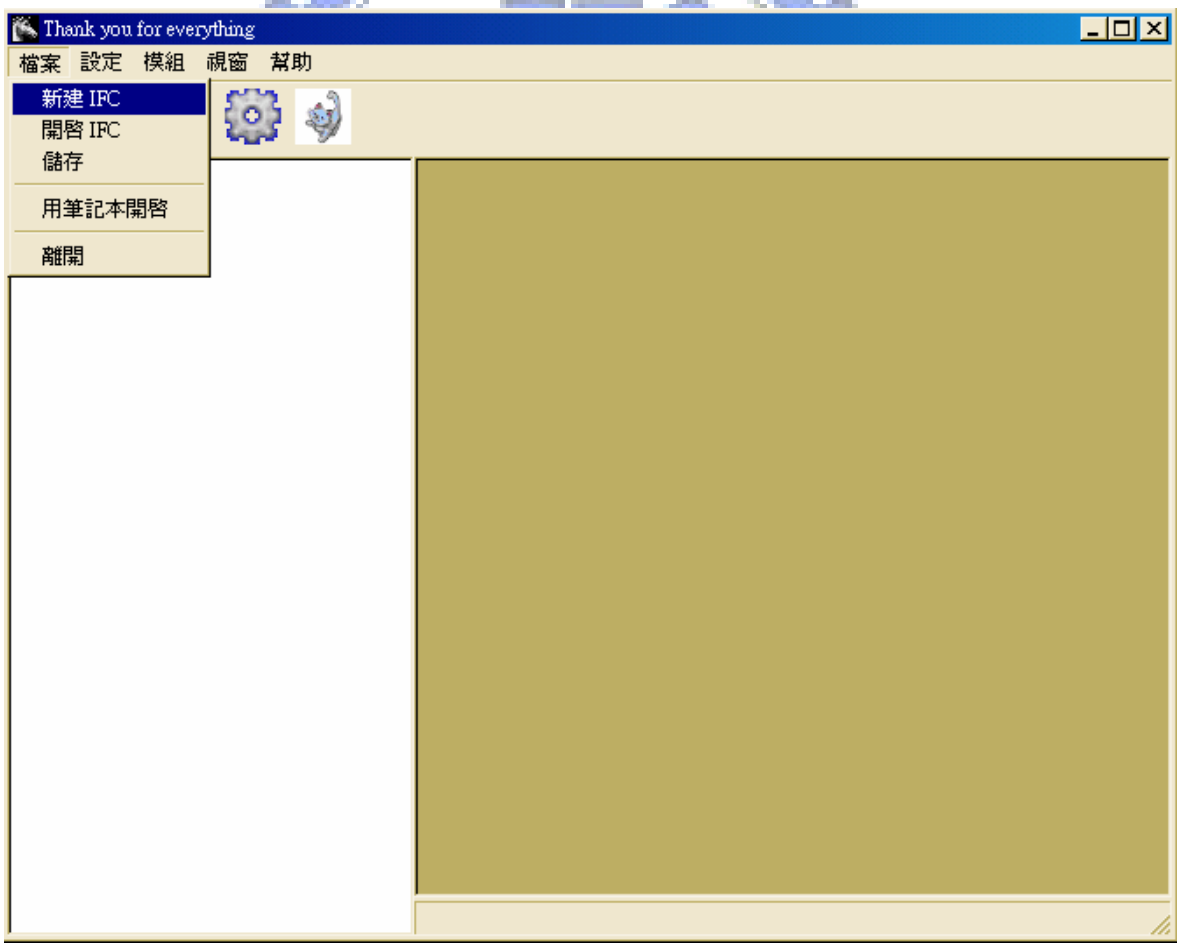

圖 5-15 新建 IFC 檔案

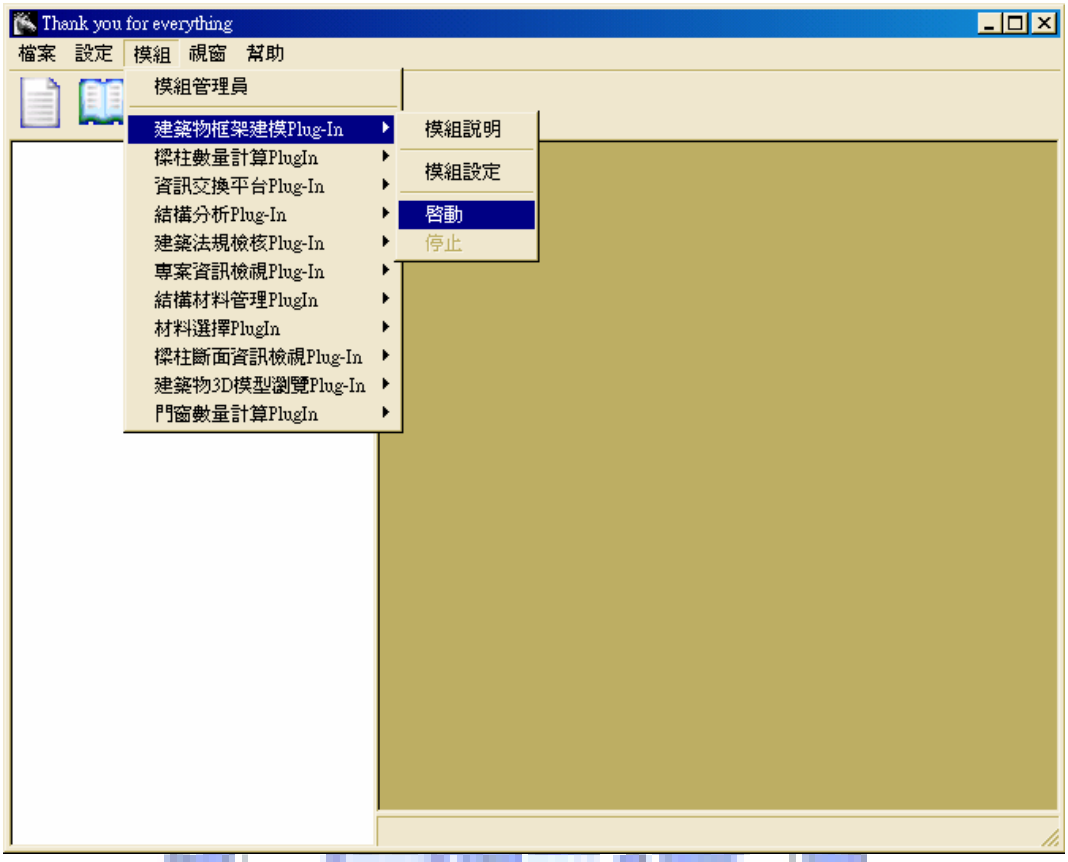

圖 5-16 啟動 Plug-In

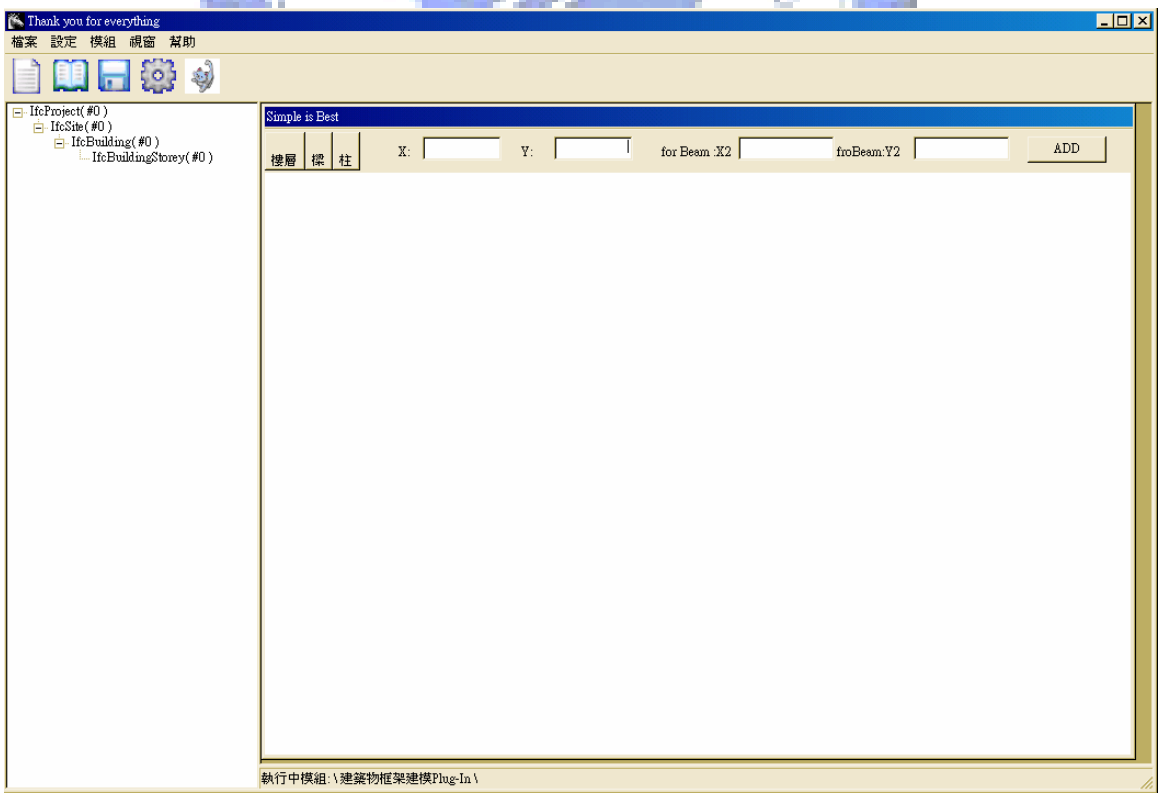

圖 5-17 初始畫布視窗

圖 5-18 新增樓層-1

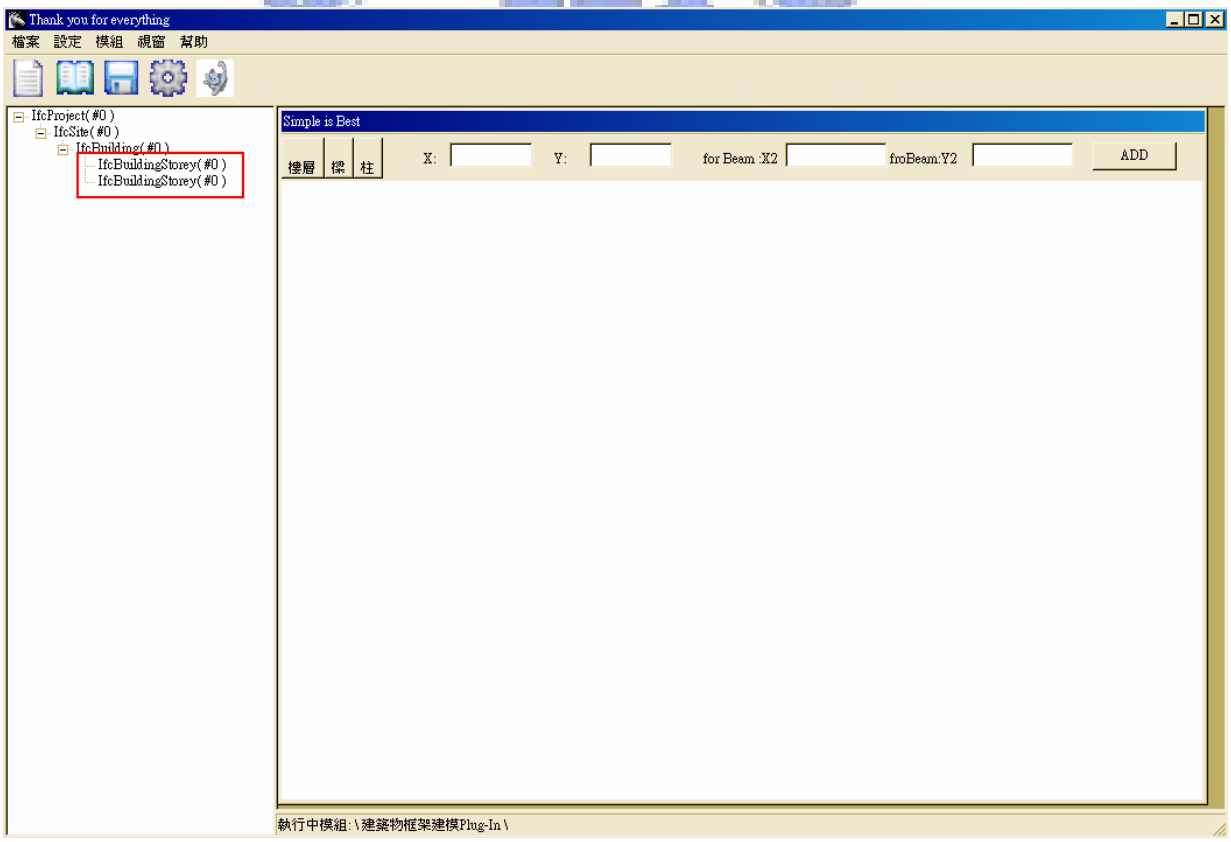

圖 5-19 新增樓層-2

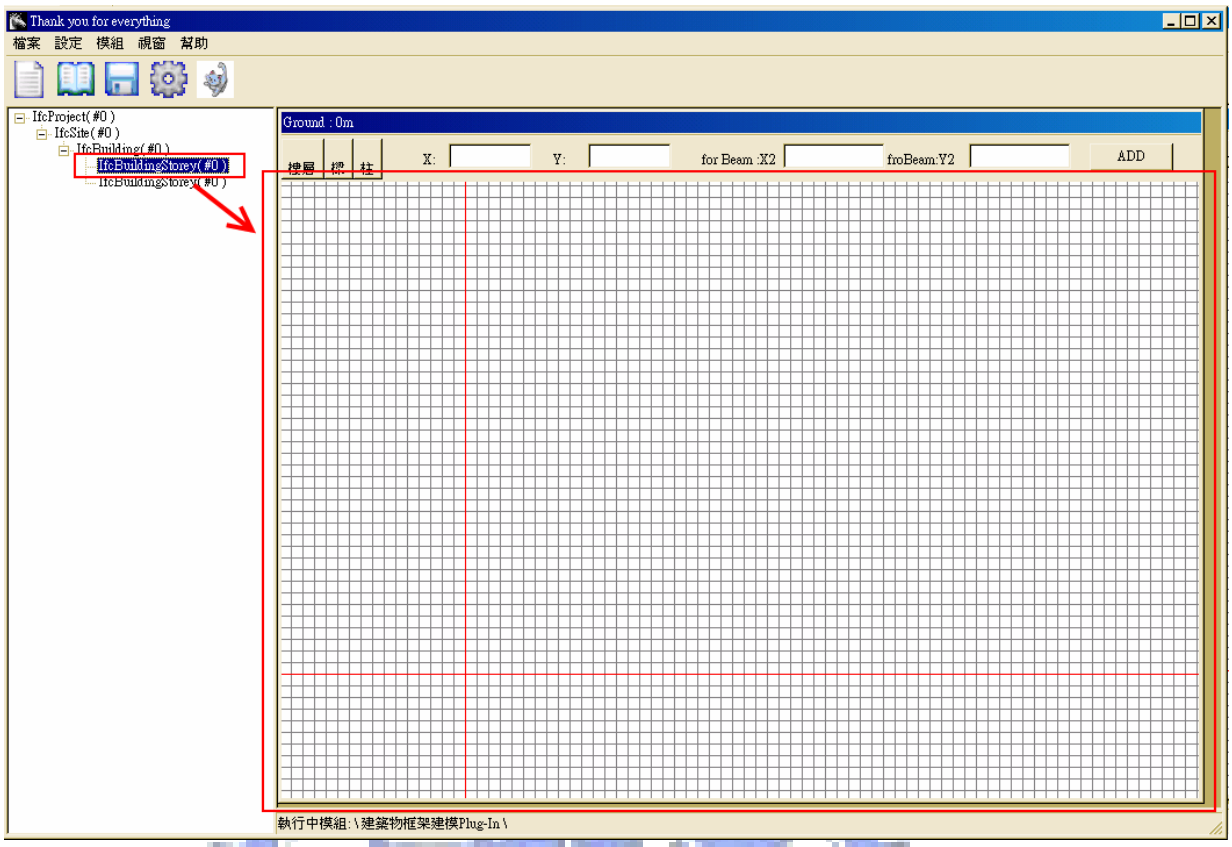

圖 5-20 選擇樓層

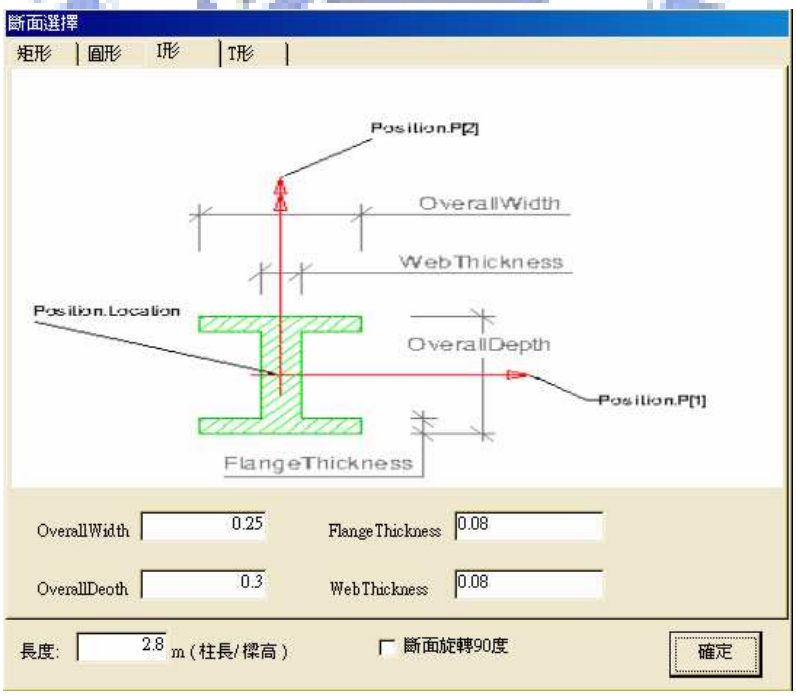

圖 5-21 斷面選擇視窗

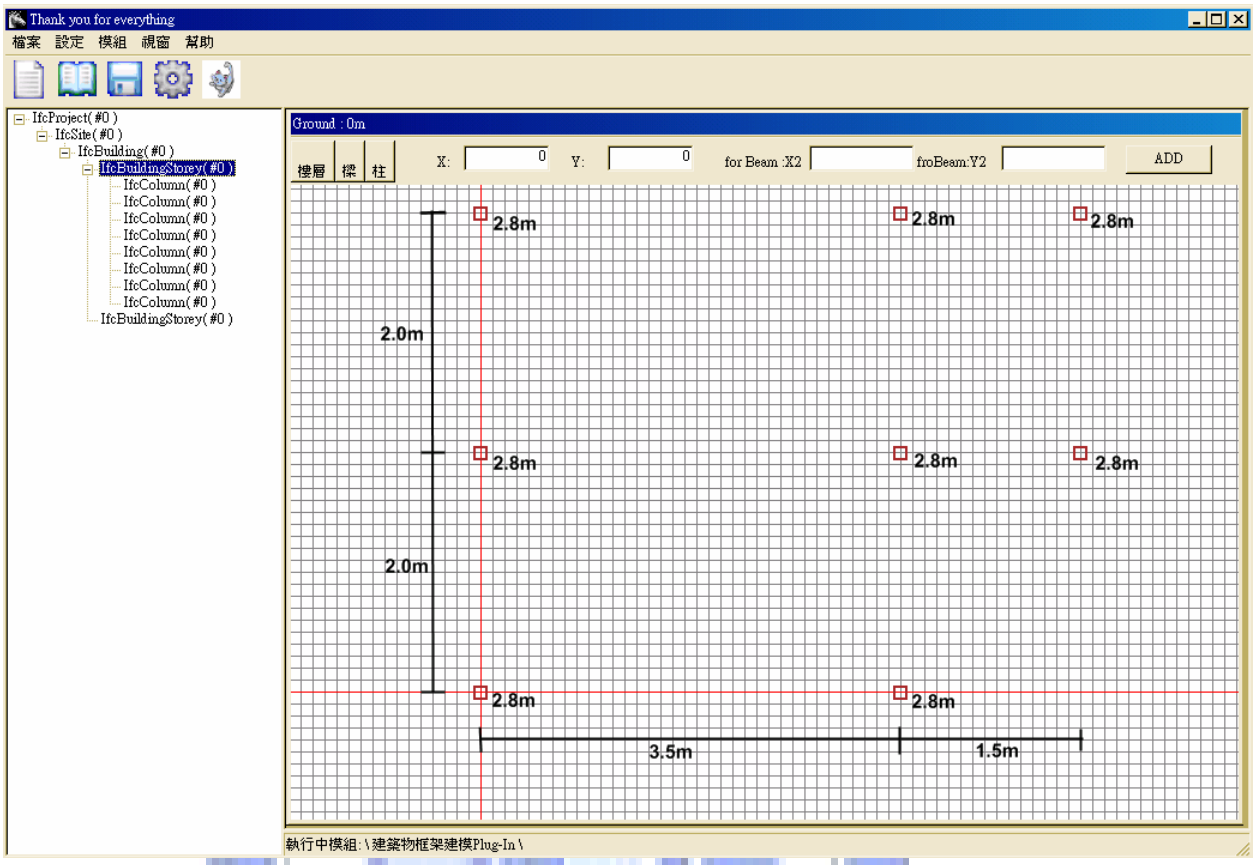

圖 5-22 新增柱元件

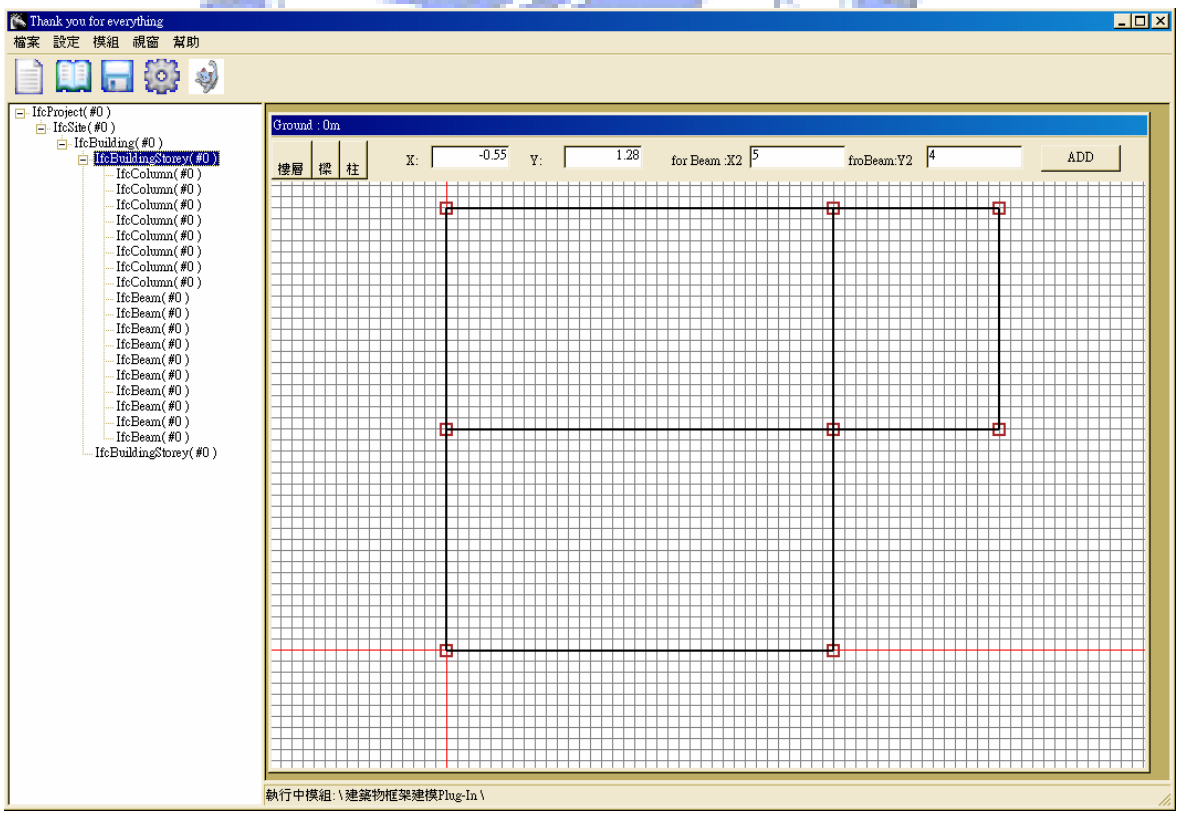

圖 5-23 新增樑元件

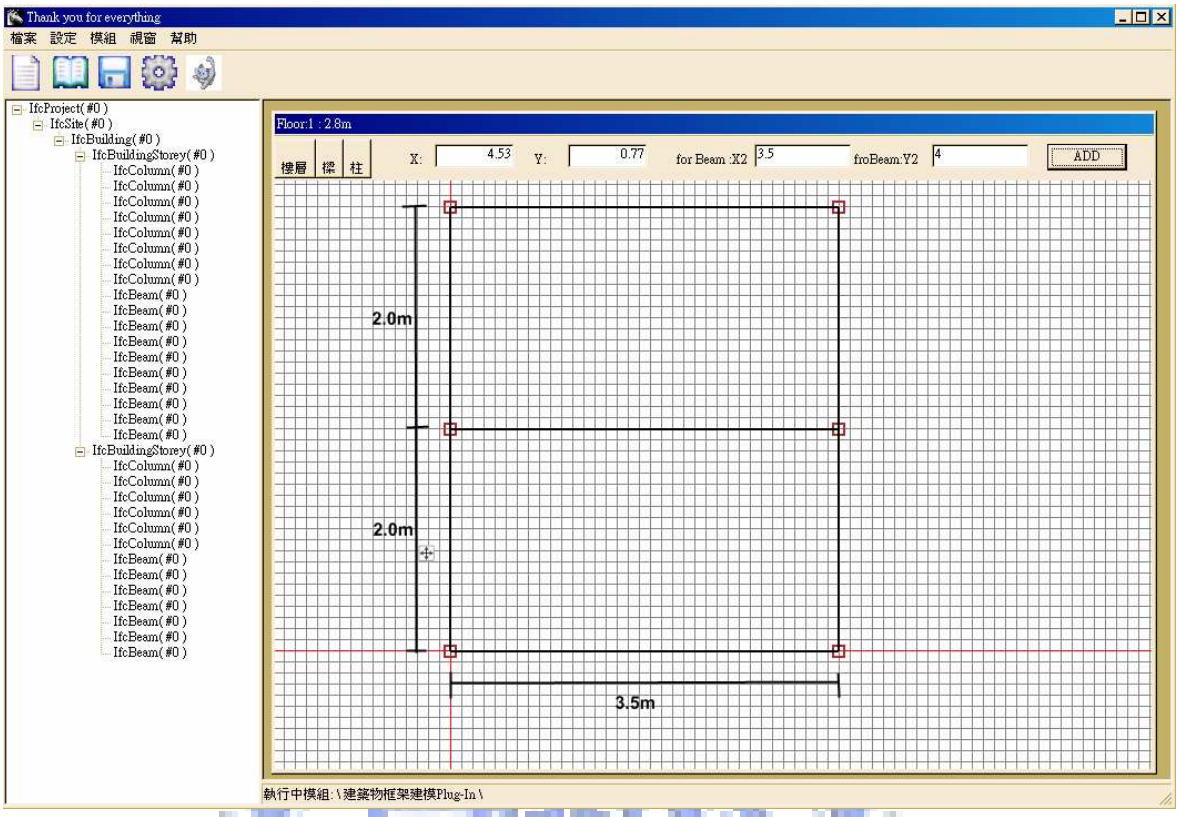

圖 5-24 新增樑柱元件

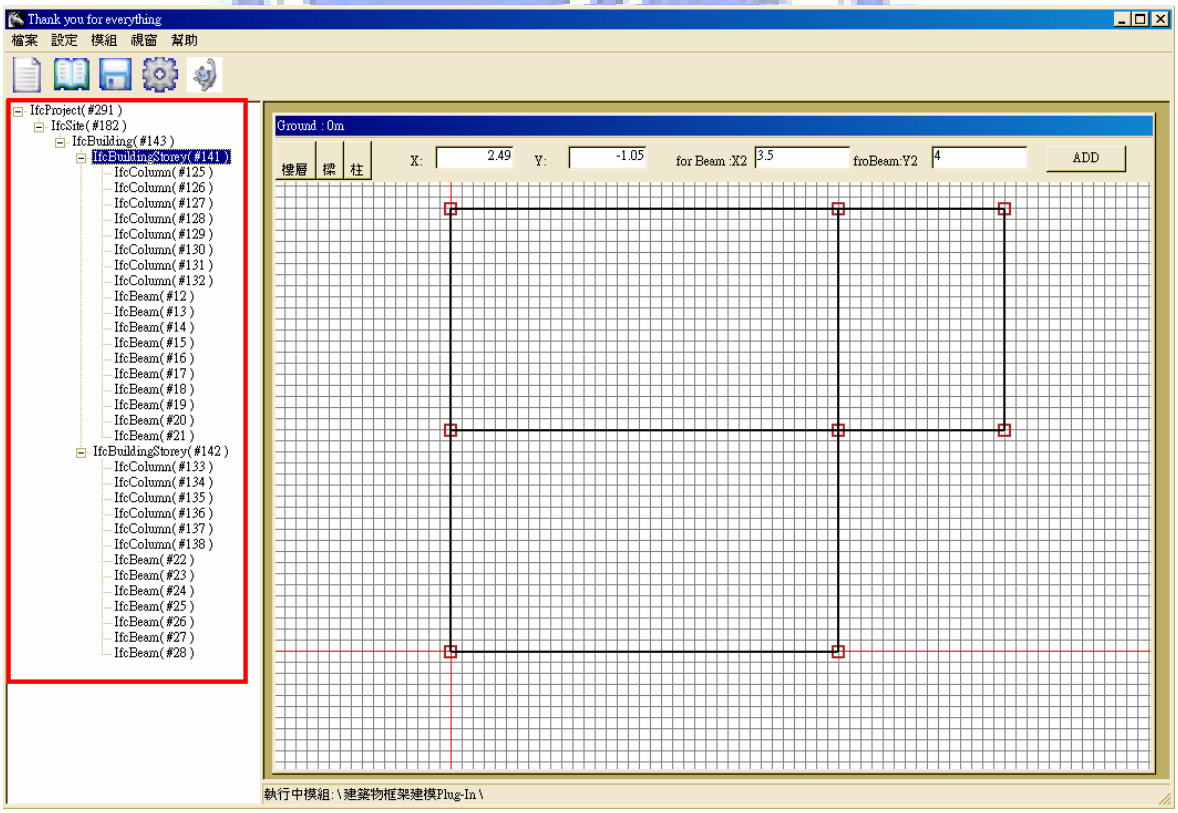

圖 5-25 樹狀結構更新

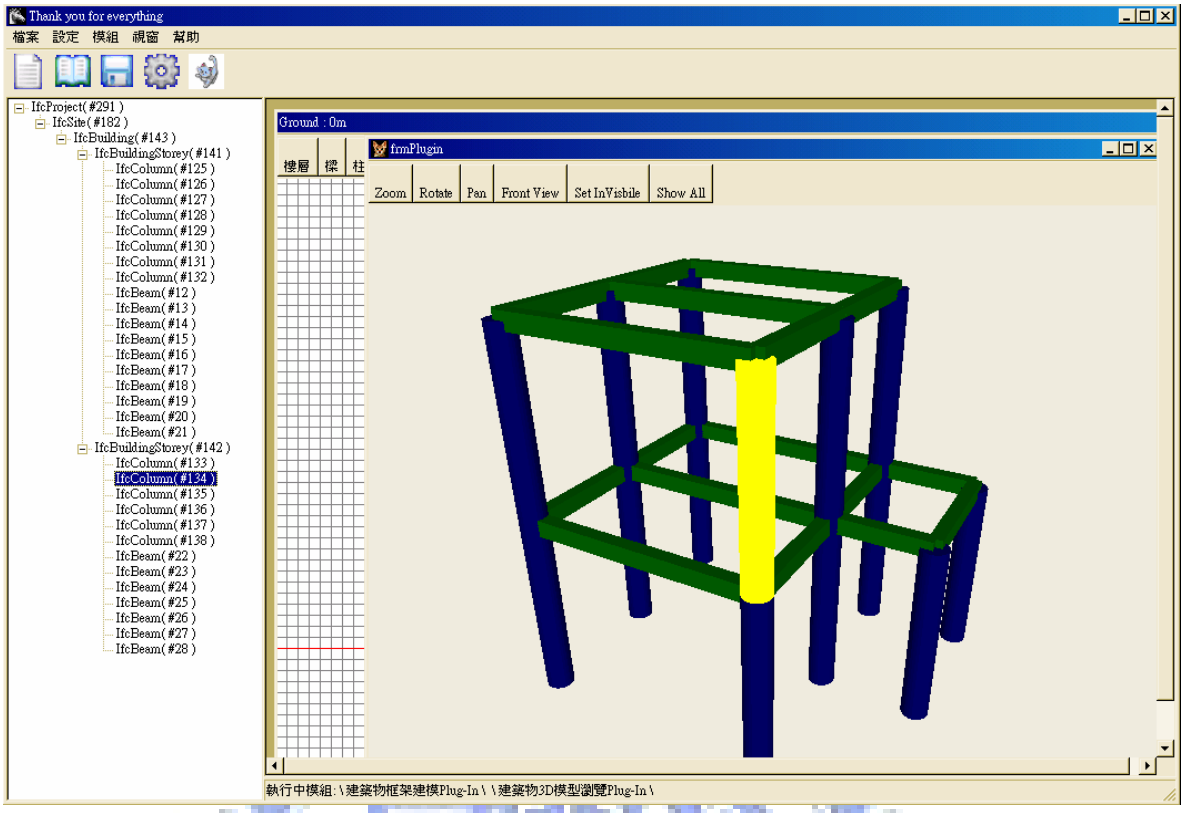

圖 5-26 啟動 3D 模型瀏覽 Plug-In 觀看模型

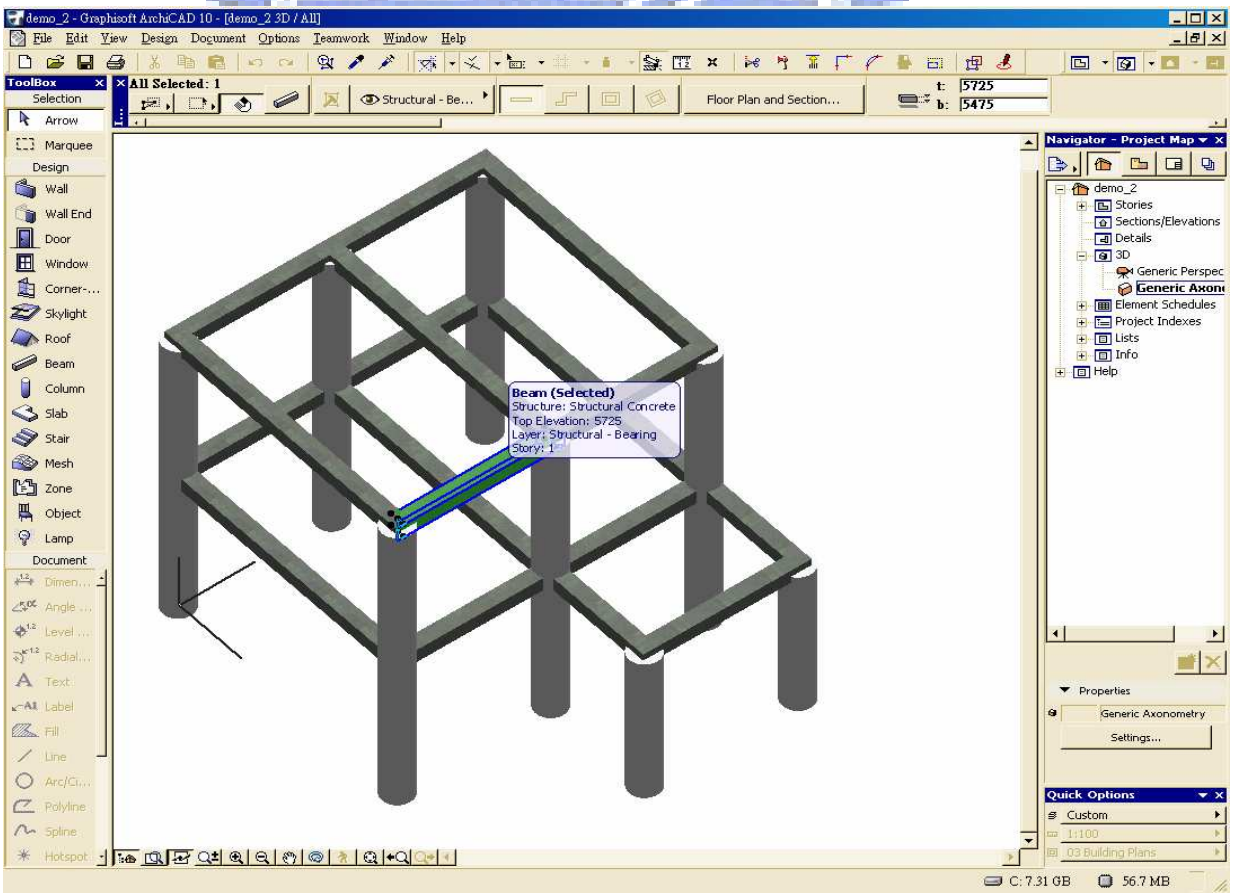

圖 5-27 將模型載入 ArchiCad

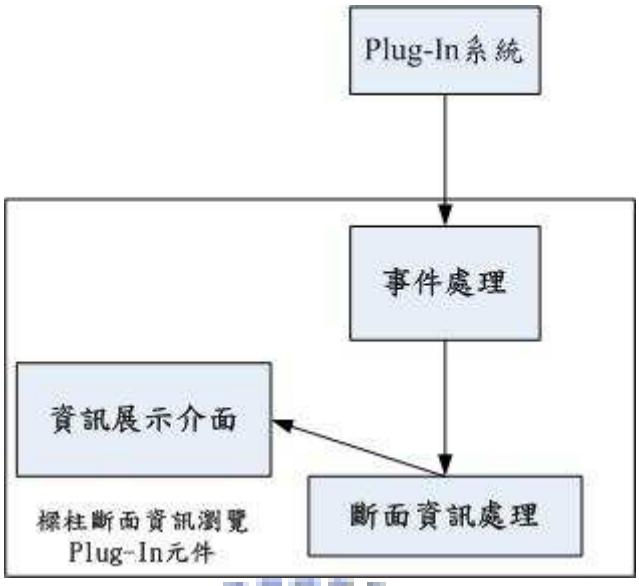

圖 5-28 樑柱斷面資訊瀏覽 Plug-In 元件架構

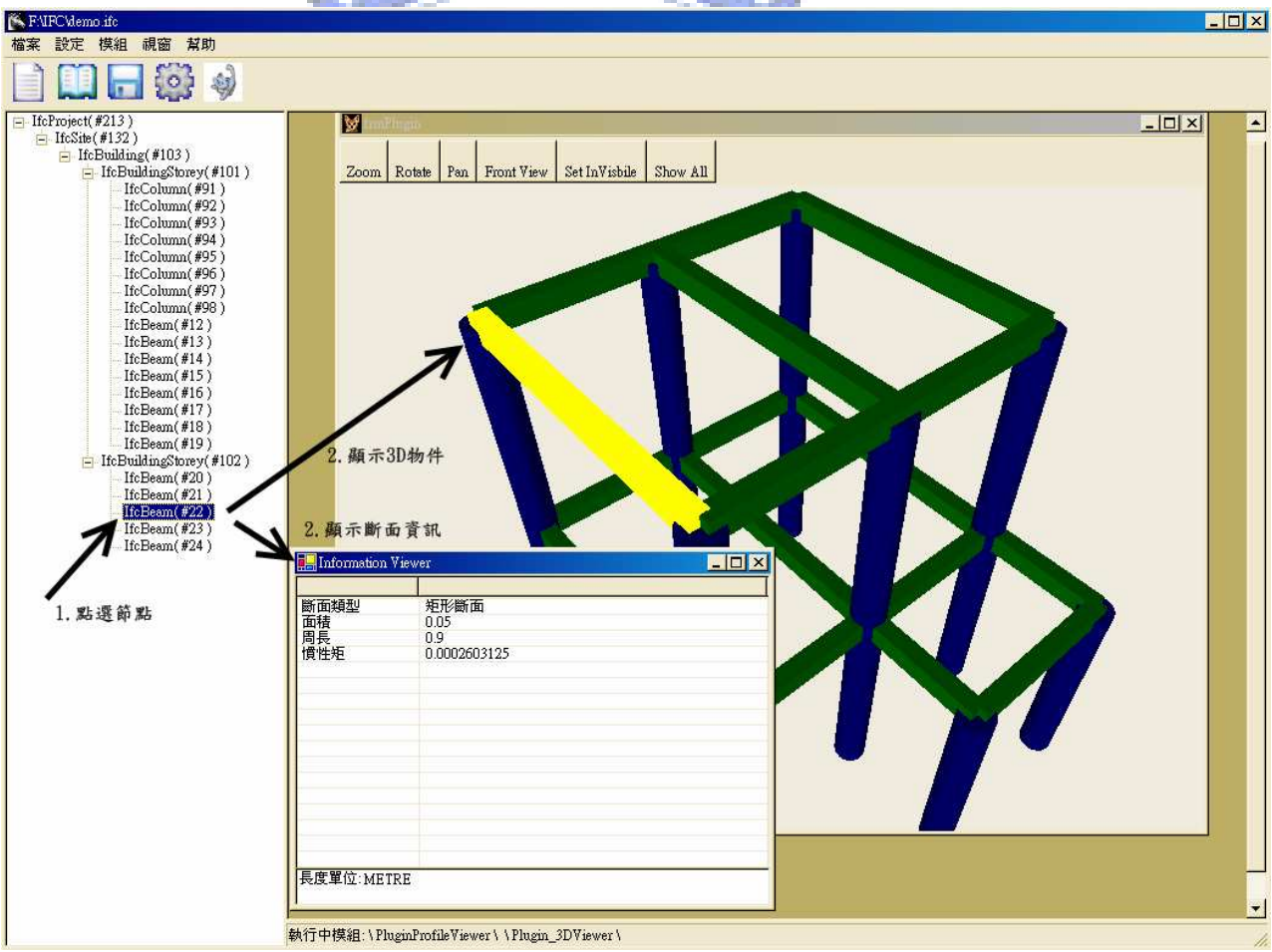

圖 5-29 樑柱斷面資訊瀏覽 Plug-In 元件-1

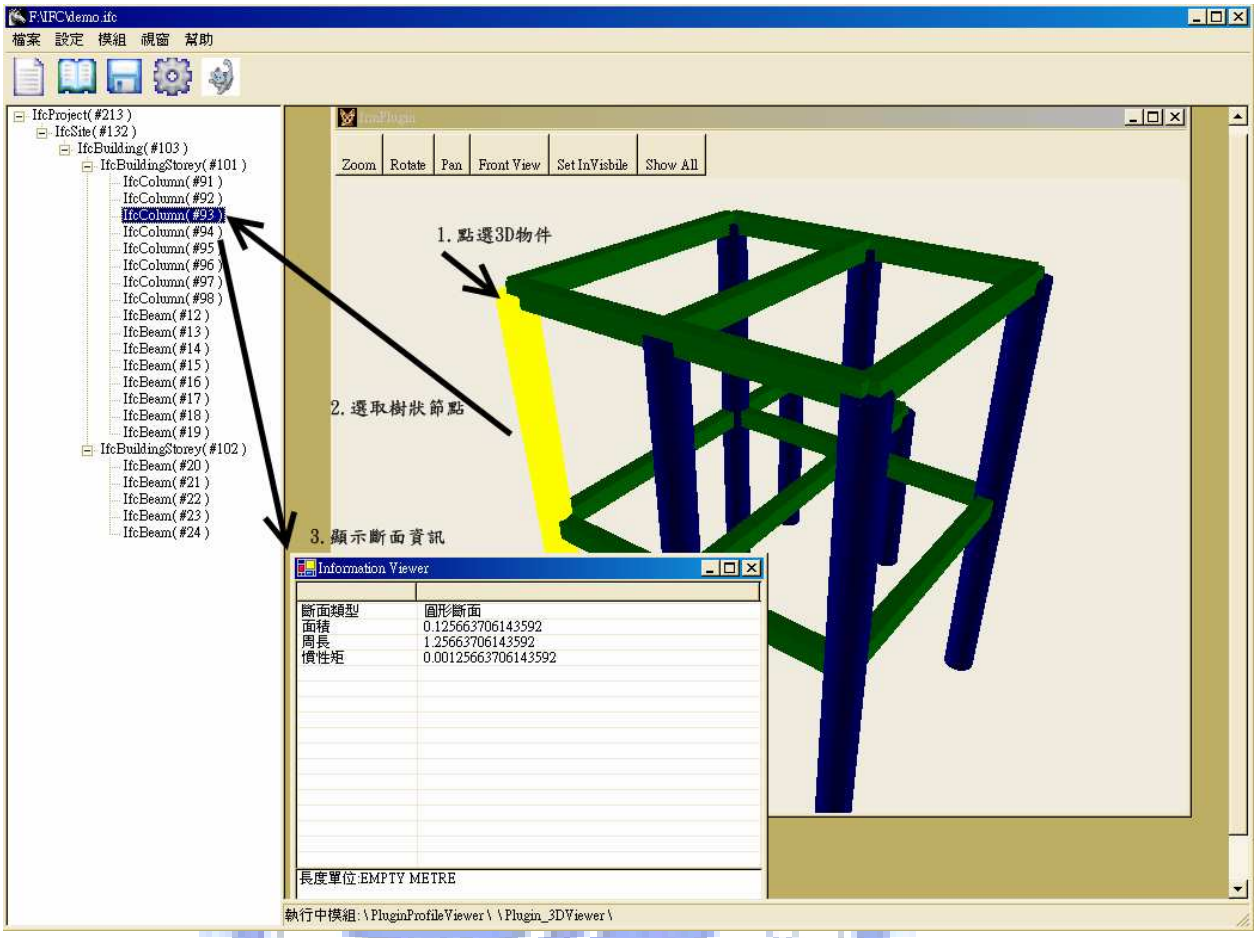

圖 5-30 樑柱斷面資訊瀏覽 Plug-In 元件-2

附件一

```
Public Class IfcCreator

    Public dirZ As IfcDirection
    Public dirX As IfcDirection
    Public dirY As IfcDirection
    Public carZero3D As IfcCartesianPoint
    Public carZero2D As IfcCartesianPoint
    Public Zero3D As IfcAxis2Placement3D
    Public Zero2D As IfcAxis2Placement2D
    Public reZero2D As IfcAxis2Placement2D

    Private Model As IFCsvr.Design
    Private OwnerHistory As IfcOwnerHistory
    Private GeometricRepresentationContext As IfcGeometricRepresentationContext
    Private Project As IfcProject
    Private Site As IfcSite
    Private SitePlacement As IfcLocalPlacement
    Private Building As IfcBuilding
    Private BuildingPlacement As IfcLocalPlacement
    Private agg_Site As IfcRelAggregates
    Private agg_Building As IfcRelAggregates
    Private agg_Storey As IfcRelAggregates
    Public nStories As Integer
    Public Stories(101) As StoreyData
    Public nBeam As Integer
    Public nColumn As Integer

    Public Sub New(ByRef _Model As IFCsvr.Design)
        Me.Model = _Model
        Me.OwnerHistory = Me.setOwnerHistory()
        Me.Project = Me.setProject

        Me.Site = Me.setSite
        agg_Site = New IfcRelAggregates
        agg_Site.GlobalId = Base.GenGuid
        agg_Site.OwnerHistory = Me.OwnerHistory
```

```
        agg_Site.Name_Optional = "ProjectContainer for Sites"
        agg_Site.RelatingObject = Me.Project.Entity
        ReDim agg_Site.RelatedObjects(1)
        agg_Site.RelatedObjects(1) = Site.Entity
        agg_Site.CreateEntity(Model)

        Me.Building = Me.setBuilding
        agg_Building = New IfcRelAggregates
        agg_Building.GlobalId = Base.GenGuid
        agg_Building.OwnerHistory = Me.OwnerHistory
        agg_Building.Name_Optional = "SiteContainer For Buildings"
        agg_Building.RelatingObject = Me.Site.Entity
        ReDim agg_Building.RelatedObjects(1)
        agg_Building.RelatedObjects(1) = Me.Building.Entity
        agg_Building.CreateEntity(Model)

        agg_Storey = New IfcRelAggregates
        agg_Storey.GlobalId = Base.GenGuid
        agg_Storey.OwnerHistory = Me.OwnerHistory
        agg_Storey.Name_Optional = "BuildingContainer for BuildigStories"
        agg_Storey.RelatingObject = Me.Building.Entity
        agg_Storey.CreateEntity(Model)

        nStories = 0
        nBeam = 0
        nColumn = 0

        carZero2D = New IfcCartesianPoint(0, 0)
        Zero2D = New IfcAxis2Placement2D
        Zero2D.Location = carZero2D
        Zero2D.RefDirection_Optional = New IfcDirection(1, 0)
        reZero2D = New IfcAxis2Placement2D
        reZero2D.Location = carZero2D
        reZero2D.RefDirection_Optional = New IfcDirection(0, 1)
        Model.FileAuthorisation = "Pingting54967"
    End Sub

    Private Function setOwnerHistory() As IFCool.IfcOwnerHistory
```

```
    Dim Actor As New IFCool.IfcActorRole
    Actor.Role = IFCool.Enumerations.IfcRoleEnum.ENGINEER

    Dim Person As New IFCool.IfcPerson
    Person.GivenName_Optional = "Pingting"

    Dim Organization As New IFCool.IfcOrganization
    ReDim Organization.Roles_Optional(1)
    Organization.Roles_Optional(1) = Actor

    Dim Application As New IFCool.IfcApplication
    Application.ApplicationDeveloper = Organization
    Application.ApplicationFullName = "project"
    Application.ApplicationIdentifier = "54967"
    Application.Version = "0.1"

    Dim OrgPer As New IFCool.IfcPersonAndOrganization
    OrgPer.TheOrganization = Organization
    OrgPer.ThePerson = Person

    Dim OwnerH As New IFCool.IfcOwnerHistory
    OwnerH.ChangeAction = IFCool.Enumerations.IfcChangeActionEnum.NOCHANGE
    OwnerH.CreationDate = 0
    OwnerH.State_Optional = IFCool.Enumerations.IfcStateEnum.READWRITE
    OwnerH.OwningApplication = Application
    OwnerH.OwningUser = OrgPer
    OwnerH.CreateEntity(Model)
    Return OwnerH
End Function

Private Function setUnit() As IfcUnitAssignment

    Dim siuL As New IFCool.IfcSIUnit
    siuL.UnitType = Enumerations.IfcUnitEnum.LENGTHUNIT
    siuL.Name = Enumerations.IfcSIUnitName.METRE

    Dim siuSL As New IFCool.IfcSIUnit
    siuSL.UnitType = Enumerations.IfcUnitEnum.AREAUNIT
```

```
        siuSL.Name = Enumerations.IfcSIUnitName.SQUARE_METRE

        Dim siuR As New IFCool.IfcSIUnit
        siuR.UnitType = Enumerations.IfcUnitEnum.PLANEANGLEUNIT
        siuR.Name = Enumerations.IfcSIUnitName.RADIAN

        Dim assUnit As New IfcUnitAssignment
        ReDim assUnit.Units(3)
        assUnit.Units(1) = siuL
        assUnit.Units(2) = siuSL
        assUnit.Units(3) = siuR
        assUnit.CreateEntity(Model)
        Return assUnit
    End Function

    Private Function setProject() As IfcProject
        Dim prj As New IfcProject
        prj.GlobalId = Base.GenGuid
        prj.OwnerHistory = Me.OwnerHistory
        prj.Name_Optional = "Pingting's Project"

        Dim rpc As New IfcGeometricRepresentationContext
        rpc.ContextType_Optional = "letMePass"
        rpc.ContextIdentifier_Optional = "Pingting"
        Dim plac3D As New IfcAxis2Placement3D
        plac3D.Location = New IfcCartesianPoint(0, 0, 0)
        plac3D.Axis_Optional = New IfcDirection(0, 0, 1)
        plac3D.RefDirection_Optional = New IfcDirection(1, 0, 0)
        Me.Zero3D = plac3D
        Me.dirX = plac3D.RefDirection_Optional
        Me.dirY = New IfcDirection(0, 1, 0)
        Me.dirZ = plac3D.Axis_Optional
        Me.carZero3D = plac3D.Location

        rpc.WorldCoordinateSystem = plac3D
        rpc.CoordinateSpaceDimension = 3
        GeometricRepresentationContext = rpc
        ReDim prj.RepresentationContexts(1)
```

```
        prj.RepresentationContexts(1) = rpc
        prj.UnitsInContext = Me.setUnit
        prj.CreateEntity(Model)
        Return prj
End Function

Private Function setSite() As IfcSite
        Dim sit As New IfcSite
        sit.GlobalId = Base.GenGuid
        sit.OwnerHistory = Me.OwnerHistory
        sit.Name_Optional = "Site"

        SitePlacement = New IfcLocalPlacement
        SitePlacement.RelativePlacement = Me.Zero3D
        sit.ObjectPlacement_Optional = SitePlacement

        sit.CompositionType = Enumerations.IfcElementCompositionEnum.ELEMENT
        sit.CreateEntity(Model)
        Return sit
End Function

Private Function setBuilding() As IfcBuilding
        Dim bud As New IfcBuilding
        bud.GlobalId = Base.GenGuid
        bud.OwnerHistory = Me.OwnerHistory
        bud.Name_Optional = "Building"

        BuildingPlacement = New IfcLocalPlacement
        BuildingPlacement.PlacementRelTo_Optional = SitePlacement
        BuildingPlacement.RelativePlacement = Me.Zero3D
        bud.ObjectPlacement_Optional = BuildingPlacement

        bud.CompositionType = Enumerations.IfcElementCompositionEnum.ELEMENT
        bud.CreateEntity(Model)
        Return bud
End Function

Public Function AddStorey(ByVal Name As String, ByVal Height As Double) As Integer
```

```
        nStories = nStories + 1
        Stories(nStories).Storey = New IfcBuildingStorey
        Stories(nStories).Storey.GlobalId = Base.GenGuid
        Stories(nStories).Storey.Name_Optional = Name
        Stories(nStories).Storey.OwnerHistory = Me.OwnerHistory
        Stories(nStories).Storey.CompositionType = IfcElementCompositionEnum.ELEMENT
        Stories(nStories).Storey.Elevation_Optional = Height

        Stories(nStories).Placement = New IfcLocalPlacement
        Stories(nStories).Placement.PlacementRelTo_Optional = Me.BuildingPlacement
        Dim plac3d As New IfcAxis2Placement3D
        plac3d.Axis_Optional = Me.dirZ
        plac3d.RefDirection_Optional = Me.dirX
        plac3d.Location = New IfcCartesianPoint(0, 0, Height)
        Stories(nStories).Placement.RelativePlacement = plac3d
        Stories(nStories).Storey.ObjectPlacement_Optional = Stories(nStories).Placement
        Stories(nStories).Storey.CreateEntity(Model)

        Stories(nStories).Contain = New IfcRelContainedInSpatialStructure
        Stories(nStories).Contain.GlobalId = Base.GenGuid
        Stories(nStories).Contain.Name_Optional = "for Building Element"
        Stories(nStories).Contain.OwnerHistory = Me.OwnerHistory
        Stories(nStories).Contain.RelatingStructure = Stories(nStories).Storey.Entity
        Stories(nStories).Contain.CreateEntity(Model)

        ReDim Preserve Me.agg_Storey.RelatedObjects(Me.nStories)
        Me.agg_Storey.RelatedObjects(Me.nStories) = Stories(nStories).Storey.Entity
        Me.agg_Storey.UpdateEntity()

        Return nStories
    End Function

    Public Function AddBeam(ByVal idxStorey As Integer, ByVal X1 As Double, ByVal Y1 As
Double, ByVal X2 As Double, ByVal Y2 As Double, ByVal Height As Double, ByRef Profile As
IfcProfileDef) As IFCool.IfcBeam

        nBeam = nBeam + 1
        Dim bem As New IfcBeam
```

```
bem.GlobalId = Base.GenGuid
bem.OwnerHistory = Me.OwnerHistory
bem.Name_Optional = "Beam:" & Me.nBeam.ToString

Dim blocal As New IfcLocalPlacement
blocal.PlacementRelTo_Optional = Me.Stories(idxStorey).Placement
Dim bplac3d As New IfcAxis2Placement3D
bplac3d.Location = New IfcCartesianPoint(X1, Y1, Height)
bplac3d.Axis_Optional = Me.dirZ
Dim mx As Double = X2 - X1
Dim my As Double = Y2 - Y1
Dim length As Double = (mx ^ 2 + my ^ 2) ^ 0.5
bplac3d.RefDirection_Optional = New IfcDirection(mx / length, my / length)
blocal.RelativePlacement = bplac3d
bem.ObjectPlacement_Optional = blocal
Dim shapeDef As New IfcProductDefinitionShape
Dim shapeRep As New IfcShapeRepresentation
shapeRep.ContextOfItems = Me.GeometricRepresentationContext
shapeRep.RepresentationIdentifier_Optional = "Body"
shapeRep.RepresentationType_Optional = "SweptSolid"

Dim extrud As New IfcExtrudedAreaSolid
Dim turnAround As New IfcAxis2Placement3D
turnAround.Location = Me.carZero3D
turnAround.Axis_Optional = Me.dirX
turnAround.RefDirection_Optional = Me.dirY
extrud.Position = turnAround
extrud.ExtrudedDirection = Me.dirZ
extrud.Depth = length
extrud.SweptArea = Profile

ReDim shapeRep.Items(1)
shapeRep.Items(1) = extrud
ReDim shapeDef.Representations(1)
shapeDef.Representations(1) = shapeRep

bem.Representation_Optional = shapeDef
bem.CreateEntity(Me.Model)
```

```
        Dim nObj As Integer
        If Me.Stories(idxStorey).Contain.RelatedElements Is Nothing Then
            nObj = 0
        Else
            nObj = Me.Stories(idxStorey).Contain.RelatedElements.Length - 1
        End If
        ReDim Preserve Me.Stories(idxStorey).Contain.RelatedElements(nObj + 1)
        Me.Stories(idxStorey).Contain.RelatedElements(nObj + 1) = bem.Entity
        Me.Stories(idxStorey).Contain.UpdateEntity()
        Return bem
    End Function

    Public Function AddColumn(ByVal idxStorey As Integer, ByVal X As Double, ByVal Y As
Double, ByVal Depth As Double, ByRef Profile As IfcProfileDef) As IFCool.IfcColumn
        nColumn = nColumn + 1
        Dim col As New IfcColumn
        col.GlobalId = Base.GenGuid
        col.OwnerHistory = Me.OwnerHistory
        col.Name_Optional = "Column:" & Me.nColumn.ToString

        Dim clocal As New IfcLocalPlacement
        clocal.PlacementRelTo_Optional = Me.Stories(idxStorey).Placement

        Dim cplac3d As New IfcAxis2Placement3D
        cplac3d.Location = New IfcCartesianPoint(X, Y, 0)
        cplac3d.Axis_Optional = Me.dirZ
        cplac3d.RefDirection_Optional = Me.dirX
        clocal.RelativePlacement = cplac3d
        col.ObjectPlacement_Optional = clocal

        Dim shapeDef As New IfcProductDefinitionShape
        Dim shapeRep As New IfcShapeRepresentation
        shapeRep.ContextOfItems = Me.GeometricRepresentationContext
        shapeRep.RepresentationIdentifier_Optional = "Body"
        shapeRep.RepresentationType_Optional = "SweptSolid"
        Dim extrud As New IfcExtrudedAreaSolid
        extrud.Position = Me.Zero3D
        extrud.ExtrudedDirection = Me.dirZ
```

```
        extrud.Depth = Depth
        extrud.SweptArea = Profile

        ReDim shapeRep.Items(1)
        shapeRep.Items(1) = extrud
        ReDim shapeDef.Representations(1)
        shapeDef.Representations(1) = shapeRep

        col.Representation_Optional = shapeDef
        col.CreateEntity(Me.Model)
        Dim nObj As Integer
        If Me.Stories(idxStorey).Contain.RelatedElements Is Nothing Then
            nObj = 0
        Else
            nObj = Me.Stories(idxStorey).Contain.RelatedElements.Length - 1
        End If
        ReDim Preserve Me.Stories(idxStorey).Contain.RelatedElements(nObj + 1)
        Me.Stories(idxStorey).Contain.RelatedElements(nObj + 1) = col.Entity
        Me.Stories(idxStorey).Contain.UpdateEntity()

        Return col
End Function

Public Function getStoreyIdx(ByVal Height As Double) As Integer
        For i As Integer = 1 To Me.nStories
            If Stories(i).Storey.Elevation_Optional = Height.ToString Then
                Return i
            End If
        Next
        Return -1
End Function

Public Function getStoreyIdx(ByVal Name As String) As Integer
        For i As Integer = 1 To Me.nStories
            If Stories(i).Storey.Name_Optional = Name Then
                Return i
            End If
        Next
```

```
            Return -1
        End Function

End Class

Public Structure StoreyData
    Public Storey As IfcBuildingStorey
    Public Contain As IfcRelContainedInSpatialStructure
    Public Placement As IfcLocalPlacement
End Structure
```

附件二

```
ISO-10303-21;
HEADER;
/* Generated by software containing ST-Developer
  * from STEP Tools, Inc. (www.steptools.com)
  */

FILE_DESCRIPTION(
/* description */ (''),
/* implementation_level */ '2;1');

FILE_NAME(
/* name */ 'demo',
/* time_stamp */ '2008-06-02T18:06:54+08:00',
/* author */ (''),
/* organization */ (''),
/* preprocessor_version */ 'ST-DEVELOPER v10',
/* originating_system */ '',
/* authorisation */ 'Pingting54967');

FILE_SCHEMA (('IFC2X3'));
ENDSEC;

DATA;
#10=IFCRECTANGLEPROFILEDEF(.AREA.,$,#29,0.2,0.25);
#11=IFCRECTANGLEPROFILEDEF(.AREA.,$,#29,0.2,0.25);
#12=IFCBEAM('HWKhb3ZOtUqTfsRGoPujOg',#301,'Beam:1',$,$,#159,#102,$);
#13=IFCBEAM('//0SGl5Il0+Jl9bxKOcAyQ',#301,'Beam:2',$,$,#160,#103,$);
#14=IFCBEAM('7deBrByUUE6XcFUqCwtWtw',#301,'Beam:3',$,$,#161,#104,$);
#15=IFCBEAM('wToXuep9jUixm04NrgABGw',#301,'Beam:4',$,$,#162,#105,$);
#16=IFCBEAM('g4hgAY/GIkGO2iul5lFSHg',#301,'Beam:5',$,$,#163,#106,$);
#17=IFCBEAM('9uu5qBy+MUiCkAgy+WCcIw',#301,'Beam:6',$,$,#164,#107,$);
#18=IFCBEAM('WgBXtM9drEii56pC5fS3XA',#301,'Beam:7',$,$,#165,#108,$);
#19=IFCBEAM('fBesuVqg6Eq8l4cghwuMtQ',#301,'Beam:8',$,$,#166,#109,$);
#20=IFCBEAM('SQY4vcDECU6AurytEon0Eg',#301,'Beam:9',$,$,#167,#110,$);
#21=IFCBEAM('Ndtsv8a/VEGaprkhYXY3iQ',#301,'Beam:10',$,$,#168,#111,$);
#22=IFCBEAM('gMee47T170K5u9p8t2z+KQ',#301,'Beam:11',$,$,#175,#118,$);
```

#23=IFCBEAM('pD3X1KZKgESKPNbNZU+Kfw',#301,'Beam:12',$,$,#176,#119,$);
#24=IFCBEAM('PJVM7LiWxU65RzOR+4fF7g',#301,'Beam:13',$,$,#177,#120,$);
#25=IFCBEAM('nW2DxMMl+UGtxlAn4cXHOw',#301,'Beam:14',$,$,#178,#121,$);
#26=IFCBEAM('uBBMm4u82kaHOcYN0X4KAw',#301,'Beam:15',$,$,#179,#122,$);
#27=IFCBEAM('zjTin/JNtUSfN2d2Hi+Adg',#301,'Beam:16',$,$,#180,#123,$);
#28=IFCBEAM('+srwGpUmSE28GOH9hEpbDQ',#301,'Beam:17',$,$,#181,#124,$);
#29=IFCAXIS2PLACEMENT2D(#208,#185);
#30=IFCCIRCLEPROFILEDEF(.AREA.,$,#29,0.2);
#31=IFCCIRCLEPROFILEDEF(.AREA.,$,#29,0.2);
#32=IFCEXTRUDEDAREASOLID(#30,#239,#183,2.8);
#33=IFCEXTRUDEDAREASOLID(#30,#239,#183,2.8);
#34=IFCEXTRUDEDAREASOLID(#30,#239,#183,2.8);
#35=IFCEXTRUDEDAREASOLID(#30,#239,#183,2.8);
#36=IFCEXTRUDEDAREASOLID(#30,#239,#183,2.8);
#37=IFCEXTRUDEDAREASOLID(#30,#239,#183,2.8);
#38=IFCEXTRUDEDAREASOLID(#30,#239,#183,2.8);
#39=IFCEXTRUDEDAREASOLID(#30,#239,#183,2.8);
#40=IFCEXTRUDEDAREASOLID(#10,#251,#183,3.5);
#41=IFCEXTRUDEDAREASOLID(#10,#253,#183,3.5);
#42=IFCEXTRUDEDAREASOLID(#10,#255,#183,3.5);
#43=IFCEXTRUDEDAREASOLID(#10,#257,#183,2.);
#44=IFCEXTRUDEDAREASOLID(#10,#259,#183,2.);
#45=IFCEXTRUDEDAREASOLID(#10,#261,#183,2.);
#46=IFCEXTRUDEDAREASOLID(#10,#263,#183,2.);
#47=IFCEXTRUDEDAREASOLID(#10,#265,#183,1.5);
#48=IFCEXTRUDEDAREASOLID(#10,#267,#183,1.5);
#49=IFCEXTRUDEDAREASOLID(#10,#269,#183,2.);
#50=IFCEXTRUDEDAREASOLID(#31,#239,#183,2.8);
#51=IFCEXTRUDEDAREASOLID(#31,#239,#183,2.8);
#52=IFCEXTRUDEDAREASOLID(#31,#239,#183,2.8);
#53=IFCEXTRUDEDAREASOLID(#31,#239,#183,2.8);
#54=IFCEXTRUDEDAREASOLID(#31,#239,#183,2.8);
#55=IFCEXTRUDEDAREASOLID(#31,#239,#183,2.8);
#56=IFCEXTRUDEDAREASOLID(#11,#277,#183,3.5);
#57=IFCEXTRUDEDAREASOLID(#11,#279,#183,3.5);
#58=IFCEXTRUDEDAREASOLID(#11,#281,#183,3.5);
#59=IFCEXTRUDEDAREASOLID(#11,#283,#183,2.);
#60=IFCEXTRUDEDAREASOLID(#11,#285,#183,2.);

#61=IFCEXTRUDEDAREASOLID(#11,#287,#183,2.);
#62=IFCEXTRUDEDAREASOLID(#11,#289,#183,2.);
#63=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#32));
#64=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#33));
#65=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#34));
#66=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#35));
#67=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#36));
#68=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#37));
#69=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#38));
#70=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#39));
#71=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#40));
#72=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#41));
#73=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#42));
#74=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#43));
#75=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#44));
#76=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#45));
#77=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#46));
#78=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#47));
#79=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#48));
#80=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#49));
#81=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#50));
#82=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#51));
#83=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#52));
#84=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#53));
#85=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#54));
#86=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#55));
#87=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#56));
#88=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#57));
#89=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#58));
#90=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#59));
#91=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#60));
#92=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#61));
#93=IFCSHAPEREPRESENTATION(#290,'Body','SweptSolid',(#62));
#94=IFCPRODUCTDEFINITIONSHAPE($,$,(#63));
#95=IFCPRODUCTDEFINITIONSHAPE($,$,(#64));
#96=IFCPRODUCTDEFINITIONSHAPE($,$,(#65));
#97=IFCPRODUCTDEFINITIONSHAPE($,$,(#66));
#98=IFCPRODUCTDEFINITIONSHAPE($,$,(#67));

#99=IFCPRODUCTDEFINITIONSHAPE($,$,(#68));
#100=IFCPRODUCTDEFINITIONSHAPE($,$,(#69));
#101=IFCPRODUCTDEFINITIONSHAPE($,$,(#70));
#102=IFCPRODUCTDEFINITIONSHAPE($,$,(#71));
#103=IFCPRODUCTDEFINITIONSHAPE($,$,(#72));
#104=IFCPRODUCTDEFINITIONSHAPE($,$,(#73));
#105=IFCPRODUCTDEFINITIONSHAPE($,$,(#74));
#106=IFCPRODUCTDEFINITIONSHAPE($,$,(#75));
#107=IFCPRODUCTDEFINITIONSHAPE($,$,(#76));
#108=IFCPRODUCTDEFINITIONSHAPE($,$,(#77));
#109=IFCPRODUCTDEFINITIONSHAPE($,$,(#78));
#110=IFCPRODUCTDEFINITIONSHAPE($,$,(#79));
#111=IFCPRODUCTDEFINITIONSHAPE($,$,(#80));
#112=IFCPRODUCTDEFINITIONSHAPE($,$,(#81));
#113=IFCPRODUCTDEFINITIONSHAPE($,$,(#82));
#114=IFCPRODUCTDEFINITIONSHAPE($,$,(#83));
#115=IFCPRODUCTDEFINITIONSHAPE($,$,(#84));
#116=IFCPRODUCTDEFINITIONSHAPE($,$,(#85));
#117=IFCPRODUCTDEFINITIONSHAPE($,$,(#86));
#118=IFCPRODUCTDEFINITIONSHAPE($,$,(#87));
#119=IFCPRODUCTDEFINITIONSHAPE($,$,(#88));
#120=IFCPRODUCTDEFINITIONSHAPE($,$,(#89));
#121=IFCPRODUCTDEFINITIONSHAPE($,$,(#90));
#122=IFCPRODUCTDEFINITIONSHAPE($,$,(#91));
#123=IFCPRODUCTDEFINITIONSHAPE($,$,(#92));
#124=IFCPRODUCTDEFINITIONSHAPE($,$,(#93));
#125=IFCCOLUMN('9tulbqFkmUWDQj7Fu3ULTw',#301,'Column:1',$,$,#151,#94,$);
#126=IFCCOLUMN('NSEuRofBOkq0dwSeaUz4tw',#301,'Column:2',$,$,#152,#95,$);
#127=IFCCOLUMN('bTl40kgJM06rc6dQ4qgLVg',#301,'Column:3',$,$,#153,#96,$);
#128=IFCCOLUMN('9qLmLgtPrkaqhUzWdIot6A',#301,'Column:4',$,$,#154,#97,$);
#129=IFCCOLUMN('d87Zcmw/a0az6T0O21KGLA',#301,'Column:5',$,$,#155,#98,$);
#130=IFCCOLUMN('b1UBRXQFjkyjX+Dm1WLubA',#301,'Column:6',$,$,#156,#99,$);
#131=IFCCOLUMN('srRip61t6kqFu4VOkKhEXA',#301,'Column:7',$,$,#157,#100,$);
#132=IFCCOLUMN('iXcpj91Q1EaKjVRhpQjcUw',#301,'Column:8',$,$,#158,#101,$);
#133=IFCCOLUMN('tca/lNppw0+Mqq0izHZPPw',#301,'Column:9',$,$,#169,#112,$);
#134=IFCCOLUMN('KO6TLP4lS0qVuKEfS/WLkw',#301,'Column:10',$,$,#170,#113,$);
#135=IFCCOLUMN('2QstRZWuz0m2pi0FA7WHVg',#301,'Column:11',$,$,#171,#114,$);
#136=IFCCOLUMN('gXUYBXHaOUq7/Cf90ZeUZg',#301,'Column:12',$,$,#172,#115,$);

#137=IFCCOLUMN('ChW5bK9rkEqGglmzMbL2dQ',#301,'Column:13',$,$,#173,#116,$);
#138=IFCCOLUMN('wClr8CdUkk6bXK6a3bOh4w',#301,'Column:14',$,$,#174,#117,$);
#139=IFCRELCONTAINEDINSPATIALSTRUCTURE('WFQ/F7/a8Ei69rCXVU/qyA',#301,
'for Building Element',$,(#125,#126,#127,#128,#129,#130,#131,#132,#12,#13,
#14,#15,#16,#17,#18,#19,#20,#21),#141);
#140=IFCRELCONTAINEDINSPATIALSTRUCTURE('RFisUWKe30CYHBFNYmQR9w',
#301,'for Building Element',$,(#133,#134,#135,#136,#137,#138,#22,#23,#24,#25,
#26,#27,#28),#142);
#141=IFCBUILDINGSTOREY('stsK+eiBKEeZuw0M3t2vfg',#301,'Ground',$,$,#149,$,
$,.ELEMENT.,0.);
#142=IFCBUILDINGSTOREY('6fpSqEB5xEuVOH7mBSdtGw',#301,'Floor:1',$,$,#150,
$,$,.ELEMENT.,2.8);
#143=IFCBUILDING('5C8uE4rnI0CKl5kDbeDJ2Q',#301,'Building',$,$,#148,$,$,.ELEMEN
T.,0.,0.,$);
#144=IFCRELAGGREGATES('xsbYfw7cIUu+hKH/giaTrA',#301,
'ProjectContainer for Sites',$,#291,(#182));
#145=IFCRELAGGREGATES('oYJoUxa00UmscQ4zsHqbMA',#301,
'SiteContainer For Buildings',$,#182,(#143));
#146=IFCRELAGGREGATES('SasGXQOfmUeyX3kDh304vg',#301,
'BuildingContainer for BuildigStories',$,#143,(#141,#142));
#147=IFCLOCALPLACEMENT($,#239);
#148=IFCLOCALPLACEMENT(#147,#239);
#149=IFCLOCALPLACEMENT(#148,#240);
#150=IFCLOCALPLACEMENT(#148,#241);
#151=IFCLOCALPLACEMENT(#149,#242);
#152=IFCLOCALPLACEMENT(#149,#243);
#153=IFCLOCALPLACEMENT(#149,#244);
#154=IFCLOCALPLACEMENT(#149,#245);
#155=IFCLOCALPLACEMENT(#149,#246);
#156=IFCLOCALPLACEMENT(#149,#247);
#157=IFCLOCALPLACEMENT(#149,#248);
#158=IFCLOCALPLACEMENT(#149,#249);
#159=IFCLOCALPLACEMENT(#149,#250);
#160=IFCLOCALPLACEMENT(#149,#252);
#161=IFCLOCALPLACEMENT(#149,#254);
#162=IFCLOCALPLACEMENT(#149,#256);
#163=IFCLOCALPLACEMENT(#149,#258);
#164=IFCLOCALPLACEMENT(#149,#260);

#165=IFCLOCALPLACEMENT(#149,#262);
#166=IFCLOCALPLACEMENT(#149,#264);
#167=IFCLOCALPLACEMENT(#149,#266);
#168=IFCLOCALPLACEMENT(#149,#268);
#169=IFCLOCALPLACEMENT(#150,#270);
#170=IFCLOCALPLACEMENT(#150,#271);
#171=IFCLOCALPLACEMENT(#150,#272);
#172=IFCLOCALPLACEMENT(#150,#273);
#173=IFCLOCALPLACEMENT(#150,#274);
#174=IFCLOCALPLACEMENT(#150,#275);
#175=IFCLOCALPLACEMENT(#150,#276);
#176=IFCLOCALPLACEMENT(#150,#278);
#177=IFCLOCALPLACEMENT(#150,#280);
#178=IFCLOCALPLACEMENT(#150,#282);
#179=IFCLOCALPLACEMENT(#150,#284);
#180=IFCLOCALPLACEMENT(#150,#286);
#181=IFCLOCALPLACEMENT(#150,#288);
#182=IFCSITE('l0ECTjHl50Gl/HiuoxMw9A',#301,'Site',$,$,#147,$,$,.ELEMENT.,
(0,0,0),(0,0,0),0.,$,$);
#183=IFCDIRECTION((0.,0.,1.));
#184=IFCDIRECTION((1.,0.,0.));
#185=IFCDIRECTION((1.,0.));
#186=IFCDIRECTION((1.,0.));
#187=IFCDIRECTION((0.,1.,0.));
#188=IFCDIRECTION((1.,0.));
#189=IFCDIRECTION((1.,0.));
#190=IFCDIRECTION((0.,1.));
#191=IFCDIRECTION((0.,1.));
#192=IFCDIRECTION((0.,1.));
#193=IFCDIRECTION((0.,1.));
#194=IFCDIRECTION((1.,0.));
#195=IFCDIRECTION((1.,0.));
#196=IFCDIRECTION((0.,1.));
#197=IFCDIRECTION((1.,0.));
#198=IFCDIRECTION((1.,0.));
#199=IFCDIRECTION((1.,0.));
#200=IFCDIRECTION((0.,1.));
#201=IFCDIRECTION((0.,1.));

#202=IFCDIRECTION((0.,1.));
#203=IFCDIRECTION((0.,1.));
#204=IFCCARTESIANPOINT((0.,0.,0.));
#205=IFCCARTESIANPOINT((0.,0.,0.));
#206=IFCCARTESIANPOINT((0.,0.,2.8));
#207=IFCCARTESIANPOINT((0.,0.,0.));
#208=IFCCARTESIANPOINT((0.,0.));
#209=IFCCARTESIANPOINT((3.5,0.,0.));
#210=IFCCARTESIANPOINT((3.5,2.,0.));
#211=IFCCARTESIANPOINT((3.5,4.,0.));
#212=IFCCARTESIANPOINT((0.,2.,0.));
#213=IFCCARTESIANPOINT((0.,4.,0.));
#214=IFCCARTESIANPOINT((5.,4.,0.));
#215=IFCCARTESIANPOINT((5.,2.,0.));
#216=IFCCARTESIANPOINT((0.,0.,2.8));
#217=IFCCARTESIANPOINT((0.,2.,2.8));
#218=IFCCARTESIANPOINT((0.,4.,2.8));
#219=IFCCARTESIANPOINT((0.,0.,2.8));
#220=IFCCARTESIANPOINT((0.,2.,2.8));
#221=IFCCARTESIANPOINT((3.5,0.,2.8));
#222=IFCCARTESIANPOINT((3.5,2.,2.8));
#223=IFCCARTESIANPOINT((3.5,2.,2.8));
#224=IFCCARTESIANPOINT((3.5,4.,2.8));
#225=IFCCARTESIANPOINT((5.,2.,2.8));
#226=IFCCARTESIANPOINT((0.,0.,0.));
#227=IFCCARTESIANPOINT((3.5,0.,0.));
#228=IFCCARTESIANPOINT((0.,2.,0.));
#229=IFCCARTESIANPOINT((3.5,2.,0.));
#230=IFCCARTESIANPOINT((0.,4.,0.));
#231=IFCCARTESIANPOINT((3.5,4.,0.));
#232=IFCCARTESIANPOINT((0.,0.,2.8));
#233=IFCCARTESIANPOINT((0.,2.,2.8));
#234=IFCCARTESIANPOINT((0.,4.,2.8));
#235=IFCCARTESIANPOINT((0.,0.,2.8));
#236=IFCCARTESIANPOINT((0.,2.,2.8));
#237=IFCCARTESIANPOINT((3.5,0.,2.8));
#238=IFCCARTESIANPOINT((3.5,2.,2.8));
#239=IFCAXIS2PLACEMENT3D(#204,#183,#184);

#240=IFCAXIS2PLACEMENT3D(#205,#183,#184);
#241=IFCAXIS2PLACEMENT3D(#206,#183,#184);
#242=IFCAXIS2PLACEMENT3D(#207,#183,#184);
#243=IFCAXIS2PLACEMENT3D(#209,#183,#184);
#244=IFCAXIS2PLACEMENT3D(#210,#183,#184);
#245=IFCAXIS2PLACEMENT3D(#211,#183,#184);
#246=IFCAXIS2PLACEMENT3D(#212,#183,#184);
#247=IFCAXIS2PLACEMENT3D(#213,#183,#184);
#248=IFCAXIS2PLACEMENT3D(#214,#183,#184);
#249=IFCAXIS2PLACEMENT3D(#215,#183,#184);
#250=IFCAXIS2PLACEMENT3D(#216,#183,#186);
#251=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#252=IFCAXIS2PLACEMENT3D(#217,#183,#188);
#253=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#254=IFCAXIS2PLACEMENT3D(#218,#183,#189);
#255=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#256=IFCAXIS2PLACEMENT3D(#219,#183,#190);
#257=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#258=IFCAXIS2PLACEMENT3D(#220,#183,#191);
#259=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#260=IFCAXIS2PLACEMENT3D(#221,#183,#192);
#261=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#262=IFCAXIS2PLACEMENT3D(#222,#183,#193);
#263=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#264=IFCAXIS2PLACEMENT3D(#223,#183,#194);
#265=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#266=IFCAXIS2PLACEMENT3D(#224,#183,#195);
#267=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#268=IFCAXIS2PLACEMENT3D(#225,#183,#196);
#269=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#270=IFCAXIS2PLACEMENT3D(#226,#183,#184);
#271=IFCAXIS2PLACEMENT3D(#227,#183,#184);
#272=IFCAXIS2PLACEMENT3D(#228,#183,#184);
#273=IFCAXIS2PLACEMENT3D(#229,#183,#184);
#274=IFCAXIS2PLACEMENT3D(#230,#183,#184);
#275=IFCAXIS2PLACEMENT3D(#231,#183,#184);
#276=IFCAXIS2PLACEMENT3D(#232,#183,#197);
#277=IFCAXIS2PLACEMENT3D(#204,#184,#187);

#278=IFCAXIS2PLACEMENT3D(#233,#183,#198);
#279=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#280=IFCAXIS2PLACEMENT3D(#234,#183,#199);
#281=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#282=IFCAXIS2PLACEMENT3D(#235,#183,#200);
#283=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#284=IFCAXIS2PLACEMENT3D(#236,#183,#201);
#285=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#286=IFCAXIS2PLACEMENT3D(#237,#183,#202);
#287=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#288=IFCAXIS2PLACEMENT3D(#238,#183,#203);
#289=IFCAXIS2PLACEMENT3D(#204,#184,#187);
#290=IFCGEOMETRICREPRESENTATIONCONTEXT('Pingting','letMePass',3,0.,#239,$);
#291=IFCPROJECT('+kE9+FQBxUODGjuiU+YEjA',#301,'Pingting''s Project',$,$,
$,$,(#290),#295);
#292=IFCSIUNIT(*,.LENGTHUNIT.,$,.METRE.);
#293=IFCSIUNIT(*,.AREAUNIT.,$,.SQUARE_METRE.);
#294=IFCSIUNIT(*,.PLANEANGLEUNIT.,$,.RADIAN.);
#295=IFCUNITASSIGNMENT((#292,#293,#294));
#296=IFCAPPLICATION(#298,'0.1','project','54967');
#297=IFCACTORROLE(.ENGINEER.,$,$);
#298=IFCORGANIZATION($,$,$,(#297),());
#299=IFCPERSON($,$,'Pingting',(),(),(),(),());
#300=IFCPERSONANDORGANIZATION(#299,#298,());
#301=IFCOWNERHISTORY(#300,#296,.READWRITE.,.NOCHANGE.,0,$,$,0);
ENDSEC;
END-ISO-10303-21;