國立臺灣大學工學院土木工程學系

碩士論文

Department of Civil Engineering

College of Engineering

National Taiwan University

Master Thesis

BIM 模型修改之可互操作物件式資訊回饋系統

An Interoperable Object-based Information Feedback System for BIM Model Modification

陳奐廷

Huan-Ting Chen

指導教授：謝尚賢 博士

Advisor: Shang-Hsien Hsieh, Ph.D.

中華民國 103 年 7 月

July 2014

誌謝

總是得之於人者太多，卻又不能只謝天，惟感激涕零之俗不可免。

首先，我要非常感謝授業恩師謝尚賢教授，從大學到碩士班跟隨謝老師做研究，耳濡目染，如沐春風，所謂傳道、授業、解惑者，皆傾囊相授，不一而足，更重要的是謝老師給我很大的自由和彈性，在學業表現上始能有揮灑自如的空間。

再者，我要感謝擔任論文口試委員的郭榮欽教授，郭老師身懷老驥伏櫪之志，總是融實務經驗於學術研究，將專業知識娓娓道來，受益無窮，對我的鼓勵和肯定必當銘記於心，虛懷若谷，不敢托大。

我也要感謝與郭老師偕同擔任論文口試委員的周建成教授和吳翌禎教授，周老師和吳老師的寶貴建議，使得論文能盡善盡美，不勝感激。

台大土木 CAE 組是思想多元、融洽無間的研究空間，老師們平易近人，智慧和經驗皆與時俱進，感謝陳俊杉教授和康仕仲教授的提攜照顧，也感謝邱仁鈿教授返組傳授新知，醍醐灌頂，薪火相傳。

此外，我也要感謝所有 CAE 組的同學、學長、學姊、學弟、學妹和助理們，良朋益友，肝膽相照，切磋琢磨，浴乎沂，風乎舞雩，詠而歸，人生樂事，不外如是。

要感謝的人誠然難以盡數，我的摯友們，在此一併謝過，相逢自是有緣，願心領神會。

最後，我要感謝我的家人們，雖離鄉背井，惟親情的溫暖猶天涯咫尺，感謝你們的支持和體諒，銘感五內，無以復加。論文口試後，外公由阿彌陀佛接應至西方極樂，祝福外公在佛國淨土精進修行，早日成就佛道。

文末，以四字箴言告別我的求學生涯：「增重，再見。」

摘要

BIM (Building Information Modeling)模型由分屬各種領域的(multidisciplinary)模型元件(model element)所構成，模型元件則包含眾多參數化屬性(parametric attributes)，而參數化屬性又可分為定量的(quantitative)幾何資訊和定性的(qualitative)非幾何資訊等，故在 BIM 的技術面、流程面及應用面，資訊管理皆是不可或缺的重要議題。

BIM 在資訊管理層面有兩大特色：其一為工程資訊的生命週期管理，因建築物或設施歷經規劃、設計、施工乃至營運維護等不同階段，其生命週期所涵蓋之時間相當長，而 BIM 模型又隨生命週期而有不同的發展程度(LOD, Level Of Development)，故 BIM 模型的生命週期資訊量多且雜，恰好呼應近年來巨量資料(big data)的發展趨勢；其二為跨領域的(interdisciplinary)資訊整合，來自不同領域的專家透過協同作業使工程專案之效率提升，而領域知識(domain knowledge)不同則語意資訊(semantic information)亦不同。

因此，工程專案在生命週期中經常變動的特性使得 BIM 模型(即 BIM 模型元件中的參數化屬性)必須隨之修改或更新，而此般修改或更新又必須根據來自不同領域的利害關係人(stakeholder)所提供正確合宜之更新數據，故 BIM 模型的修改可謂一種資訊回饋的流程。

有鑑於 BIM 模型修改之異質(heterogenous)協同設計需求，本研究首先針對不同的 BIM 模型修改應用情境進行分析，並歸納提出「應用情境導向之資訊管理框架」，在此框架下可開發一「可互操作物件式資訊回饋系統」，以提升 BIM 模型修改或更新流程之效率。本系統以資料庫來管理回饋資訊，並設計相應的管理介面予模型管理者(model manager)，讓其可透過本系統向利害關係人取得回饋資訊，並能在模型元件的物件層級(object-level)自動地修改或更新 BIM 模型，省去手動操作修改模型的繁複步驟。此外，本系統自行設計免綱要式(schema-free)的資料結構，並開發資訊回饋端的網路應用程式，以增強不同 BIM 工具之間的相互操作性(interoperability)，而能滿足跨平台、非同步的(asynchronous)協同設計需求。

關鍵詞：建築資訊塑模、BIM、模型修改、資訊回饋、相互操作性

Abstract

A BIM (Building Information Modeling) model consists of multidisciplinary model elements. A model element includes numerous parametric attributes, such as quantitative geometric information and qualitative non-geometric information. Hence, information management is a significant issue for BIM technology, process, and applications.

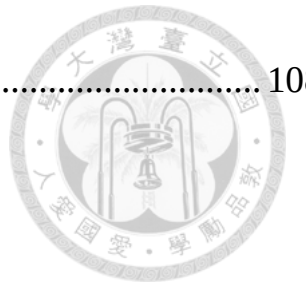
There are two key features of information management with regard to BIM: lifecycle management of engineering information and integration of interdisciplinary information. Due to the long time span of a building lifecycle through different phases, from project planning, design, construction, to operation and maintenance, there is an abundance of complicated lifecycle information with regard to the model elements with different LODs (Level Of Development). In addition, different disciplines provide a variety of domain knowledge, and the productivity and efficiency increase through collaboration among domain experts. The semantic information is also diverse in accordance with the differences in domain knowledge.

In sum, a BIM model always needs to be modified or updated because of the frequent changes in a building project lifecycle. In other words, the parametric attributes within the model elements need to be modified or updated. The aforementioned process can be defined as information feedbacks because each modification or update is based on the correct data provided by stakeholders from different domains.

Concerning the needs for collaboration of BIM model modification, this research conducted a scenario analysis and proposed a scenario-oriented information management framework. Based on the framework, an interoperable object-based information feedback system was developed to improve the efficiency of the process. This system managed the feedback information by using a database approach. A corresponding management user interface was designed for a model manager to manipulate feedback information provided by stakeholders. A BIM model could also be automatically modified or updated at object-level of its model elements through the designed system instead of operating the model modification manually. In addition, this research designed a unique schema-free data structures and developed a web application for the information feedback end in order to enhance the interoperability among different BIM tools and fulfill the needs for cross-platform and asynchronous collaboration.

Keywords: Building Information Modeling (BIM), model modification, information feedback, interoperability

目錄



圖目錄

表目錄

第一章　緒論

1.1　研究緣起

此研究為 BIM (Building Information Modeling)技術(Eastman et al., 2011)的衍生應用，濫觴於主題為「LiveBIM」(Hsieh et al., 2013)之國科會研究計畫，該研究擬以具經驗參數之動態 BIM 模型提昇公共生活空間生命週期管理，由於未來將有大量的分析後資訊必須整合至 BIM 模型，於是開始發想如何設計一套資訊交換與管理機制，此機制除了需要擷取 BIM 模型中之資訊以進行分析和模擬，更重要的是必須能夠讓分析和模擬後的資訊回饋至 BIM 模型。

經過文獻回顧與市場調查，就自動化和批次處理的需求而言，從 BIM 模型中擷取資訊之方法相當多，然而能夠自動化且大量地將資訊整合、反饋回 BIM 模型之方法則相對少，即便有也存在技術門檻較高、相互操作性差或效率不彰等問題。

反思，建築物之生命週期如此冗長，BIM 模型元件之發展程度(LOD, Level Of Development)也越分越細，BIM 模型在一個工程專案中隨生命週期發展、傳遞，BIM 模型本來就應該會經常變動(或謂成長)，而且是由許多不同領域專家(domain expert)來共同創造、維護、整合同一組 BIM 模型，故 BIM 模型的修改需求理應相當殷切才是，於是本研究開始由此需求面來思考。

經過歸納與整理，本研究認為 BIM 模型修改之應用情境可分為三大類(第二章將詳述)，而其核心步驟即是將修改或更新後的資訊或資料回饋至 BIM 模型元件之參數化屬性，同時應以物件資訊交換之觀點來管理，故一個系統化的概念始成，即物件參數化屬性的資訊交換與管理機制，其中資訊交換必須是雙向的，也就是包含資訊回饋的部分，並且這整個系統和機制的設計冀望能符合實際需求，即能適用於 BIM 模型修改之各種應用情境。

因此，本研究的理念為：技術必須以需求來推動，即整個機制和系統的設計必須以 BIM 模型修改的應用情境為導向，整個系統核心可作為資訊溝通、交換、

傳遞和儲存的居中媒介，BIM 模型管理者(model manager)和其他工程專案利害關係人(stakeholder)則分居左右，前者為資訊管理和請求者，後者則為資訊提供和回饋者，本研究係以「資訊回饋至 BIM 模型」之協同設計需求為主要考量。

1.2 研究背景與現況分析

1.2.1 研究背景

BIM 的相互操作性(interoperability)是個極為重要的議題，甚至還有專門的國際性組織負責推廣(例如：先前的 the International Alliance for Interoperability (IAI)，現為 buildingSMART)。就本研究的觀點而言，相互操作性無可避免地必須受 BIM 設計工具(通常也是建模工具)所箝制，原因很顯而易見：若 BIM 模型的修改或更新可被認定是一種重新設計或再造的過程，吾人沒有理由不使用 BIM 設計工具來操作此一流程，否則 BIM 設計工具便沒有存在的必要，除非有朝一日所有的 BIM 設計工具均支援完全的相互操作性(full interoperability)，或是市面上僅剩一種 BIM 設計工具。前者根據六年前 Pazlar and Turk (2008)的研究看來是不切實際的(unrealistic)，若以 Steel et al. (2012)的研究來衡量，本人認為直到現在也是；而後者則並不符合 BIM 至少現階段的生態發展趨勢。

既然本研究主張 BIM 模型修改時的相互操作性將為所使用的 BIM 設計工具所限制，本研究理當有責任先釐清何謂相互操作性，但由於相互操作性有許多不同的切入點和定義，此處僅就論文標題所宣稱的「可互操作」加以定義，而本研究研發之系統如何得以達到相互操作性則留待 3.1 節再論。

所謂「可互操作」之英文為 interoperable，此詞彙已被收錄在牛津字典(Oxford Dictionaries)當中，因此吾人可認定此詞彙有其通俗版本的解釋，其定義翻譯成中文為「電腦系統或軟體之間有能力交換(exchange)和利用資訊」，而 exchange 在牛津字典中的英文解釋可翻譯為「給予某物並且收到同類型的某物以作為回報」，由此可見，所謂「可互操作」必定具有雙向溝通和交換的特性，若以物件資訊的

單方面角度來看，則包含資訊請求端和資訊回饋端兩部分。就本研究的著眼點而言，「可互操作」乃指 BIM 模型管理者可依據現有的模型資訊向利害關係人提出請求，此為資訊請求端；而利害關係人根據模型管理者所提供之模型資訊，再提供其新的模型修改建議，此為資訊回饋端。理論上，資訊請求端掌控 BIM 模型之管理權限；資訊回饋端則握有正確合宜的修改或更新之資訊，兩端所處之平台、環境和所使用之軟體、系統，很有可能不具有直接進行資訊交換的途徑，甚至兩端人員所處之時間和空間亦無法同步作業，有關現階段的相互操作性解決方案和限制將留待 2.6 節再行說明，以下先針對 BIM 模型的資訊擷取和資訊回饋進行回顧。

BIM 模型主要由建模(modeling)工具來建置，故建模工具是最基礎的 BIM 設計工具，若將建模工具視為一個獨立的電腦系統，則將資訊從此系統提取出來或存放進去的動作，即是一種資訊擷取或資訊回饋的操作，而在目前市場佔有率較高的幾個建模工具中，具備資訊擷取的功能和方法較多，資訊回饋的功能和方法則較少，且普遍存在技術門檻較高、相互操作性較差和效率不彰等問題。

事實上，資訊回饋至 BIM 模型之前往往先有資訊擷取的過程，因為資訊回饋端至少必須知道 BIM 模型的專案資訊、物件識別碼、參數化屬性識別碼、參數資料格式等基本資訊，因此資訊擷取的方法將會影響資訊回饋的途徑(approach)，以下就目前市面上三大建模工具(Autodesk Revit, Graphisoft ArchiCAD, Bentley AECOsim)共同有的功能來分析各種 BIM 模型資訊交換(即包含擷取和回饋)之方法或途徑，並以入門或操作門檻之高低順序來闡述。

(1) 電子資料表(spreadsheet)

大部分的 BIM 設計工具皆有輸出明細報表(schedule)之功能，並可以在輸出後其他試算表軟體(如：Microsoft Excel)來編輯，但若要修改表格中的屬性值或即時反映(即動態交互參照)修改模型後的屬性值，則必須回到原本

採用的 BIM 設計工具來操作，除非使用 1.2.3 節將介紹的商業套件，但此方法仍會有一些限制，將在 1.2.4 節探討。基本上，電子資料表位於 BIM 設計工具的內部環境，難以與外界溝通，也就是連基本的相互操作性都有蠻大的問題，因此較適合用於資訊擷取和輸出，資訊回饋則較不適用。

(2) 公開標準格式(open standard)

與 BIM 相關的公開標準格式很多，可以達到一定程度相互操作性的卻很少，這些公開標準較知名的有：IFC (Industry Foundation Classes)、COBie (Construction Operations Building Information Exchange)、gbXML (Green Building XML)等等。目前大部分的 BIM 設計工具皆支援 IFC 格式的檔案輸出，而 COBie、gbXML 等其他公開標準格式也或多或少有支援，即便沒有支援亦可透過非官方之外掛程式(plug-in)來輸出該格式之檔案。公開標準是一種追求完美的理想，IFC 的相互操作性在近年來的確有所提升，而 COBie 和 gbXML 等格式則較不具通用性，因 COBie 和 gbXML 分別較適用在設備管理和能源分析。但整體而言，無論何種公開標準，在資訊回饋方面的相互操作性仍有待改善，可以想像的是：以公開標準格式將所需資訊從 BIM 設計工具中擷取出來，相較於用同樣格式回饋至相同(甚至不同)的 BIM 設計工具容易太多了，其使用限制和詳細原因將在 2.6.3 節探究。

(3) 開放資料庫互連(ODBC, Open DataBase Connectivity)

用外部資料庫來管理 BIM 模型中的資訊不失為一個解決相互操作性的好方法，而 ODBC 是個發展已久的標準資料庫介面，但由於資料庫的操作門檻和維護成本皆不低，若模型管理者和利害關係人欲使用資料庫來進行資訊交換，額外開發資料庫應用程式和依實際需求來客製化使用介面或許是必要的。另一方面，ODBC 的應用實例大多搭配傳統常用的關聯式

(relational)資料庫，而關聯式資料庫的資料綱要(schema)對於 BIM 模型多變且複雜的物件導向(object-oriented)結構在進行物件關聯對應(ORM, Object Relational Mapping)時很難概括承受，這也是本研究決定採用免綱要式(schema-free)的 NoSQL 資料庫之原因，有關適用於 BIM 模型的資料庫選用原則將在 3.4 節和 4.2 節再論。

(4) 應用程式介面(API, Application Programming Interface)

大部分的 BIM 設計工具皆提供 API 讓具程式開發能力之使用者延伸其核心功能，資訊擷取和回饋當然是非常普遍的應用，但 API 的使用門檻又比用資料庫來管理高出許多，因使用 API 來進行資料交換除了必須具備物件導向的程式開發能力，還需對 BIM 設計工具的物件核心架構相當了解，且不同的 BIM 設計工具有不同的 API，更添難度，因此模型管理者和利害關係人欲直接透過 API 來進行資料交換可謂強人所難。此外，API 的應用範圍和彈性雖高，但若只「單純」採用 API 來開發 BIM 設計工具的外掛程式，則又回到以檔案為基礎(file-based)之資料交換模式，對於協同作業、版本管控和相互操作性而言不見得較有效率。

以上四種 BIM 模型的資訊交換方法或途徑各有其優劣，更詳細的使用限制剖析待 1.2.4 節再論，接下來先回顧一些實務上有關「將資訊回饋至 BIM 模型」的學術研究和商業套件，以驗證本研究的論述。

1.2.2 學術研究回顧

在學術文獻中，以資料庫或API之方式將資訊回饋至BIM模型的研究並不多，本研究主要回顧一個以資料庫來更新設備管理資訊的研究，以及一個以 API 來開發參數化設計輔助工具的研究。

Liu and Issa (2012)以資料庫來管理設施維護之相關資訊，並在 Autodesk Revit 平台將共用參數(shared parameter)以 ODBC 介面(採用官方的 Revit DB Link 套件)

之方式匯出至關聯式資料庫進行管理，同時和 BIM 模型進行雙向的資訊傳遞，該研究同時提及 BIM 設計工具或建模軟體的屬性並不完全適用於設施維護管理，因設施維護管理所需之資訊不會是 BIM 模型內全部的資訊，而且操作建模軟體需要經過特別訓練，讓設施維護管理人員來操作具有客製化使用者介面的系統，並以資料庫來管理或許較恰當，該研究還提到另一個現實面考量：BIM 設計工具通常購置成本極高，這也是本研究提倡專業分工(將在 2.3 節詳述)的原因。

Yenerim and Yan (2011)以 Autodesk Revit 之 API 來開發複雜屋頂結構之參數化設計輔助工具。該研究的參數化設計和本研究經常提及之參數化屬性不太一樣，所謂參數化設計乃指透過參數數值來達到複雜曲面、結構或外觀的建築構件之設計需求，該研究首先將複雜的參數式屋頂結構元件設計好，再透過 Revit API 將幾何變數等資訊匯出至試算表軟體，當然也可將此資料表視為一種外部資料庫，這麼做的目的是為了建立屋頂和樓板之間的物件幾何關係，亦即需要經過複雜演算過程的物件幾何關係和參數值可先在試算表軟體先設計好，再透過 API 將設計結果回饋至 BIM 模型中。

1.2.3 商業套件回顧

從商業套件(commercial package)的發展可觀察資訊回饋至 BIM 模型之應用趨勢和限制，除了 1.2.1 節所提到的 ODBC 介面為主要 BIM 設計工具的資料庫交換模式外，一些商業套件主要基於 BIM 設計工具之 API 來進行外掛程式的開發，考量多數 BIM 使用者並不具資料庫管理能力和程式開發能力，而公開標準格式的相互操作性又有待改善，故這些商業套件主要開發了客製化的使用者介面，透過 API 讓 BIM 設計工具內部的資料表或明細表可和外部試算表軟體互通有無。因試算表軟體和表格型態的操作介面對於辦公室軟體(office suite)來說已發展地相當成熟，舉 Autodesk Revit 的外掛程式為例，至少有三種類似的商業套件，例如：Ideate, Inc. 之 BIMLink、i-Theses 之 BIMiTs extensions for Revit platform 以及 BIMCODER, Inc.

之 Schedule Sync，皆可讓使用者將明細表匯出至 Microsoft Excel 軟體，在試算表中做完修改後，再由其外掛程式匯入至 Autodesk Revit，同時模型元件的參數化屬性再連動地被修改，事實上這些外掛程式僅是簡化操作 ODBC 資料庫連接的流程或步驟而已，根本上脫離不了關聯式資料綱要(即以表格欄位為面向)的範疇。

這些外掛程式對於模型管理者處理一些參數化屬性固然方便(例如：1.2.2 節所提到的參數化設計便不必大費周章開發外掛程式來應付資料輸出、輸入的需求(Yenerim and Yan, 2011))，但對於多方合作的協同作業便不是那麼有效率，若模型管理者和利害關係人需要不停傳遞資料表之檔案，抑或將資料庫管理權限進一步地(即使是部分)分享給所有利害關係人，在管理流程上就顯得複雜許多了。

1.2.4 使用限制分析

本節針對 1.2.1 節至 1.2.3 節的回顧內容，作一綜合性的限制分析。由於這些方法仍然無法完全解決將資訊回饋至 BIM 模型之實際需求，惟關於 IFC 之相互操作性議題已申明將在 2.6.3 節再論，原因是 IFC 仍絕對有其存在的必要性。

首先，仍必須再度清楚定義何謂「資訊回饋至 BIM 模型」，始能充分了解 1.2.1 至 1.2.3 節所述方法之使用限制。誠如 1.2.1 節所述，資訊交換必然是雙向的，且非一人所能完成的，故資訊回饋可視為 BIM 模型資訊交換的「回程」，當資訊回饋至 BIM 模型時，代表著 BIM 模型中的模型元件必須被修改(modification)、修正(revision)、更新(update)或再造(reproduction)，而由於自動化和批次處理之需求，手動操作 BIM 設計工具來修改模型將顯得效率低落，於是有內部電子資料表、公開標準格式、資料庫管理、API 外掛程式開發和一些商業套件的替代方法產生。

大致上，資訊回饋至 BIM 模型的定義和方法如上述，接著以條列之方式來描述這些資訊回饋方法的使用限制：

(1) 商業套件和 ODBC 介面支援的資料庫種類太少，目前仍以一種試算表軟體(Microsoft Excel)和兩種傳統關聯式資料庫(Microsoft Access, Microsoft SQL Server)為主。

(2) 資料庫管理的工作流程太繁複，技術性門檻也不低，模型管理者和利害關係人若要使用則必須額外開發資料庫應用程式，且模型存取權限的管控和彈性沒有 API 來得高。

(3) 關聯式資料綱要是以表格欄位為面向，故符合此綱要的資料結構(包含：電子資料表、關聯式資料庫、COBie 等)皆無法處理非結構化的資料(unstructured data)，而模型元件卻會隨著生命週期發展而有不同的 LOD，因此模型元件的參數化屬性可能是經常變動的，甚至連參數資料的數值單位也有可能改變。

(4) 若單純以資料表格型式來交換資料，BIM 模型便喪失視覺化的功能，除非額外將幾何資料以其他格式輸出為 BIM 模型瀏覽器之可讀格式。

(5) BIM 模型是以物件導向技術來建構模型元件，若以關聯式資料綱要來儲存模型元件中的資訊，則資料結構中或需要額外記錄最新的物件交互參照索引值(key)，並且必須了解物件之間的繼承(inheritance)關係，以免修改參數化屬性值時牽一髮而動全身。

(6) 物件導向中尚有多型(polymorphism)的概念，因此一個模型元件可能會有多重的身分，例如：一道牆的實作元件(instance)可能是一扇窗的主體元件(host object)，也可能是構成一個房間邊界片段(room boundary segment)的類別成員(class member)，而此類別成員卻是該主體元件的父類別(superclass)。若以關聯式資料綱要來儲存模型元件時，還要設法記錄和判斷該物件的執行期型別資訊(RTTI, RunTime Type Information)，否則根據實際需求的異同，資料表的數量可能會非常多。

總之，關聯式資料綱要並不能完全滿足 BIM 技術架構的儲存需求，且關聯式資料庫用來儲存模型資訊的效率不見得比較高，此部分於 3.4 節將再說明。然而，API 的彈性和應用層面雖廣，但客製化的技術門檻卻很高，且資料庫管理模式的確較以檔案為基礎的管理模式有許多優點，因此再度回到「資訊回饋至 BIM 模型」的需求面來看，應思考如何整合公開標準、資料庫和 API 的長處，才是本研究的最佳解決方案，故將上述這些途徑或方法截長補短並加以延伸，可定調為本研究之研究方法。

1.3 研究目的

本研究之主要目的係考量BIM模型修改之異質(heterogeneous)協同設計需求，開發一可互操作物件式資訊回饋系統，以供資訊請求端模型管理者和資訊回饋端利害關係人使用。由於工程專案參與者不一定具備操作 BIM 設計工具之能力，又或因為跨平台和非同步作業之需求，當資訊回饋端使用者無法直接操作 BIM 設計工具或無法直接以檔案方式進行資料交換時,本研究開發之系統可作為解決方案。本研究首先針對不同的 BIM 模型修改應用情境進行分析，並歸納提出「應用情境導向之資訊管理框架」，在此框架下可開發一「可互操作物件式資訊回饋系統」，以提升 BIM 模型修改或更新流程之效率。本系統以資料庫來管理回饋資訊，並設計相應的管理介面予模型管理者,讓其可透過本系統向利害關係人取得回饋資訊，並能在模型元件的物件層級(object-level)自動地修改或更新 BIM 模型,省去手動操作修改模型的繁複步驟。此外,本系統自行設計免綱要式(schema-free)的資料結構，並開發資訊回饋端的網路應用程式，以增強不同 BIM 工具之間的相互操作性(interoperability)，而能滿足跨平台、非同步的(asynchronous)協同設計需求。

1.4 研究方法與流程

在 1.2 節中所提及的許多資訊交換之途徑和方法，由於其各有優、缺點，故截長補短或許是可行的解決方案，惟必須回歸到資訊回饋至 BIM 模型的需求面來思

考，故先進行應用情境之分析，然後歸納提出一個相應的資訊管理架構，再根據此大架構來設計和實作可普遍適用(general)於各應用情境之資訊回饋系統，故本節依據整體研究的邏輯順序來條列研究方法與流程，當中包含：方法論、方法、流程和寫作邏輯。此外，由於研究流程和本論文之撰寫架構相符，故本節之研究方法與流程亦依照論文架構來闡述。

(1) 需求分析

首先，會需要將資訊回饋至 BIM 模型的應用情境有很多，故應回歸到需求面來探討這些應用情境的相關性，尋求一個普遍適用的資料交換通則和找出關鍵步驟，否則欲開發的系統將無法兼顧多數人的需求(在佛法中稱法門無量誓願學)，那還不如根據需求去開發一個客製化的外掛程式。因此在第二章將針對應用情境作全面的分析，並進行專家訪談以證實所分析的應用情境在實務上確有其需求，接著再針對現有的相互操作性解決方案進行分析。因相互操作性是在異質系統之間進行資料交換的必要條件，在了解現有解決方案之限制後，最後再針對其不足之處歸納出一個可行的資訊管理應用架構。

(2) 系統分析

在分析完應用情境之後，可在其資訊管理應用架構下，實現一個以應用情境為導向的資訊框架。所謂架構指的是 architecture，而框架指的是 framework，故架構比較屬於概念性的整體雛形或應用模式，而框架則有需要詳細定義所採用的方法、應用的層面、適用的對象以及運用的流程，故第三章初始可先將此框架框好，便可釐清欲開發系統之產品定位，同時檢查其運作機制和操作邏輯等，最後就整體系統的相容性、延伸性、可擴展性與安全性作進一步的探討。

(3) 系統設計

系統設計則主要是程式設計的工作，故第四章會描述整體系統設計的過程，從版本規劃開始確立整體系統開發的收斂範圍以免無限上綱，接著在此收斂範圍內進行更細部的架構設計，其中包含：資料儲存方式、資料管理方法、資訊傳遞方法、各應用端的程式設計等，並將程式設計結果以圖示之。

(4) 系統實作

第五章為實作部分，由於程式碼部分不若使用者介面可視覺化呈現，故必須清楚交代所採用工具的使用方法，以及如何按照原先計劃的設計模式來進行程式實作，接著展示系統設計成果之操作方法，最後則進行使用者訪談、測試與驗證，以了解整體系統是否已設計完善。

(5) 結論

第六章作為總結可分為兩部分來談：其一為本研究之整體貢獻，即所研發之系統可解決問題的應用範圍和限制；其二為本研究之未來展望，即描述本研究尚未完善之處及可延伸、加值應用之方向，以待有緣人能更進一步、持續深入研究。

第二章　應用情境分析

2.1　BIM 模型修改之協同作業模式

考量到建築構件和設施在施工和營運維護階段經常變動的特性，早在 2009 年便有學者指出 BIM 模型需隨生命週期更新的需求(Akcamete et al., 2009)，該研究以案例研究的方式來分析工程專案中的 RFI (Request For Information)及 RCO (Request for Change Order)等文件，將這些設計變更以不同的生命週期、維護模式、發生頻率、物件變更類型以及物件變更對其他物件之影響予以分析、比較，並強調 BIM 設計工具對於計算支援(computational support)的重要性，以檢核參數化模型元件之間的相關性、連動性、一致性。

上述研究的重點在於探究模型元件之間的邏輯關係，而本研究的重點則在於「為資訊回饋至 BIM 模型的流程設計一套管理機制和應用程式系統」，故技術應以實務上的需求來推動，即整個機制和系統的設計必須以 BIM 模型修改的應用情境為導向。

若由需求面來考量這些應用情境，其對於 BIM 模型修改的共同需求其實可歸因於協同作業的需求。BIM 的協同作業模式有許多分類法，若依據目的或用途的不同可分為協同設計(collaborative design)和協同整合(collaborative integration)，若依據操作環境的不同則可分為同質協同作業(homogeneous collaboration)和異質協同作業(heterogeneous collaboration)。

協同設計主要著重 BIM 塑模的分工合作，多使用者(multi-user)透過建模工具在虛擬空間進行設計工作，例如：Autodesk Revit Server 的中央檔案、Graphisoft ArchiCAD BIM Server 的 Teamwork 等(Chen and Hou, 2014)；協同整合主要著重 BIM 模型和生命週期資訊的管理和協調，多使用者透過管理系統來進行模型整合、版次管理、衝突檢測、專案資訊交換等，例如：Bentley ProjectWise、Autodesk Vault 等(Shafiq et al., 2013)。故本研究之「BIM 模型修改」係屬「協同設計」之範疇。

同質協同作業指的是多使用者在相同的操作平台或環境下進行協同作業，而異質協同作業則代表多使用者所使用的操作平台或環境不同(Li et al., 2007; Huang et al., 2009)。根據 1.2.1 節所述，在 BIM 模型修改的資訊傳遞過程中，由於資訊回饋端使用者不具備或無法直接操作 BIM 建模工具，故本研究之「BIM 模型修改」係屬「異質協同作業」之範疇。

根據上述兩種 BIM 協同作業模式之分類，本研究係針對「異質協同設計」之需求來進行系統開發，因為 BIM 模型的建置需要隨生命週期發展，過程中又必須和許多工程專案參與者或利害關係人共同合作、互相交換資訊，從而產生兩個協同作業流程中的特性或需求：其一為跨平台作業，因協同作業者所使用之設備、平台、環境可能不盡相同，又或協同作業者可能不具備操作 BIM 設計工具之能力(如：資訊回饋端使用者無法操作 BIM 建模工具)；其二為非同步作業，因協同作業者可能身處不同時間或空間(如：人在工地或國外)，而無法進行同步作業。

因此，本章從 BIM 模型在其生命週期中的發展開始聯想，將其在實務上可能的應用情境歸納為三大類，並在 2.2 節至 2.4 節中詳述，必須強調的是這些應用情境旨在描述一種意圖(intent)或需求，即「因有此意圖或需求，故在這樣的情境下需要一個更健全的(robust)資訊回饋方式」，而非單純指「在此應用情境下會需要修改或更新 BIM 模型」，因此應用情境是以「因」來分類和定義，而有些應用情境看起來非常近似，抑或不同類型的應用情境在實務上或許會融合在一起，則是因緣和合所呈現之「果」，惟此般細節之處在描述應用情境時將再說明。

另外，所有的應用情境皆有一個大前提：因為本研究之目的為進行 BIM 模型的修改或更新，故所適用的應用情境必然已具備至少一個建置得差不多的 BIM 模型。這個前提並不是在要求模型元件的發展程度，而是指在某一個生命週期階段的建模作業必須達到一定的完成度，本研究可協助進行小部分的模型修改或更新，而並非將資訊回饋的協同作業機制加諸於最初的建模作業流程中。

2.2 節至 2.4 節分別敘述三大類的應用情境，總計有十二種應用情境。為了驗證這些應用情境在實務上是否確有其需求，2.5 節將分析專家訪談的結果，並且為了方便比較各應用情境之訪談結果，三種不同類別的應用情境將一同以數字來進行編號。另一方面，相互操作性是在異質系統之間進行資料交換的必要條件，故 2.6 節針對現有的解決方案進行分析，然後在 2.7 節再針對其不足歸納出一個可行的資訊管理應用架構。

2.2 團隊間情境

所謂「團隊間」情境的英文為 inter-team scenario，指的是資訊回饋至 BIM 模型的過程中，進行資訊交換的雙方分屬不同的團隊(註：1.2.1 節已說明資訊回饋前往往先有資訊擷取，故以資訊交換稱之)，若沿用 1.2.1 節的描述，即「資訊請求端」和「資訊回饋端」分屬不同的團隊，而所謂「團隊」則無明確定義，因目前的 BIM 作業流程和運作模式在不同的公司可能會有很大的差異(蔡孟君, 2013)，故「團隊間」可能是兩家不同的公司，抑或同一家公司裡面的兩個不同部門，以下(1)至(5)為五個團隊間情境的應用情境描述。

(1) 結構分析或細部結構設計之結果反饋至建築模型

建築專案通常會使用許多不同廠商(vendor)的軟體以完成建築設計、結構分析、專案管理等特定任務，故資料和資訊在專案參與各方之間共享之需求殷切，而在異質環境共享資料則需要各方皆有一個共同的資料模型(Shen et al., 2010)。事實上，在三維空間將結構構件進行結構分析的歷史比 BIM 技術更為悠久，因此結構分析軟體的歷史當然也比 BIM 設計工具要久，而結構分析軟體和 BIM 設計工具之間的資料相互操作性一直以來都有許多相關的學術研究，此處舉一較完整的期刊論文為例：Jeong et al. (2009)的研究實際測試預鑄混凝土構件之 BIM 設計工具和建築設計之 BIM 設計工具之間的相互操作性，該研究提到 IFC 格式雖是目前較好的

解決方案，但完整而有效的相互操作性卻有很大的改善空間。就建築結構的設計流程而言，當建築師初步完成量體配置後，將此建築模型交由結構工程師進行細部結構設計或進行結構分析，結構工程師只需要調整而非完全重建一個新的建築模型(Porwal and Hewage, 2013)，而雙方所使用的設計工具非常有可能是不同的，倘若結構分析或細部結構設計完成後，斷面尺寸或構件位置必須調整，則建築模型也應當一並修改，此時本應用情境將產生「結構細節」資訊回饋至 BIM「建築」模型之需求。

(2) 能源分析或節能設計之結果反饋至建築模型

綠建築或節能設計是近年來相當熱門的研究領域，而基於和結構分析軟體相同的理由，能源分析軟體和 BIM 設計工具的相互操作性一直以來也有許多學術研究，且由於能源分析之面相較廣(如：能耗、照明、通風等)，不同能源分析軟體或有不同的模擬引擎，故能源分析軟體的檔案格式也相當多(如：gbXML、IDF、INP 等)，其相互操作性則又更複雜許多(Maile et al., 2007)，這也是將能源分析獨立為一個應用情境的原因。此外，由於綠建築必須經過官方認證，故沒有經過認證的模擬引擎所分析之結果只能作為參考。另外，能源分析有時只需要外殼部分的建築模型，故建築師初步完成量體配置後，若欲進行能源分析尚需將模型先依需求處理好再行匯出，而當設計量體需要依據能源分析結果進行修改時，此時本應用情境將產生「能源分析」資訊回饋至 BIM「建築」模型之需求。

(3) RFI (Request For Information)

工程專案資訊經常會以工程圖說和規範(specifications)的型式提供給承包商(contractor)和其轉包商(subcontractor)(以下合稱為包商)，當這些專案文件不完整、有衝突或發生錯誤時，包商可訴諸 RFI 這個正式程序來取得或澄清額外需要的資訊(Mohamed et al., 1999)。雖然 BIM 技術的導入已被

公認可有效降低 RFI 之頻率(Azhar, 2011)，但實務上仍不可能完全沒有 RFI 之需求，故當包商自行建置或整合 BIM 模型時，若發現有疑義仍必須發佈 RFI，當其所要求之資訊必須修改或更新部分 BIM 模型元件時，包商必須按照指示進行修改模型的動作，此時本應用情境將產生「RFI」資訊回饋至 BIM 模型之需求。

(4) 不同生命週期階段的模型(元件)交付需建置(或整合成)新模型

目前 BIM 技術尚未全面普及，採用整合性專案交付(IPD, Integrated Project Delivery)的案例也不多，故並非每個生命週期階段皆導入 BIM 技術，這也是本研究將此應用情境和上一個 RFI 應用情境分開之原因。RFI 應用情境並不限於專案參與者必須導入 BIM，而本應用情境則主要針對 BIM 模型在不同生命週期階段進行交付的情況，故隱含著交付 BIM 模型的雙方皆導入 BIM，即乙方有意圖將甲方所交付其之 BIM 模型進行重新建置或整合。BIM 模型之內容與細節取決於該生命週期階段之實際應用需求，故乙方並不會(也不需要)將所有資訊整合至單一 BIM 模型，甚至會發展出自己所需之 BIM 模型(謝尚賢, 2013)。例如：營造廠(假設為乙方)若欲根據BIM模型來出施工圖，其大多自行重建新的BIM模型(因需求不同)，惟仍必須參考上一個發展程度的 BIM 模型來建置(若甲方具備的話)，故乙方欲建置或整合的新BIM模型或會需要參考甲方所提供之原有BIM模型，若甲方所提供之 BIM 模型有相互操作性之問題(例如：甲方提供之模型檔案格式或版本並非完全相容於乙方之平台或環境)，則雙方必須設法排除此問題(責任或需取決於契約)，但無論解決之道為何，此時本應用情境將產生「原有 BIM 模型」資訊回饋至「新」BIM 模型之需求。

(5) 模型交由行政機關審批或交由業主審核

目前國內外已有不少學術研究和業界應用案例將 BIM 技術應用於自動化法規檢測，且在不同生命週期階段皆有應用的可能性(郭榮欽, 2013)。另一方面，BIM 模型的三維視覺化特性亦可提供業主更直觀的資訊呈現，故業主可藉由 BIM 模型來了解是否滿足其需求。因此，BIM 模型交由行政機關審批或交由業主審核應為未來之發展趨勢，若行政機關或業主認為某些工程細節需要調整或修改，則 BIM 模型或可能也需要一併進行修改或更新，此時本應用情境將產生「行政機關或業主之意見」資訊回饋至 BIM 模型之需求。

2.3 團隊內情境

所謂「團隊內」情境的英文為 intra-team scenario，指的是資訊回饋至 BIM 模型的過程中，進行資訊交換的雙方屬於同一個的團隊，若沿用 1.2.1 節的描述，即「資訊請求端」和「資訊回饋端」分屬同一個團隊，而所謂「團隊內」則如同 2.2 節所解釋，可能是同一家公司，抑或同一家公司裡面的同一個部門，以下(6)至(8)為三個團隊內情境的應用情境描述。

(6) 設計人員與建模人員之專業分工

BIM 技術的發展或使得製圖員(draftman)漸漸消失，取而代之的則是建模員(modeler)和模型管理者(model manager)，以應付 BIM 模型的建置和管理需求(Porwal and Hewage, 2013)。人類社會的分工合作已越趨專業化，在尚未有 BIM 的年代，從事設計的工程師和負責 CAD (Computer-Aided Design)製圖的人員也經常是採取專業分工，而 BIM 技術的導入更衍生出模型管理之人力需求，故專業分工理論上應更加細緻，即設計人員、建模人員、模型管理者應各司其職，尤其設計人員有其專業領域背景，更應讓其專注於專業設計工作，在目前若欲設計人員「精通」相較於 CAD

製圖軟體更為複雜的 BIM 設計工具未免太強人所難，設計人員應只需要知道以 BIM 技術進行設計之基本概念，而複雜的建模工作應交給專業建模人員和模型管理者來負責，即設計人員可將其設計成果交予建模人員來建置 BIM 模型，即使在不久的將來，設計人員也具備直接使用 BIM 設計工具的能力，但 BIM 模型有許多建置上的操作細節仍可由建模人員來負責，設計人員只需要將基本的設計概念用 BIM 設計工具來呈現即可。綜合上述，此時本應用情境將產生「設計人員」所提供之資訊經由「建模人員」回饋至 BIM 模型之需求。

(7) 最佳化軟體授權使用

BIM 設計工具通常售價不菲，除非公司之規模夠大，否則很難以大量授權之方式將軟體安裝在公司所保管之裝置上，即便以網路授權之方式可安裝之裝置或可超過軟體授權數目，但同時可進行作業之裝置仍以實際購入之軟體授權數目為限，因此大部分導入 BIM 之公司仍會在成本考量之下僅採購有限數目的軟體授權。在此限制下，軟體授權若能被妥善利用始能將成本效益最大化，因為在某些情況下，使用者並非真正具有操作 BIM 設計工具之需求，例如：僅修改部分專案資訊或某些模型元件之參數化屬性資料，而非必須利用 BIM 設計工具所提供之使用者操控介面來進行視覺化輔助設計或建模。可行的替代方案有很多，例如：更詳細的專業分工安排、開發專屬流程之 API 外掛程式、開發資料庫應用程式並搭配資料庫來管理等等，但無論採用何種方法，若原欲直接操作 BIM 設計工具者現在改為不直接操作，則此時本應用情境將產生「間接操作者」所提供之資訊經由「直接操作者」回饋至 BIM 模型之需求。

(8) 節省高效能硬體設備購置費用或跨平台應用

此應用情境和上一個頗為類似，上一個應用情境源自於「軟體成本」之考量，而此應用情境則屬於「硬體成本」之考量。由於 BIM 設計工具之硬體規格需求頗高，尤其實務上更要求操作的流暢度和即時反應，故硬體設備之採購亦是一筆為數不小的開銷，雖然虛擬桌面基礎架構(VDI, Virtual Desktop Infrastructure)(郭榮欽 et al., 2013)或精簡客戶端(thin client)或許可將硬體需求集中化(centralized)管理，但建置類似的高規格雲端環境亦需要額外成本，且軟體授權費用是相同的，故此類雲端環境較適合規模較大的公司。此外，並非每一台裝置所使用之作業系統和軟體環境皆支援 BIM 設計工具，甚至時間或空間也會限制人員僅能使用行動裝置(如：在工地時使用手機或平板電腦較方便)，此為非同步(asynchronous)協同作業之需求，故諸如此類以「硬體考量」為出發點之應用情境，若能透過其他媒介來修改或更新 BIM 模型(如：建置跨平台之雲端 BIM 模型伺服器)，則此時本應用情境將產生「非 BIM 設計工具裝置」所提供之資訊經由「網路環境」回饋至 BIM 模型之需求。

2.4 經驗式情境

所謂「經驗式」情境的英文為 empirical scenario，指的是資訊回饋至 BIM 模型的過程中，進行資訊交換的雙方不一定屬於同個或不同團隊，但最重要的特徵為：「資訊回饋端」必須將某些「經驗」提供予「資訊請求端」，所謂「經驗」可能是建築物的營運維護資料、物業管理資訊、設備感測器(sensor)資料、使用者經驗、現場測量資訊等等，以下(9)至(12)為四個經驗式情境的應用情境描述。

(9) LiveBIM 架構下的參數回饋機制

LiveBIM 研究計畫(Hsieh et al., 2013)為本研究計畫之緣起，該研究計畫擬研發能將使用經驗納入 BIM 模型之參數中之技術，以提昇公共生活空間

(設施)之生命週期管理。目前的 BIM 研究多在於模擬實體空間(設施)的幾何、物理屬性參數及空間關係，尚未見從使用者的角度來思考 BIM 之生活空間(設施)生命週期管理應用，將社會環境變遷及人與空間的互動行為等因素考量進 BIM 模擬中，故該研究導入目前在智慧生活科技領域已採用的 Living Lab(生活實驗室)技術，於實際的生活環境中挑選一些具代表性之空間來建構具可操控及可量測性之生活實驗室，然後從實驗觀察中建立可用來模擬使用者行為之參數模型，並動態地擷取所需之參數值，建構具經驗參數之動態 BIM 模型，以供回饋生命週期管理中不同階段之應用。在此研究計畫中尚有許多不同的研究議題和其應用情境(如：智慧監測、防災疏散、空間管理、營運維護等)，故當某個研究議題欲將其所分析或模擬之使用者行為模式與 BIM 模型之參數化屬性結合時(如：依據使用者經驗最佳化之逃生模式可回饋至 BIM 模型中之隔間配置)，此時即產生「使用者經驗」資訊回饋至 BIM 模型之需求。

(10) 物業管理或營運維護管理之生命週期資訊回饋

營運與維護(O&M, Operation and Maintenance)階段往往是建築物生命週期中歷時最久的，在此階段中建築物會需要修繕、更新，建築設備也會需要檢查、替換，甚至室內擺設、隔間和裝潢也可能有很大的變化，故 2.1 節所提到 Akcamete et al. (2009)之研究才會指出 BIM 模型有可能經常需要更新。事實上，本應用情境和上一個應用情境也有點類似，上一個應用情境主要針對「使用者經驗」進行資訊回饋，而本應用情境則單純指建築物「生命週期資訊」之回饋，例如：以物業管理之角度來看，某個設備更換零件後應會需要調整 BIM 模型元件之參數，抑或某個傢俱擺放位置改變之後英也會需要調整 BIM 模型元件之三維空間定位。在這些情況下，此時即產生「生命週期資訊」資訊回饋至 BIM 模型之需求。

(11) 建置既有建築物模型之現場量測資訊回饋

雖然目前已有許多技術可使用雷射掃描之點雲(point cloud)來自動化建置現有(as-built)建築物之 BIM 模型，但亦有不少技術缺口(technology gap)需要花費不少時間以人工來處理，故現有建築物之自動化建模還有很長的路要走(Tang et al., 2010)。由此可見，利用點雲的輔助並不能完全滿足既有建築物的建模需求，或有可能需要到現場量測以獲得更精確之空間幾何資訊，此時即產生「現場量測數據」資訊回饋至 BIM 模型之需求。

(12) BIM 建模教學輔助工具

無論產業界導入 BIM 技術或學術界開授 BIM 課程，兩者皆免不了建置 BIM 模型的建模教學，而初學者往往無法在短時間內學會所有的建模技巧和參數設定方式，具有經驗的建模教學者或指導者有時亦不能即時提供初學者正確的解答(如：非面對面之教學)，若能讓初學者先將建模過程中所遇到之問題以某種形式發佈至另一媒介(如：以截圖之方式發佈至協同作業平台)，待教學者能提供其正確解答時，再透過該媒介將正確的建模資訊(如：正確的參數設定方式)回饋給初學者，此時即產生「教學者」所提供之資訊回饋至「初學者」以正確建置 BIM 模型之需求。

2.5 專家訪談之需求分析

2.2 節至 2.4 節敘述十二種應用情境，本研究係以資訊請求端和資訊回饋端之協同作業條件來分類，而非以生命週期來區分，主要考量為資料交換的管理模式較接近，惟生命週期不同階段的工作流程仍有差異。為了確認這十二種應用情境在實務上確有其需求，本研究透過專家訪談的方式來進行需求分析。

本研究一共進行五次的專家訪談，訪問對象和其他與會者皆具有五年以上的 BIM 相關經驗，茲以訪談順序列名單如下：

- 某營造公司之主任技師(以下簡稱 A)

- 某 BIM 服務公司之總經理和研發部研發組組長(以下簡稱 B 和 C)
- 某建築師事務所之主持人兼設計總監(以下簡稱 D)
- 某營造公司之建築專案部副理(以下簡稱 E)
- 某工程顧問公司之研發及資訊部經理、技師和 BIM 部門建築師、工程師(以下簡稱 F、G、H、I)

大部分的受訪者皆認為，這十二種應用情境在實務上大多有其需求，故本研究僅列出受訪者認為「在實務上較不易存在此種應用情境」的情況，並配合 2.2 節至 2.4 節之應用情境編號順序以條列式說明(註:A 對於應用情境(12)未表示意見)：

- E 認為應用情境(4)所描述的情況在實務上較不易發生，因甲、乙雙方在簽訂契約前應該先就相互操作性議題進行協調，若事後才發現或發生不可預期的相互操作性問題，則往往資料轉換會過於複雜而無法解決。
- E 認為應用情境(6)所描述的情況在實務上較不易發生，因為 BIM 技術在未來之發展，理想上設計人員應該也具備操作 BIM 設計工具之能力。
- A 認為應用情境(7)所描述的情況在實務上較不易發生，因為欲導入 BIM 技術的單位本應將軟體費用視為導入成本,即BIM軟體為基本而必要的開銷，故並無節省軟體費用之需求。

2.6 相互操作性之解決方案分析

本章到目前為止已歸納十二種有關「資訊回饋至 BIM 模型」之應用情境，並將其歸納為三大類，經過 2.5 節之專家訪談後，亦可認定每一種應用情境皆或多或少有其實務需求，故接著可探究這些需求是否有普遍可行的解決之道。然而，在 1.2.1 節已說明「資訊回饋」基本上隱含著「雙向資料交換」，且「雙向資料交換」能力又是「相互操作性」之必要條件，故可從相互操作性為切入點。1.2.1 節的資料交換方法或途徑基本上是「事後補救」，而以下要提的相互操作性解決方案則傾向於「事前預防」，即希望藉由採取這些方案，盡量避免相互操作性之問題。

根據英國皇家建築師學會(RIBA, Royal Institute of British Architects)之國家建築規範(NBS, National Building Specification)網站對於 BIM 之相互操作性敘述(Hamil, 2012)，相互操作性有三個層級(level)：同一供應商之軟體相互操作性、不同供應商之軟體相互操作性、公開標準之相互操作性。本人認為，這三個層級恰好是目前在實務上可能(但不一定最適合)之解決方案，因這些方式皆各有擁護者，且本人對於前兩種解決方案亦有一些親身經驗，至於第三種則是全球學術界普遍和廣泛研究的議題，故本 2.6 節將依序來探討。

2.6.1 同一供應商之軟體相互操作性

供應商的英文是 vendor，而軟體供應商通常指的就是開發軟體的公司，假若在工程專案的全生命週期中，無論 BIM 設計工具或非 BIM 設計工具(如：模型瀏覽軟體)皆採用同一家供應商的軟體，乍聽之下似乎相互操作性的問題會比較少，且供應商也有義務去解決自家產品的相互操作性，否則怎說服消費者花大把鈔票去購買如此昂貴的軟體？誠然，本人曾在臺灣某個知名的半導體公司擔任實習生，該公司從早期廠房設計的多維度模擬軟體到近幾年先在辦公大樓導入 BIM 技術，皆貫徹採用同一家軟體供應商之理念，在廠房設計的部分多年來可謂相當成功，但此般成功卻也造成 BIM 技術導入於廠房設計的考慮因素，故其先於辦公大樓的營運維護階段先嘗試以 BIM 模型來輔助管理。

事實上，BIM 技術可應用於全生命週期，不同階段、不同領域、不同需求皆有其最適合的軟體選擇，因此 Hamil (2012)也提到一個建築物不可能完全都選擇同一家軟體供應商之軟體來進行設計和建造。BIM 技術與應用講究跨領域整合和多方協同作業，即便每一個專案參與者在每一個時間點皆使用同一家軟體供應商之軟體，或許還有以下因素需考量：

(1) 同一家公司的產品若是併購或收購而來，其相互操作性如何？

(2) 同一家公司的產品若是屬不同領域，其相互操作性如何？(如：機台設備之三維模型是否可完全相容於 BIM 設計工具)

(3) 同一個軟體如果不同領域的功能太多，其個別功能的專業度是否可被使用者信賴？又或軟體供應商應如何定價？使用者是否買單？

(4) 同一個軟體的不同版本或無法向下支援(即舊版軟體無法讀取新版格式)

2.6.2 不同供應商之軟體相互操作性

商用軟體的檔案格式或多或少是商業機密，又因底層資料模型(data model)不可能完全一樣，故不同軟體供應商的軟體要用相同方法來解析其檔案格式難度頗高，除非具有市場區隔或功能性不同的軟體供應商才有可能存在完全的相互操作性。然而，出於商業競合的考量，或許也會有兩家公司願意合作設法使其旗下軟體之相互操作性提高(如：Tekla Structures 對於 Autodesk Revit 的支援，可謂事一強以攻眾弱的連橫政策)，但多家公司以聯盟的方式來共同協商整個 BIM 軟體產業鏈的相互操作性，似乎可能性又低了些，故在同一個工程專案中採用不同供應商之軟體還是有其不小的相互操作性風險。

另一個臺灣知名的半導體公司可作為借鏡，該公司因為專業領域不同的考量，即採用不同的 BIM 設計工具，如：管路模型採用 Bentley 之 OpenPlant、建築模型採用 Autodesk Revit Architecture、機電模型採用 Autodesk Revit MEP、結構模型採用 Tekla Structures、設備機台模型採用 Autodesk Inventor，因本人曾間接接觸該公司在整合這些模型時所遇到之問題，故採用太多不同家公司產品的結果可能是整合上的一大難題。

2.6.3 公開標準之相互操作性

就完全的相互操作性而言，訂定其公開標準是一種追求完美的理想，就純粹排列組合的角度來看確有其道理，若有 n 種不同檔案格式或平台需要直接相互轉換或溝通會需要 n(n-1)種轉換機制，而若採單一公開標準的理想狀況，則只需要

2n 種轉換機制，IFC 格式便是基於這種前提的產物，且因 BIM 公開標準在全生命週期的應用確有其必要性(如：模型整合和備份之需求)，其已儼然成為 BIM 公開標準格式的代名詞。

然而，BIM 設計工具皆有其原生(native)的檔案格式，目前在主流市場中亦尚無基於 IFC 格式的 BIM 設計工具，且 IFC 格式對於各種 BIM 設計和非設計工具之間的相互操作性一直以來都有許多相關的學術研究，可推論 IFC 格式的相互操作性仍有更加完善的空間，故接著將藉由幾篇文獻來回顧一些 IFC 格式在相互操作性上的問題，但必須特別強調的是：具有相互操作性的問題不代表本研究即認為 IFC 格式並非一個好的公開標準或沒有存在的必要性。

諸如 IFC、gbXML 和 COBie 等公開標準雖對於資料交換有一定的貢獻，但也不是萬靈丹，各有各的限制，例如：Abanda et al. (2013)提出三個上述公開標準的限制：第一，語言規格本身的限制，因公開標準需兼顧大眾需求，故不可能基於特定領域的(domain-specific)語意學(semantics)來量身打造，這使得某些特殊需求將無法描述，例如：IFC 在物件外型上對於複雜曲面的支援度較低、gbXML 所產生的 HVAC 資訊於其他軟體之相容性較低，且 gbXML 和 IFC 在幾何資訊及其相互關係的描述也不盡相同、COBie 對於某些模型元件無法正確地轉譯；第二，描述 3D (three-dimensional)幾何資訊的描述方式和交換格式太多了，3D 資訊科技發展良久，在不同的產業有不同的 3D 檔案格式，幾何資訊是視覺化溝通的基礎，若要將不同格式進行轉檔便有資料遺失(data loss)或發生錯誤的可能性，但重新建模又很沒有效率；第三，工程專案的利害關係人(stakeholder)來自不同背景和領域，所以對於相同資訊的語意表達方式便會不同，但實際上可能是在描述同一個事物。

上述第二和第三個限制或可以這樣理解：IFC 在幾何資訊的資料綱要可被各自表述，即同一套法則可依自我需求來解釋，只要可以解釋得通且外觀相符便可接受，例如：Cheng and Wu (2013)開發 BIM Object Adapter 以期在異質的 BIM 系統

之間交換和共享參數化的 BIM 物件，在該研究的文獻回顧中提到許多以 IFC 格式進行資訊交換時所發生資料遺失的情形，即便這些異質的 BIM 系統皆宣稱相容於 IFC 這樣的公開標準。上述現象的根本原因在於：即使是同一個 IFC 檔案，當其被修改時乃依附於所使用之 BIM 設計工具，而不同的 BIM 設計工具有其專屬、特定之規格，尤其是模型元件之非幾何屬性在語意(semantic)上之定義或解讀不同，於是在 IFC 檔案匯出/匯入的過程中造成資料遺失(Monteiro and Martins, 2012)。

不同的 BIM 系統仍可能以不同的方式來描述同一個物件，即便其皆符合 IFC 標準，也就是說以同一個資料綱要來各自表述同一個物件，這種情形好比看同一事物的觀點不同，所導致的陳述內容也會不同，由此可見，公開標準不一定是客觀標準，好比梁構件的樓層解讀方式在建築業和土木業是不一樣的，前者所謂「二樓的梁」和後者所稱「3F 梁」，其實是指同一個梁構件，這種情況又好似英式英語和美式英語的分別，大部分時候一致性很高，但有時卻有完全相反之別，故整個 AECO (Architecture, Engineering, Construction, and Operations)產業鏈不可能有相同的語意描述方式，如同全世界的人不可能講同一種語言。

另一個 BIM 公開標準需要關注的是模型視圖(或稱子模型)的資訊共享和管理，因此 buildingSMART 亦大力推廣 IDM (Information Delivery Manual)和 MVD (Model View Definition)。事實上，早在 2006 年即有學者提出 useful minimum 的概念(Hietanen and Lehtinen, 2006)，該研究認為應找出基於 IFC 標準的資料交換最小範圍(minimum scope)，並且這些資料必須是有用的、有需求的，如此才能確保相互操作性的可預測性和品質，而非一口氣將解決龐大資料交換之需求寄託於 IFC 上，在該研究的結論也提到一個現象：領域專家(domain expert)通常會假設資料交換的範圍(scope)即其所屬領域所創造出來或需要的資料，但理想的資料交換唯有透過領域之間的協同作業才可達至。

綜合以上所述，以本研究的觀點出發，就資料交換的角度而言，真正有用的或有需要的資訊才從 BIM 模型中擷取出來進行交換，並且資訊回饋至 BIM 模型也應如此，IFC 的相互操作性並非放諸四海皆準，尤其各領域專家有其天生不可避免的本位主義(在佛法中稱所知障)，若一味期待一套公開標準能完全符合且能充分解釋各領域的專業知識，便有點過度理想化且很難不落入一言堂的圈套。

適逢美國綜合營造公會的 BIMForum 工作小組在 2013 年 8 月公布新的 Level of Development Specification (BIMForum, 2013)，其中特別解釋 Level of Detail 與 Level of Development 之差異：Level of Detail 指的是模型元件的細節程度(即包含了多少細節)，因此是屬於模型元件的輸入資訊；而 Level of Development 指的是模型元件中的幾何與屬性資料可被信賴之程度，因此是屬於可靠的模型元件輸出資訊(謝尚賢, 2013)。本人認為這番解釋恰好可與上述 useful minimum 之概念相呼應，故可將 LOD 之概念導入 useful minimum 之概念圖，並輔以經濟學之供需理論來類比之，即 useful minimum 可視為 Level of Detail 之資訊輸入供給和 Level of Development 之資訊輸出需求所協調之供需平衡結果，本研究修改如圖 1 所示。

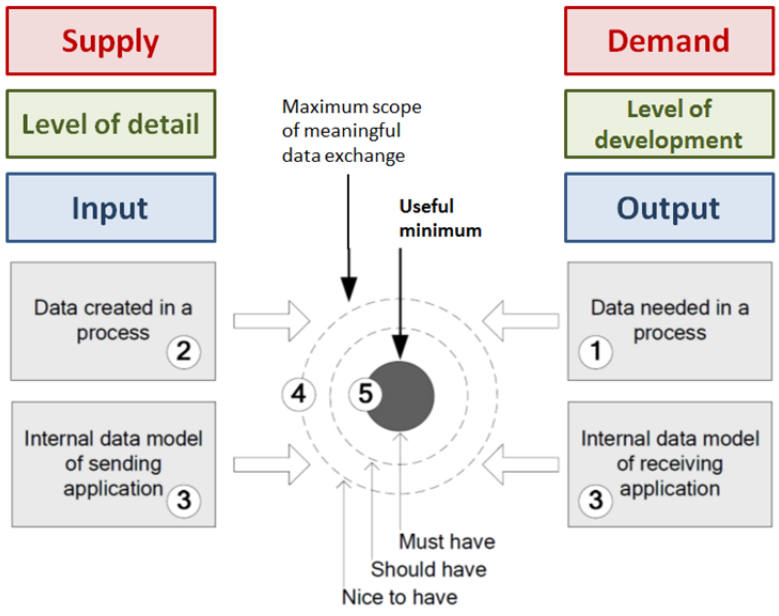


圖 1：以供需觀點來解釋 LOD 及 useful minimum (Hietanen and Lehtinen, 2006)

總結，本研究以三個面向來看待 IFC 所受的限制：市場因素、歷史借鏡和根本原因。就市場因素而言，軟體供應商對於公開標準的態度有時會自我矛盾，增加 IFC 的相互操作性有助於發展 BIM 相關產業，但卻又想將自家產品於全生命週期中布局；就歷史借鏡而言，並沒有任何公開標準是能完全相容於所有的設計工具，舉凡文書處理、影像編輯、多媒體剪輯等應用程式，似乎不同設計工具之間的完全相互操作性並不存在；就根本原因而言，公開標準必須兼顧整個產業鏈中所有參與者之需求，故必然是一種取交集和求共識的過程，一定會有少部分的相容性需求被捨棄，例如：以現實主義來衡量，當利益有交集時可各取所需(如：共同採用 IFC 格式)，而當利益有衝突時則各自為政(如：能源分析者選用 gbXML、設備管理者選用 COBie)。

本研究認為「時效性要求低」和「需求一致性低」使得 AECO 產業的公開標準難以訂定，根據 Kiviniemi (2006)的演講，推廣 IFC 的預算一直都缺乏，同時因產業鏈尚未完全整合而過於分散，生命週期太長以至於沒有主導者(process owner)，故沒有人願意當白老鼠。Kiviniemi 對於 IFC 的幾點建議本研究亦相當認同，並以括弧( )之方式為其加上註腳：

(1) 先使用現在已經有的(資料綱要)，專注於實施和使用(IFC)的品質，(而非一味追求柏拉圖式的理想)。

(2) 別認為 IFC 對於建模來說是不可或缺的，不需一次改變太多，先從 useful minimum 開始共享資料，(眼光要放遠，腳步要放慢)。

(3) IFC 並非終端使用者(end-user)的(而是生命週期)產品，強調錯綜複雜的技術議題是錯誤的訊息，(故模型版次的管控亦很重要)。

(4) 技術面是簡單的，人的因素是更加困難的，(所以 IFC 的相互操作性並非完全是技術問題，有時則是偽命題，因其屬於哲學問題)。

2.7 資訊管理之應用架構

從摘要到 1.1 節之研究緣起，本研究不斷強調「資訊管理」對於 BIM 技術的重要性，即建築物全生命週期中所累積之巨量資料應如何管理，BIM 三個英文字母中，I 最足以代表 BIM 技術之精神。

本研究以「資訊回饋至 BIM 模型」為目標，就資料交換的基本原理而言，1.2.1 節所提及之方法有一些限制或門檻，若拉高到相互操作性的層級而言，2.6 節所介紹的解決方案似乎都不甚完美，也就是說，在無論「亡羊補牢」的資料交換方法或「防患未然」的相互操作性解決方案中，皆無單一最佳的方法或解決方案能讓「資訊回饋至 BIM 模型」的過程中普遍不會遇到瓶頸。

在進入 3.1 節闡述本研究之研究方法和解決方案前，本 2.7 節欲先提出一「資訊管理之應用架構」，以資料庫之觀點來分析 BIM 資訊之管理模式。與 BIM 相關的生命週期資訊有兩種管理模式：一種是以檔案為基礎(file-based)的管理模式，即目前大部分 BIM 設計工具所採取以個別檔案為主的資料儲存方式；另一種則是資料庫的管理模式，即以物件識別碼來繫結(binding)模型中的物件，並將與物件相關之資訊(多數為非幾何資訊)連同物件識別碼儲存在資料庫中。

當太多非必要資訊儲存在單一模型檔案中時，該檔案的存取和傳輸效率會變差，因此要以哪一種管理模式來儲存生命週期資訊是一個重要課題，應以實際需要的模型資訊始納入模型檔案為原則。另一方面，資料庫比起以檔案為基礎的管理模式也有許多優點，例如：資料共享可集中化管理、資料可保有一致性和完整性、資料可透過權限管理來增加安全性、資料的可分析性、資料的並行(concurrency)處理有利於多方協同作業、資料可備份和版本管控等，Kiviniemi (2006)亦提到以檔案為基礎之資料交換並非可行的解決方案，因模型檔案過大時，小部分的修改便沒有效率，且資料的結構和內容不同使得雙向資料交換難以進行，而資料的所有權(ownership)和修正機制亦無法管控，故 Kiviniemi 建議以模型庫(model

repository)或模型伺服器(model server)的方式來儲存 BIM 模型，如此一來就等同於將以檔案為基礎的管理模式轉化為資料庫的管理模式，此時模型可部分交換，每個軟體僅需要與相關的資料進行溝通，不同的人員對於資料共享有不同的權限，故模型版本和修正之管控可於物件層級來進行。

因此，在 BIM 設計工具尚無法脫離以檔案為基礎的管理模式時，目前已有兩種資料庫型態來輔助管理 BIM 模型和相關資訊，一種為以物件識別碼來繫結模型物件之外部資料庫，其目的為以外部資料庫來管理額外的生命週期資訊；另一種為模型庫或模型伺服器，其目的為以資料庫來管理模型檔案使其更具彈性(如 Kiviniemi 所述)，而本研究將引入第三種資料庫型態。

第三種資料庫型態為以「資訊回饋」為訴求之資料庫管理系統(DBMS, DataBase Management System)，由於 2.2 至 2.4 節所述之應用情境各有不同的運作流程，故本研究引入另一個資料庫來專門管理回饋資訊，可稱之「資訊回饋資料庫管理系統」(以下簡稱本資料庫)，本資料庫將作為回饋資訊和 BIM 模型的中介，不管何種應用情境，回饋資訊將不會直接回饋至 BIM 模型，而是經由本資料庫間接地來修改或更新 BIM 模型，本資料庫等同於扮演守門員的角色，如此規劃和設想的原因是為了兼顧各種應用情境的需求，因為各種應用情境所適用的資料交換途徑或有很大的差異，例如：可能是外部資料庫對本資料庫、檔案對本資料庫、應用軟體對本資料庫或人員操作對本資料庫。

本資料庫一方面將直接與以檔案為基礎之 BIM 模型介接，另一方面則概括承受各種不同的資料交換途徑或方法，可有效管理資訊的交換和進出 BIM 模型，進而在資訊回饋至 BIM 模型時有較好的相互操作性，原因有二：本資料庫之資料綱要可根據不同的 BIM 設計工具來設計、本資料庫所儲存和管理之資訊以資訊請求端之最小需求(即 useful minimum)為主要考量。

圖 2 描繪本資料庫之運作方式，以及本節所提到三種不同型態資料庫的連接示意圖，整體架構所形成之概念即為本節之標題：「資訊管理之應用架構」。

在圖 2 中，實心箭頭代表資料交換的方向，即資料流之方向，每一個箭頭皆代表一種資料交換機制，而虛線則代表以物件識別碼作為模型物件繫結。在此應用架構中，居中的為以檔案為基礎之 BIM 模型，以及與其介接之資訊回饋資料庫管理系統，連接此二者的兩個箭頭即代表資訊擷取和資訊回饋之交換機制。

在圖 2 中居右的為各種模型庫或模型伺服器，例如：基於 Autodesk Revit 軟體之模型庫或基於 IFC 之模型庫，由於 Autodesk Revit 之模型庫屬於同步(synchronous)操作，故以單一個雙向箭頭來表示，又 IFC 模型庫主要以版本備份為主，故以一個單向箭頭來表示。

在圖 2 中居左的為各種欲回饋至 BIM 模型檔案之途徑或方法，例如：感測器資料之資料庫管理系統，其記錄設備運作和維修日誌，故以物件識別碼和 BIM 模型檔案中的物件作繫結(虛線示之)，當此外部資料庫有資訊欲回饋至 BIM 模型時(如：更換零件尺寸)，則透過資訊回饋資料庫管理系統間接地修改或更新 BIM 模型。此外，同樣居左的還有可能是各種檔案(如：IFC 或 COBie)或應用軟體(如：分析工具)，這些檔案或應用軟體可直接經由 BIM 模型檔案將所需資訊輸出，惟當這些檔案或應用軟體欲將資訊回饋至 BIM 模型檔案時，必須間接地透過資訊回饋資料庫管理系統，此為本研究特地規劃的把關和緩衝機制，有關資訊回饋資料庫管理系統之規畫和設計將於 3.4 節和 4.2 節再論。

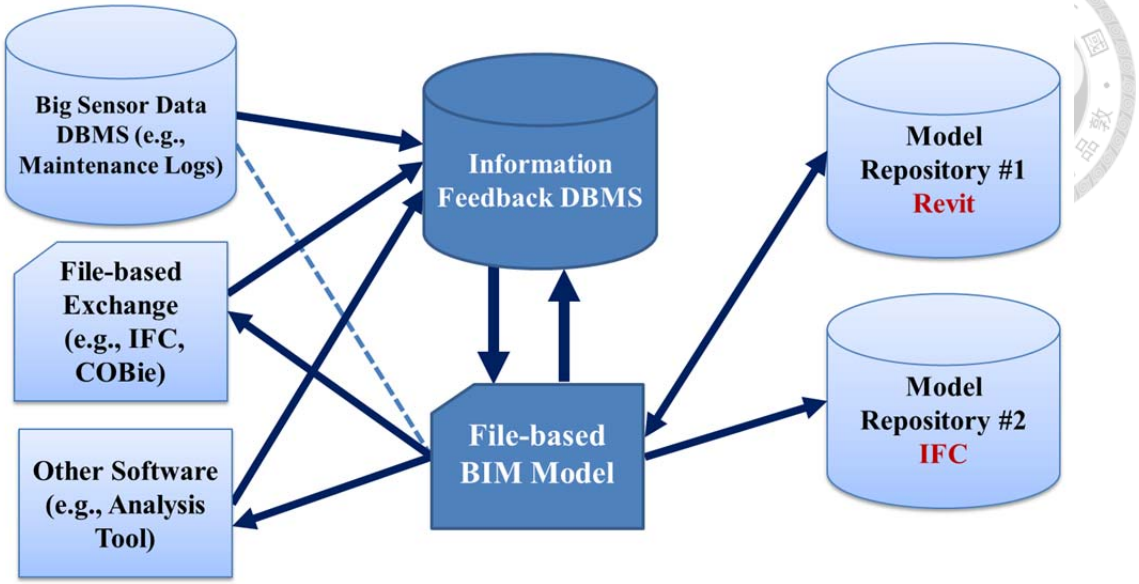


圖 2：資訊回饋資料庫管理系統運作示意圖(即資訊管理之應用架構)

第三章　系統分析

3.1　應用情境導向之資訊管理框架

本研究於圖 2 提出一「資訊管理之應用架構」，其整合以檔案為基礎之管理模式和資料庫之管理模式來管理 BIM 生命週期資訊，與本研究目的最相關的為在圖 2 居中的資訊回饋資料庫管理系統，而和此資訊回饋資料庫管理系統有所互動的 BIM 模型檔案和各種資訊回饋途徑和方法則同樣屬於本研究之範疇，至於在圖 2 居右的模型庫部分則不在本研究之範疇。

圖 2 所呈現的是一種整體概念的應用架構，接下來則要針對資訊回饋至 BIM 模型的各種應用情境(詳如 2.2 至 2.4 節)來量身訂做一個「應用情境導向之資訊管理框架」，即更詳細地以資訊技術原理來發展資訊回饋至 BIM 模型之方法論，故在此框架所涵蓋的範圍主要針對在圖 2 中所居左和居中的部分。

在圖 2 居中的包含資訊回饋資料庫管理系統，由於在 1.2.1 節已說明資訊回饋前往往先有資訊擷取，故此資料庫系統主要負責儲存和傳送欲修改或更新之 BIM 模型中，相關聯之基本物件資訊(如：物件識別碼、參數化屬性識別碼、參數化屬性值等)，並且儲存和接收從各種途徑或方法所回饋之資訊。

然而，考量各種應用情境皆有透過網路傳輸資訊或跨平台操作之需求，並非每一種資料庫系統皆具備多人協同作業之管理介面，即便是有，就安全性和權限管控的角度而言，讓資訊回饋端的工程專案參與者直接管理資料庫並不妥當，何況資訊回饋端的使用者也不一定就熟悉資料庫的管理方法，若以呈現模型資訊為考量，開發一個以網頁為基礎(web-based)之視覺化介面或許較適合，並可針對不同的應用情境來客製化不同的網頁使用者介面，對於資訊回饋端的使用者來說也更容易上手，因此架設至少一個模型呈現網頁伺服器(web server)來作為資訊回饋端之使用者介面有其必要，此模型呈現網頁伺服器必須兼備網路應用程式和資料庫應用程式，網路應用程式可讓資訊回饋端使用者來查看模型基本資訊(如：專案

資訊或一部分的模型的視覺化呈現)，而資料庫應用程式則負責和資訊回饋資料庫管理系統進行資訊交換，就呈現模型資訊之需求而言，模型呈現網頁伺服器和資訊回饋資料庫系統之間則以物件識別碼來進行物件對應(object mapping)，如此一來便不必一次性地讀取所有的模型資訊，而是有需要時再透過物件識別碼向資訊回饋資料庫系統進行查詢(query)。

在圖 2 居中的也包含 BIM 模型檔案，由於目前 BIM 設計工具多是以檔案為基礎之管理模式，回顧在 1.2.1 節、2.6.2 節和 2.6.3 節所述，相互操作性是受 BIM 設計工具所限制的，而欲解析特定 BIM 設計工具之原生檔案格式又不太可能，IFC 格式又存在不少相互操作性問題，故 BIM 模型檔案之格式應採用 BIM 設計工具之原生檔案格式，至於解析該模型檔案的方式則可透過 BIM 設計工具之 API，這隱含著兩個事實：其一，沒有提供 API 的 BIM 設計工具無法適用於本框架，但這種情況在 BIM 設計工具的產品市場來說並不常見，因目前 BIM 的商業化作業流程尚未全部確立，故提供 API 是增加 BIM 設計工具擴充性的最佳辦法；其二，因無法直接解析 BIM 設計工具之原生檔案格式，故資訊回饋的流程必須使用 BIM 設計工具，即無法在不執行 BIM 設計工具的情況下修改或更新某個 BIM 模型檔案，不過既然 BIM 設計工具絕對有其存在的必要性，執行它應屬無可厚非。

在圖 2 居左的為各種資訊回饋至 BIM 模型檔案之途徑或方法(即資訊回饋端)，由於資訊回饋資料庫系統為本框架下之回饋資訊接收者(以下簡稱本接收者)，故可能是檔案對本接收者、應用軟體對本接受者、資料庫對本接受者或人員操作對本接受者。若為檔案或人員操作對本接受者，則恰好可基於上述模型呈現網頁伺服器來設計整合性的使用者介面，即此網頁使用者介面應包含呈現模型資訊之操作介面和回饋資訊之操作介面，其中回饋資訊之操作介面可讓人員直接透過網頁介面來修改或更新物件資訊，抑或透過檔案上傳之方式讓網頁伺服器來處理具有回饋資訊之上傳檔案。若為軟體或資料庫對本接收者，則較難直接基於模型呈現網

頁伺服器來整合資訊回饋之操作介面，除非將呈現模型之網頁操作介面嵌入(embed)軟體之外掛程式或資料庫應用程式中，因軟體或資料庫中欲回饋之資訊仍要經由本接收者，故較適合的方式為根據本接受者之資料庫型態來開發 web 服務(web service)之 API，例如：開發一套基於 HTTP (HyperText Transfer Protocol)通訊協定的 API (採用此方法之原因待 4.3 節再說明)，則此套可和本接收者進行資訊交換之 API 既可提供給資訊回饋端之應用軟體或資料庫使用，亦可提供給資訊請求端之 BIM 設計工具使用。圖 3 為本「應用情境導向之資訊管理框架」示意圖。

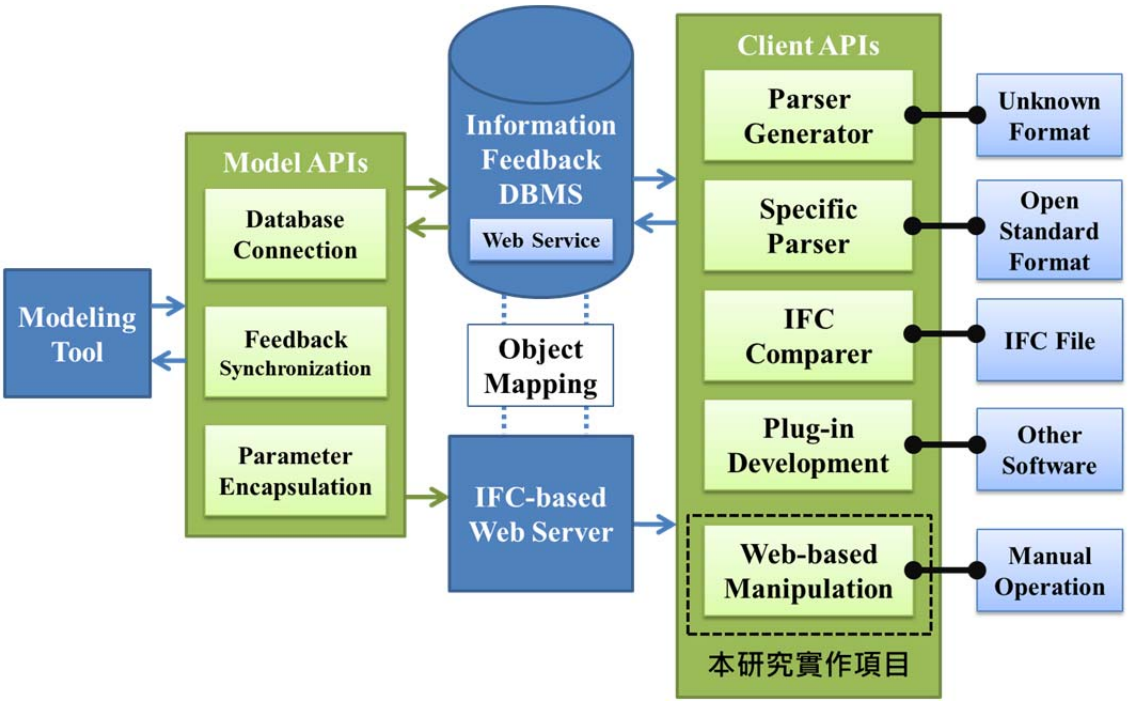


圖 3：應用情境導向之資訊管理框架

在圖 3 中，箭頭代表資訊流方向，虛線代表物件對應，雙圓頭直線則代表客戶端應用程式介面所綁定之資訊回饋途徑或方法，有關各構成要素(component)的資訊交換原理，已根據各種應用情境之普遍適用原則在此框架中被實現，惟具體採用的技術或工具仍保有彈性和轉換空間，以下針對各構成要素逐項說明：

(1) 建模工具(Modeling Tool)

自本節後，BIM 設計工具將以建模工具來代表之，因建模工具為最基本的建置 BIM 模型之軟體，即在本研究中資訊回饋之目的地，模型管理者將可透過建模工具之外掛程式作為資訊請求端。

(2) 模型端應用程式介面(Model APIs)

模型端應用程式介面為整合建模工具 API 的一套類別庫(class library)，其至少必須具備三個功能模組：資料庫連結模組、回饋同步模組和參數封裝模組，惟實務上進行系統開發時的命名空間(namespace)不一定要按照此分類法，只要能達到這三個功能模組的需求即可。顧名思義，資料庫連結模組負責建模工具和資訊回饋資料管理系統之間的資料交換(詳如 4.3 節)，即負責網路資料傳輸部分；回饋同步模組負責將回饋資訊整合，以待模型管理者確認並同步修改或更新模型；參數封裝模組負責將模型資訊封裝為 IFC 格式之檔案並上傳至基於 IFC 之網頁伺服器，或是封裝為適用之資料結構(詳如 4.2 節)並傳送至資訊回饋資料庫管理系統。

(3) 資訊回饋資料庫管理系統(Information Feedback DBMS)

資訊回饋資料庫管理系統包含至少一個可透過網路傳輸資料的資料庫系統，以及一套適用於該資料庫型態的 web 服務，web 服務可作為模型應用程式介面和客戶端應用程式介面的資料傳輸介面，亦可作為資訊回饋資料庫管理系統之防火牆(firewall)。必須強調的是：資訊回饋資料庫的設計原則是以建模工具(即資訊請求端)的最小需求為考量，即資訊回饋資料庫的資料綱要可根據不同建模工具之資料模型(data model)來設計，並且資訊回饋資料庫所儲存之資訊也取決於由建模工具所建置的 BIM 模型，當此 BIM 模型有資訊回饋需求時，才讓其相關的物件資訊透過模型應用程式介面傳送至資訊回饋資料庫系統。

(4) 基於 IFC 之網頁伺服器(IFC-based Web Server)

雖然在 2.6.3 節提及不少 IFC 格式的相互操作性問題，但 IFC 格式仍是目前 BIM 軟體的公開標準格式，故幾乎所有的建模工具皆支援將模型輸出為 IFC 格式，故模型呈現網頁伺服器可設計為基於 IFC 之網頁伺服器，如此一來，至少模型幾何資訊的擷取可透過 IFC 格式來輸出，同時透過物件識別碼和資訊回饋資料庫管理系統的物件作對應。簡單來說，基於 IFC 之網頁伺服器主要工作為在網頁介面上將 BIM 模型視覺化以供資訊回饋端參考(無論以何種型式)，就像一般文件輸出成 PDF (Portable Document Format)格式供他人參考一樣，而 IFC 檔案大小通常較大，因資訊回饋資料庫系統要求高效能(詳如 3.4 節)，故不適合直接傳送至資訊回饋資料庫系統儲存，另一個做法是將 IFC 檔案上傳至模型庫，再讓基於 IFC 的網頁伺服器向 IFC 模型庫進行查詢，惟模型庫非本研究之範疇。

(5) 客戶端應用程式介面(Client APIs)

客戶端應用程式介面主要目的為解析各種途徑或方法之回饋資訊，並將解析後之回饋資訊傳送至資訊回饋資料庫系統，即必須具備和資訊回饋資料庫管理系統中的 web 服務溝通之能力，而因應各種資訊回饋途徑或方法之不同，客戶端應用程式介面不一定是類別庫，也可將客戶端應用程式介面設計成 web 服務或網頁伺服器，這部分保留較大的彈性，因為每一種資訊回饋之途徑或方法皆必須有相應之客戶端應用程式介面，只要客戶端應用程式介面能接收和解析其回饋資訊，並且能將解析後回饋資訊透過資訊回饋資料庫系統之 web 服務傳回即可，圖 3 右側列出幾個可能的回饋途徑或方法，其客戶端應用程式介面以各種模組來解釋必須具備的功能，惟實務上開發時不一定要以此模組來作為命名空間，茲分別說明如下：

I. 語法分析器產生器模組(parser generator)

有些回饋資訊之檔案格式並非公開標準格式，故必須先利用語法分析器之產生器來建構其文法(grammar)，此文法必須由資訊回饋端負責定義，待適用的語法分析器產生後再進行語法分析(parsing)，另一種方式則是資訊回饋端自行提供語法分析器。此外，資訊回饋端必須提供一套建模工具物件識別碼和參數化屬性識別碼之對應規則。

II. 特定語法分析器模組(specific parser)

特定語法分析器專門處理公開標準格式的檔案(如：COBie、gbXML等)，由於公開標準格式的資料綱要事先已經知道，因此可直接將其檔案進行語意分析，並將欲回饋之資訊以 web 服務方式傳回資訊回饋資料庫管理系統。此外，資訊回饋端必須提供在公開標準格式的資料綱要中，建模工具物件識別碼和參數化屬性識別碼之對應規則。

III. IFC 比較器模組(IFC comparer)

由於 IFC 格式為普遍適用的 BIM 公開標準格式，若回饋資訊是以 IFC 格式存在(如：原有之 IFC 檔案經過某種分析後產生新的 IFC 檔案)，即可直接與原先之 IFC 檔案進行比對(即存放在基於 IFC 之網頁伺服器上之 IFC 模型)，然後將有差異的部分回饋即可。此外，資訊回饋端必須提供在 IFC 格式的資料綱要中，建模工具物件識別碼和參數化屬性識別碼之對應規則。

IV. 外掛開發模組(plug-in development)

外掛開發模組則主要針對應用軟體或資料庫的資訊回饋模式，因其或需要開發一個專屬的外掛程式或資料庫應用程式，故此時外掛開發模組適合以類別庫的型式存在，此類別庫可和資訊回饋資料庫系統之 web 服務介接，同時可接受由應用軟體或資料庫透過其 API 所輸出之

回饋資訊，而透過應用軟體或資料庫之 API 所輸出之回饋資訊必須由資訊回饋端指定一套建模工具物件識別碼和參數化屬性識別碼之對應規則。

V. 網頁操作模組(web-based manipulation)

網頁操作模組提供一網頁使用者介面讓資訊回饋端使用者操作以回饋資訊，可自行客製化較符合應用情境之使用者介面，惟此模組主要以人員操作為目的，故較不適合批次處理。此外，建模工具物件識別碼和參數化屬性識別碼可直接以其人類可讀(human-readable)名稱在網頁上呈現，以利使用者參考並將資訊藉由手動輸入來回饋。

上述 I、II、III 及 V 模組可連同基於 IFC 之網頁伺服器整合在一起，抑或各自發展為 web 服務或獨立的網頁伺服器，至於 IV 模組之類別庫主要用於開發應用軟體之外掛程式或開發資料庫之資料庫應用程式。

在圖 3 應用情境導向之資訊管理框架中，無論使用何種建模工具、何種資訊回饋途徑或方法(檔案、應用軟體、資料庫或人員操作)，透過資訊回饋資料庫系統的中介皆可讓資訊回饋至由建模工具所建置之 BIM 模型，資訊請求端之模型管理者可透過模型應用程式介面將模型基本資訊發佈至基於 IFC 之網頁伺服器，以及將相關資訊請求發佈至資訊回饋資料庫管理系統，另一方面，資訊回饋端之使用者便可查看和參考，並透過客戶端應用程式介面將回饋資訊事先解析，解析完畢後再傳回至資訊回饋資料庫管理系統，待資訊請求端之模型管理者接受即可修改或更新 BIM 模型，詳細運作機制和操作邏輯待 3.3 節敘述。

整體而言，應用情境導向的資訊管理框架主要以資訊回饋資料庫管理系統來解決相互操作性的問題，因資訊回饋資料庫管理系統直接透過模型端應用程式介面來介接建模工具，並且資訊回饋資料庫的資料綱要可根據不同建模工具之資料模型來設計，當有資訊欲回饋至由建模工具所建置之 BIM 模型時，資訊請求端必

須先提供模型基本資訊和一部分的物件資訊，而資訊回饋端必須透過客戶端應用程式介面來解析回饋資訊以符合資訊回饋資料庫之資料綱要，這樣好比是建模工具提供一個篩子，無論回饋資訊是以何種型式存在，只有能通過這個篩子的資料才能被傳送回資訊回饋資料庫，而這個篩子本來就是根據建模工具之資料模型所設計的，因此建模工具可將回饋資訊和 BIM 模型無縫接軌，這和以往直接將含有回饋資訊的整個檔案交由建模工具進行解析是完全不同的。

3.2 產品定位與規劃

本研究目的為讓資訊能回饋至 BIM 模型以將其修改或更新，為使各種應用情境皆能普遍適用，本研究先於 2.7 節提出一「資訊管理之應用架構」，將資料庫之管理模式導入以檔案為基礎之管理模式，並整合二者，再根據此應用架構再於 3.1 節提出一「應用情境導向之管理框架」，在此框架中明確定義各構成要素之功能需求及運作方式，且引入客戶端應用程式介面的概念，即各種資訊回饋之途徑或方法皆應有其適用的客戶端應用程式介面。

然而，「應用情境導向之管理框架」是一個兼顧各種應用情境需求的完整藍圖，就本論文研究之範疇而言，欲將整個框架中的每個構成要素都實作出來仍顯吃重，雖謂法門無量誓願學，但也必須先從一門深入，即本研究應將系統開發收斂在合理範圍，因此本節將決定實作此框架中哪些部分。

本研究欲開發的系統可視為本研究之「產品」，由於在圖 3 框架中資訊回饋端的差異較大，故此產品之定位將取決於客戶端應用程式介面的涵蓋範圍，就圖 3 所示的五種客戶端應用程式介面而言，不同模組代表客戶端應用程式介面所綁定的資訊回饋途徑或方法亦不同。

若以檔案為資訊回饋之途徑或方法(如：語法分析器產生器、特定語法分析器和 IFC 比較器)，客戶端應用程式介面的主要工作為解析檔案中之回饋資訊，故針對各種資料模型進行語法分析是必要的研究重點，惟此部分的研究性質較繁瑣，

且較無法凸顯資訊請求端和資訊回饋端之間的互動。另一方面，BIM 相關的檔案格式語法分析已有許多不同領域的學術研究，相關的演算法(algorithm)或轉換機制可謂五花八門，甚至圖 3 中的 IFC 比較器亦已有研究在開放原始碼的專案中發表(Beetz et al., 2010)，因此本研究將重心放在較具原創性的層面上。

若欲進行外掛程式或資料庫應用程式的開發，則必須選擇特定的應用軟體或資料庫來作為資訊回饋端，此時較難普遍適用於各種應用情境，且特定的應用軟體或資料庫欲將資訊回饋至資訊回饋資料庫系統必須經由其 API，此部分的歧異度則更大，因不同的應用軟體或資料庫也需要不同的客戶端應用程式介面。

因此，本研究選擇實作圖 3 中的網頁操作模組，此模組直接提供一網頁使用者介面讓人員操作，使用者可直接在此操作介面上將其所欲回饋之資訊回傳至資訊回饋資料庫系統，且此種回饋途徑可設計成 B/S (Browser/Server)架構，因此可跨平台、跨作業系統、跨裝置操作，較能在多種應用情境下使用。

至此，本研究之系統開發範圍已確立，即包含圖 3 中之模型端應用程式介面(包含建模工具外掛程式)、具 web 服務之資訊回饋資料庫系統、基於 IFC 之網頁伺服器以及資訊回饋端之網頁操作模組，本研究稱此系統為「可互操作型物件式資訊回饋系統」，詳細系統架構及其示意圖將在 4.1 節說明。為了專注於開發系統核心功能和減少不必要的資源排擠，本系統在設計層面有三項規劃：

(1) 基於 IFC 之網頁伺服器旨在讓資訊請求端提供基本的模型資訊及以視覺化呈現模型幾何資訊，以作為資訊回饋端之參考。考量在網頁操作介面上呈現三維幾何模型的效能尚未完善，故本研究僅在設計時期實作一個原型(prototype)系統作為示範(詳如 4.6 節)，實際的系統開發實作將以截圖(screenshot)之方式來呈現模型的三維幾何資訊。

(2) 如同 3.1 節所述，網頁操作模組可和基於 IFC 之網頁伺服器一同開發，為了設備管理的單純化，故本研究僅利用一個網頁伺服器來開發三維模型呈現之網頁介面及資訊回饋端之網頁操作模組。

(3) 本研究所開發之系統稱為物件式(object-based)而不稱物件導向(object-oriented 或 class-based)之原因為：本研究主要目的為進行 BIM 模型的修改或更新，並非建置一個全新的 BIM 模型，即並非無中生有的過程，故資訊回饋資料庫系統僅儲存實際需要的物件資訊，這些物件資訊源自於模型中的實作元件(即 instance)，且物件資訊應當包含實作元件之物件識別碼，故當回饋資訊經由模型應用程式介面回饋至 BIM 模型時，僅需要透過物件識別碼來識別不同的實作元件，故資訊回饋資料庫系統並不需要記錄物件的繼承和多型關係，可大幅降低資料結構的設計難度，同時資料庫類型的選擇亦能較有彈性(詳如 3.4 節)。

3.3 運作機制與操作邏輯分析

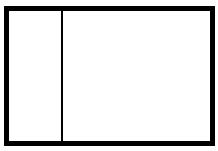
本節先藉由業務流程建模標記法(BPMN, Business Process Model and Notation) (White, 2008)來解釋欲開發系統之運作機制和操作邏輯，接著再針對流程中的任務(task)、事件(event)和閘道(gateway)逐個分析其背後的設計原理或考量，由於本研究實際應用到的 BPMN 標記種類並不多，故將有用到的 BPMN 標記符號以表列方式呈現如表 1。

由 buildingSMART 所主導的 National BIM Standard 採用 BPMN 來作為流程建模(process modeling)的主要表達方式，在其資訊交付手冊(IDM, Information Delivery Manual)中的流程圖(process map)亦建議以 BPMN 來表達整個流程進行和資訊交換之機制，目前亦有不少學術研究將 BIM 的協同作業流程以 BPMN 來表達 (Eastman et al., 2009; 蔡孟君, 2013)，惟 BIM 作業流程有時其實用不到那麼多標記符號，甚至有些標記方式太過複雜(Park et al., 2011)，故本研究欲開發系統之運作

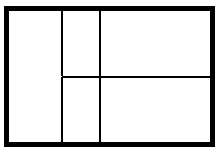
流程僅依照 BPMN 的大原則來繪製，而不包含 IDM 中流程圖所規範的諸多細節，本研究欲開發之系統名稱為「BIM 模型修改之可互操作物件式資訊回饋系統」(以下簡稱本系統)，其 BPMN 運作流程圖如圖 4 所示。

表 1：本研究使用之 BPMN 標記符號說明

泳道(swimlane)：對活動加以組織或分類的機制。



池(pool)：
單一流程(process)的容器(container)。



道(lane)：
道係在池中，將活動依職能不同分成不同角色、組織或系統。

事件：在流程中發生的事情，(可能)會影響流程。

開始事件：
表示流程由此觸發。

結束事件：
表示流程在此結束。

訊息丟擲(throw)事件：
表示將訊息發送出去。

訊息捕獲(catch)事件：
當收到訊息時啟動流程。

任務：表示單一工作單元，不會再分為更詳盡的流程細節。

一般任務：
表示單一工作單元。

使用者任務：
表示使用者操作軟體或應用程式的任務。

服務任務：
表示使用服務的任務，服務可能是 web 服務、應用程式或其他自動化的軟體工具，在本研究表示使用 web 服務。

閘道：根據不同條件決定流程的分流或合併。

排他(exclusive)閘道：
分流時，僅有一個順序流分支會被執行；合併時，等待僅有一個順序流分支完成。

內含(inclusive)閘道：
分流時，至少會有一個順序流分支被啟動；合併時，等待所有已生效的(active)順序流分支完成。

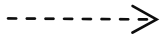
器物(artifacts)：提供流程之外的額外資訊。

資料庫(data store)：
表示流程內、外所存取的資訊。

資料物件(data object)：
表示流程中的檔案或資料。

連線物件(connecting object)：連接流程的物件。

順序流(sequence flow)：
表示活動進行的順序。



關聯(association)：
建立器物或文字至流程物件的關係。

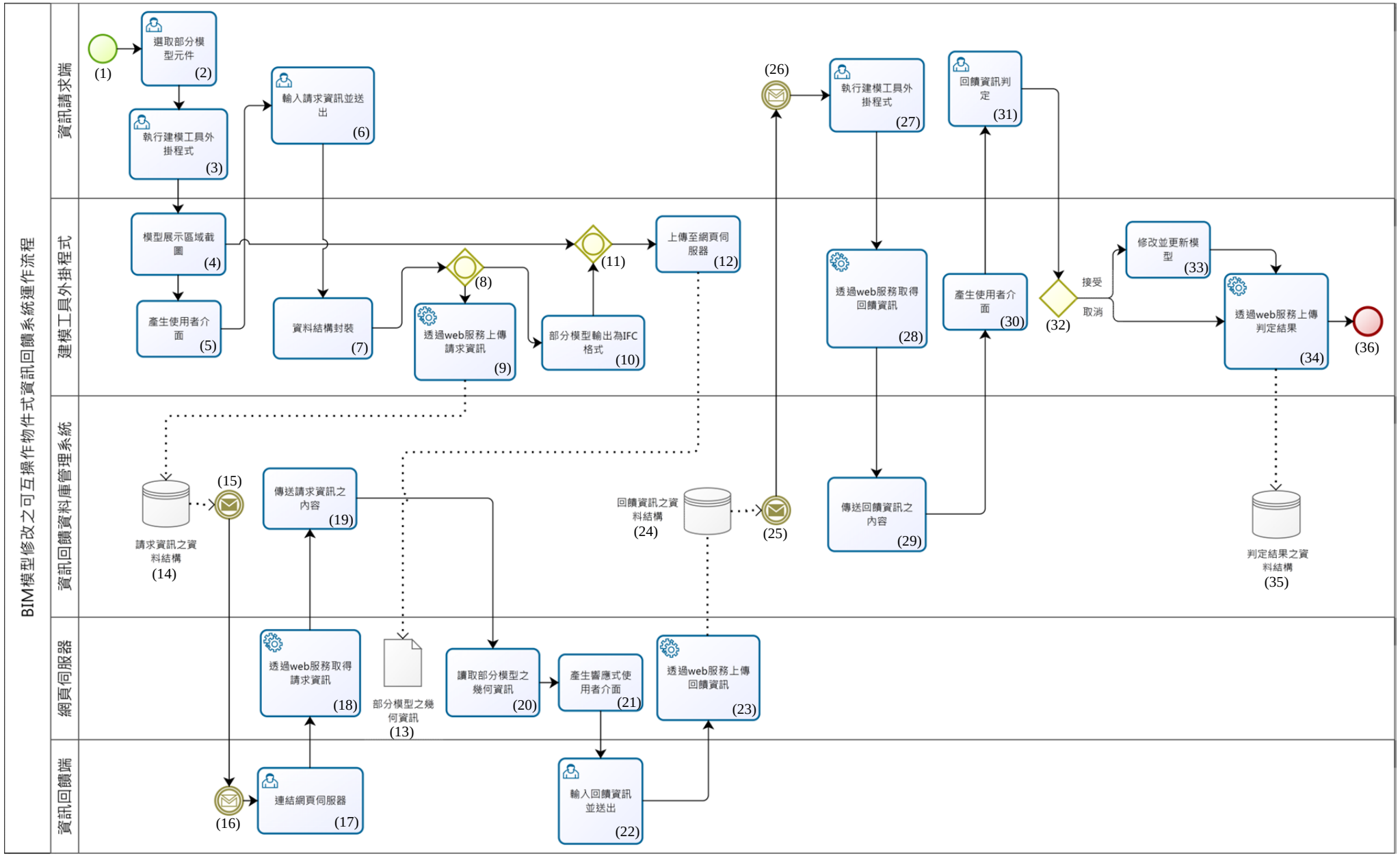


圖 4：BIM 模型修改之可互操作物件式資訊回饋系統運作流程

在圖 4 之運作流程中，泳道中共有一個池和五個道，代表本系統有五個不同的角色或系統，其中資訊請求端和資訊回饋端為操作本系統之兩方使用者，建模工具外掛程式、資訊回饋資料庫管理系統和網頁伺服器則皆為本系統之構成要素，接著以條列式說明各個任務、事件、閘道所代表的步驟：

(1) **開始事件**：即本系統之運作流程由此事件觸發。

(2) **選取部分模型元件**：資訊請求端使用者在建模工具選擇至少一個模型元件，這些模型元件是需要被修改或更新的。

(3) **執行建模工具外掛程式**：資訊請求端使用者在建模工具之使用者介面中執行本系統之建模工具外掛程式。

(4) **模型展示區域截圖**：建模工具外掛程式透過模型端應用程式介面將目前模型展示區域之螢幕畫面截圖。

(5) **產生使用者介面**：建模工具外掛程式透過模型端應用程式介面產生資訊請求之使用者介面。

(6) **輸入請求資訊並送出**：資訊請求端使用者輸入請求資訊並點選送出，請求資訊應包含：所選取模型元件之物件識別碼、所選取之參數化屬性識別碼、參數化屬性之數值單位及專案或模型元件基本資訊等等。

(7) **資料結構封裝**：建模工具外掛程式透過模型端應用程式介面將請求資訊封裝為符合資訊回饋資料庫系統資料綱要之資料結構。

(8) **內含閘道**：表示接下來至少一個任務會被啟動。

(9) **透過 web 服務上傳請求資訊**：建模工具外掛程式透過資訊回饋資料庫系統之 web 服務將資訊請求端使用者之請求資訊傳回資訊回饋資料庫。

(10) **部分模型輸出為 IFC 格式**：有關基於 IFC 之網頁伺服器部分，由於本系統在本研究中只設計一個原型系統，故若未來此功能已趨完善時，建模工具外掛程式可透過模型端應用程式介面將資訊請求端使用者所選取之模型元件輸出為 IFC 格式。

(11) **內含閘道**：表示目前進展中的任務皆完成後將合而為一。

(12) **上傳至網頁伺服器**：建模工具外掛程式透過模型端應用程式介面將模型展示區域截圖和部分模型之 IFC 檔案上傳至網頁伺服器，由於圖片和 IFC 之檔案大小相對於請求資訊來說大很多，為避免影響資訊回饋資料庫系統之資訊處理效率，故將其直接上傳至網頁伺服器。

(13) **部分模型之幾何資訊**：即已上傳之 IFC 格式模型檔案。

(14) **請求資訊之資料結構**：即由資訊回饋資料庫系統所儲存和管理之請求資訊，其資料結構符合資訊回饋資料庫系統之資料綱要。

(15) **丟擲訊息事件**：在請求資訊上傳完畢後，可向資訊回饋端發送一訊息通知，告知其可經由本系統之網頁伺服器查看請求資訊之內容。

(16) **捕獲訊息事件**：資訊回饋端使用者接收到訊息之趨動事件。

(17) **連結網頁伺服器**：資訊回饋端使用者經由執行網頁瀏覽器程式(無論使用何種裝置和平台)以連結本系統之網頁伺服器。

(18) **透過 web 服務取得請求資訊**：網頁伺服器透過資訊回饋資料庫系統之 web 服務取得請求資訊。

(19) **傳送請求資訊之內容**：資訊回饋資料庫系統透過 web 服務向網頁伺服器傳送請求資訊。

(20) **讀取部分模型之幾何資訊**：網頁伺服器將部分模型之幾何資訊(可能為截圖或 IFC 檔案)解析為瀏覽器程式可讀取之資料格式。

(21) **產生響應式使用者介面**：響應式(responsive)使用者介面之概念為根據不同裝置(如：電腦、平板、手機等)之螢幕解析度而產生不同的使用者介面。網頁伺服器整合請求資訊和部分模型之幾何資訊，透過模型端應用程式介面(即其中之網頁操作模組)產生一響應式使用者介面予資訊回饋端使用者，以讓其進行資訊回饋。

(22) **輸入回饋資訊並送出**：資訊回饋端使用者直接經由網頁伺服器所產生之響應式使用者介面來輸入欲回饋之資訊，例如：參考目前模型元件之參數化屬性值和其數值單位，輸入新的參數化屬性值，並執行送出指令。

(23) **透過 web 服務上傳回饋資訊**：網頁伺服器透過資訊回饋資料庫系統之 web 服務將資訊回饋端使用者輸入之回饋資訊回傳至資訊回饋資料庫。

(24) **回饋資訊之資料結構**：即由資訊回饋資料庫系統所儲存和管理之回饋資訊，其資料結構符合資訊回饋資料庫系統之資料綱要。

(25) **丟擲訊息事件**：在回饋資訊上傳完畢後，可向資訊請求端發送一訊息通知，告知其可由本系統之建模工具外掛程式查看回饋資訊之內容。

(26) **捕獲訊息事件**：資訊請求端使用者接收到訊息之趨動事件。

(27) **執行建模工具外掛程式**：資訊請求端使用者在建模工具之使用者介面中執行本系統之建模工具外掛程式。

(28) **透過 web 服務取得回饋資訊**：建模工具外掛程式透過資訊回饋資料庫系統之 web 服務取得資訊回饋端使用者提供之回饋資訊。

(29) **傳送回饋資訊之內容：**資訊回饋資料庫系統透過 web 服務向建模工具外掛程式傳送回饋資訊。

(30) **產生使用者介面：**建模工具外掛程式透過模型端應用程式介面產生檢視回饋資訊之使用者介面。

(31) **回饋資訊判定：**資訊請求端使用者在檢視回饋資訊後可進行判定。

(32) **排他閘道：**代表接下來將只有一個順序流(sequence flow)會被執行。資訊請求端使用者可告知本系統回饋資訊之判定結果。若回饋資訊被接受，則與請求資訊相關聯的模型元件會自動被修改或更新；若回饋資訊被取消，則模型部分不採取任何動作。

(33) **修改並更新模型：**若回饋資訊被資訊請求端使用者接受，則與請求資訊相關聯的模型元件會自動被修改或更新，即建模工具外掛程式會透過模型端應用程式介面將模型元件之參數化屬性進行修改或更新。

(34) **透過 web 服務上傳判定結果：**建模工具外掛程式透過資訊回饋資料庫系統之 web 服務回傳資訊請求端使用者之判定結果，以作為記錄。

(35) **判定結果：**即由資訊回饋資料庫系統所儲存和管理之判定結果，其資料結構符合資訊回饋資料庫系統之資料綱要。

(36) **結束事件：**即本系統之運作流程在此事件結束。

雖然工程專案在生命週期中經常變動，但並非每次變動均牽涉整個 BIM 模型中的所有物件，故釐清模型元件之間的相關性有其必要，由於參數化的模型元件同時具備幾何和非幾何屬性資料，有助於掌控模型元件之間的連動性(Sawhney and Maheswari, 2013)。有鑑於此，本研究在規劃整體運作機制時，除了前提是已經具備一個建置得差不多的 BIM 模型(2.1 節所述)，此模型中的資訊應以 useful minimum

來考量(2.6.3 節所述)，即所謂需要修改或更新的部分模型，可能是一小部分的模型元件、一小部分的參數化屬性、一小部分的幾何位置更動或一小部分的材料更換等等，至於這部分模型和整個模型或其他模型元件之相關性和連動性則必須由模型管理者去判定，因這涉及專業領域的知識，且當部分模型被程式自動修改或更新時，若導致衍生的問題，BIM 設計工具或建模工具也應該會發出警示訊息，因此整個修改和更新的機制應以實際需要的模型元件為主，不須涉及整個模型，以免涉及的範圍或層面太大，輸出的模型資訊或檔案也會太大，從而導致效率降低。

3.4 相容性、延伸性、可擴展性與安全性

產品定位和運作流程皆確定之後，接著必須選擇適合的技術或工具來進行開發，故本節針對所欲開發系統之構成要素進行多面向的剖析，以挑選符合系統需求的技術或工具來設計和實作，而多面向的考量點包含：相容性、延伸性、可擴展性和安全性，以下先針對這四個名詞說明本研究的切入點：

- **相容性(compatibility)**：主要考量不同的 BIM 協同作業者，其使用的硬體裝置、作業系統、BIM 軟體或工具等，所挑選的技術或工具應力求異質系統之間的相容性。
- **延伸性(extensibility)**：主要考量系統開發的生命週期中，若需要將功能性(functionality)延伸或增加新功能，能否有其彈性和更多的發展空間。
- **可擴展性(scalability)**：主要考量系統運行階段的可擴充性，當使用者人數達到預先規畫之上限時，能否將軟、硬體重新配置以容納更多使用者。
- **安全性(security)**：主要考量系統的資料安全性和使用者權限管理，即僅有具權限的使用者才能運用在其權限範圍內的資料。

就資訊技術層面而言，本系統主要包含三種電腦系統：應用軟體之外掛程式、資料庫管理系統和網頁伺服器，接著以上述四個面向來分析三大構成要素的系統開發技術或工具：

(1) 模型端應用程式介面及建模工具外掛程式

模型端應用程式介面是基於建模工具 API 的一套類別庫，其至少必須包含資料庫連結模組、回饋同步模組和參數封裝模組，透過建模工具 API 可將模型資訊從建模工具中讀取或寫入，而上述三個模組則基於建模工具 API 來負責處理本系統之商業邏輯(business logic)，因此模型端應用程式介面可用來開發建模工具外掛程式，此外掛程式主要提供給資訊請求端之使用者。

- **相容性**

  若使用不同的建模工具，由於其商業邏輯和 API 皆不盡相同，故外掛程式在系統開發時也會有所差異。因此，若本系統欲在不同的建模工具上運作，應規劃不同的版本，且在系統設計上可把與建模工具 API 相關的類別(class)獨立出來(如：命名空間可根據不同的建模工具 API 來區分)，則使用者介面和資料處理可較不受建模工具 API 之影響。

- **延伸性**

  雖然本系統之外掛程式在不同建模工具上必須規劃不同版本，但目前市面上主要的建模工具(Autodesk Revit, Graphisoft ArchiCAD, Bentley AECOsim 等)皆可基於 Microsoft .NET Framework 來進行外掛程式的開發，若本系統能盡量以鬆散耦合(loosely coupled)之方式來設計，則與建模工具 API 無關的模組可重複利用和擴充。

- **可擴展性**

  由於目前市面上主要的建模工具仍是以檔案為基礎，若本系統之外掛程式欲提供給多個協同作業者使用，則必須整合專案管理系統。

- **安全性**

  由於建模工具外掛程式必須存在於 local 端，故相關的執行檔、類別庫或外部檔案也必須存放於 local 端，若以保護智慧財產權的角度來看，為避免這些程式或檔案被有心人以還原工程(reverse engineering)或反編譯器(decompiler)來逆向解析，重要的模組或資料需要加密處理。

從四個面向來看，若操作邏輯和資料結構能隨不同建模工具來調整(即開發不同版本)，本系統採用何種建模工具並無太大差別，故本研究考量建模工具的軟體授權可取得性和 API 文件完整性，將採用 Autodesk Revit 來實作本系統，有關建模工具外掛程式的設計細節將在 4.4 節詳述。

(2) 資訊回饋資料庫管理系統

在 1.2.1 節和 1.2.4 節已大致說明使用傳統關聯式資料庫的限制，其主要原因為大量資料寫入的分散式處理效能不佳、資料綱要變更時的操作彈性不足等(Sasaki, 2011)。Nour (2009)曾將不同的 BIM 或 IFC 交換格式以關聯式資料庫來管理，該研究發現單一模型檔案大於 2MB 時，關聯式資料庫的效率會大幅降低，即便簡單的查詢指令亦耗費不少時間，且當物件關係以交互參照表在關聯式資料庫進行對應(mapping)時，資料結構將會過於複雜而使得 SQL 指令難以公式化，故其研究改採分散式的模型伺服器架構，將不同領域、不同軟體和不同需求的模型元件分開來管理。因此，目前已有許多研究改採 NoSQL 資料庫來處理 BIM 的模型資訊或生命週期資訊，舉例如下：

- Rasys et al. (2014)使用 MongoDB 來整合 web-based 設施維護管理資訊及其所關聯之物件 3D 幾何資訊，由於 MongoDB 採用 JSON (JavaScript Object Notation)作為資料庫綱要，然後再轉為二進制格式

BSON (Binary JSON)儲存，經由其測試已證明 MongoDB 在效率及效能上均較傳統 MySQL 關聯式資料庫更佳。

- Lin et al. (2013)的研究將欲進行分析之 IFC 模型資訊以 MongoDB 來儲存，並與傳統直接將 IFC 檔案讀取至記憶體的 file-based 操作方式作比較，就語意分析(parsing)的轉換時間而言，MongoDB 的轉換效率高出 file-based 方式許多，且對資料庫進行查詢的速度亦相當快。
- Ma (2012)的研究也顯示以物件導向資料庫(OODB, Object-Oriented DataBase)來存取 IFC 格式的資料比傳統關聯式資料庫更有效率。
- Chen and Hsieh (2013)以 Neo4j 圖形資料庫來作為規則式(rule-based)資料引擎，以在 BIM 設計階段輔助檢核綠建築的相關規範。
- Chen et al. (2013)以 HBase 資料庫搭配 MapReduce 技術來解析、擷取 3D BIM 模型物件與其屬性資料，開發結合雲端運算技術與 WebGL 技術之網路式 BIM 系統。

從以上研究的年份來看，採用 NoSQL 資料庫來管理大量的 BIM 資訊已成一種新的趨勢，但 NoSQL 資料庫有上百種，依資料庫性質不同的分類方式也有許多種，如何挑選適合本系統的資料庫管理系統實為一大難題，故本研究選擇較熱門、網路上相關資源也較多的 Redis 資料庫(Sanfilippo and Noordhuis, 2010)來作為本系統之資料庫管理系統，主要考量為 Redis 提供五種不同的資料結構，可根據實際需求來設計新的鍵值(key-value)資料結構(詳如 4.2 節)，且 Redis 為持久性(durability)的記憶體(in-memory)資料庫，既可提供高效能的 CRUD (Create, Read, Update and Delete)操作，並能將資料永久保存，同時亦支援原子操作(atomic operation)，許多知名的網站或應用程式皆有採用 Redis 作為資料庫，例如：Twitter、Github、Weibo、Pinterest、Flickr、StackOverflow、Line 等。

- **相容性**

  由於本系統需要開發 local 端的建模工具外掛程式，又要開發 server 端的網頁伺服器，故 Redis 可謂相當適合，因其在網路上有許多不同的資料庫客戶端應用程式介面，支援許多不同的開發平台和程式語言，例如：C、C++、Java、C#、JavaScript、Node.js、PHP、Ruby 等，因此 Redis 在不同平台開發的相容性理論上是沒有問題的。

- **延伸性**

  Redis 提供許多進階的資料庫管理方式，例如：資料交易(transaction)、資料生存時間(expiration)、資料排序、訊息通知(即發布訂閱模式，publish/subscribe)、管道(pipelining)、腳本(script)等。

- **可擴展性**

  雖然 Redis 不支援多執行緒(multi-thread)，但其在架構設計上亦可進行分散式運算，因 Redis 提供複製(replication)的功能，即資料可進行主從(master/slave)同步。此外，目前亦有許多雲端供應商提供平台即服務(PaaS, Platform as a Service)，亦即讓 Redis 資料庫可在雲端架設的服務，對於設備管理和可擴展性而言更添便利性。

- **安全性**

  Redis 在安全性層面並沒有太多設計，因其假設 Redis 是在可信賴的環境下運作，故 Redis 並不適合讓外界透過網際網路直接連接，雖然其仍可綁定 IP (Internet Protocol)位址、設置密碼和重新將命令(command)命名等，但額外設計一個 web 服務對於安全性來說是絕對有必要的，如此可讓外界透過 web 服務向 Redis 進行 CRUD 操作，web 服務除了可將資料庫的存取模式制式化，亦有鬆散耦合的設計彈性，惟 web 服

務若要確保安全性，仍須引入驗證(authentication)機制，有關本系統的 web 服務設計將在 4.3 節介紹。

(3) 基於 IFC 之網頁伺服器

網頁伺服器的開發可借助 web 應用程式框架(WAF, Web Application Framework)，其可用來建置動態網站、網路應用程式和網路服務等，web 應用程式框架將常用的網站開發模組整合在一起，例如：資料庫存取介面、樣板系統、資料快取、URL (Uniform Resource Locator)對應等等，在進行網頁開發時可更加有效率。由於 web 應用程式框架也有上百種，故可先從熟悉的程式語言來挑選，本研究考量對於解析 IFC 檔案格式的需求，因此採用以 Java 為開發語言的 Play Framework，許多知名的網站皆是以其來開發，例如：專業人士之社群網路 Linkedin、高等教育之線上課程平台 Coursera 等，有關網頁伺服器的設計細節將在 4.5 節詳述。

- **相容性**

  基本上，網頁是透過瀏覽器來觀看，故無論使用何種硬體裝置或作業系統都比較沒有相容性的問題，只有前端的使用者介面在不同裝置或瀏覽器上或有可能會發生問題，故本研究採響應式的網頁使用者介面，其設計細節將在 4.5 節再說明。

- **延伸性**

  如同 3.2 節所述，雖本研究只設計(而不實作)一個基於 IFC 的網頁伺服器原型系統，但為使應用情境導向之資訊管理框架(圖 3)能更加完善，爾後仍會有整合 IFC 模型之需求，這也是本研究採用 Java 語言的 web 應用程式框架之原因，因目前已有 Java 版本的 IFC 語法分析器(parser)可供使用(Tauscher and Theiler, 2013)。

- **可擴展性**

  由於 Play Framework 的 URI (Uniform Resource Identifier)路由(routing)系統是採用 RESTful (REST, REpresentational State Transfer ful)設計架構，故可藉由分散式伺服器來達成負載平衡(load balancing)。此外，目前亦有許多雲端供應商提供平台即服務，亦即讓 Play Framework 可在雲端架設的服務，對於設備管理和可擴展性而言更添便利性。

- **安全性**

  網站的安全性有很多層面，例如：攻擊行為預防、權限管理等。由於此部分並非本研究之重點，故本研究僅在 4.6 節引入開放授權(OAuth, Open Authorization)之技術概念，若本系統之網頁伺服器爾後欲與專案管理系統結合時，可透過 OAuth 服務提供方(service provider)的認證機制來進行權限管控。

第四章　系統設計

4.1　整體系統架構設計

本研究欲開發之系統稱為「可互操作型物件式資訊回饋系統」，為使此系統具鑑別度和親和力，本研究將其命名為 BIMFeeD，此後章節將以 BIMFeeD 來作為此系統之產品名稱，根據 3.2 節對於 BIMFeeD 之產品定位和規劃，其在本研究的實作項目可由圖 5 知系統架構圖來表示。

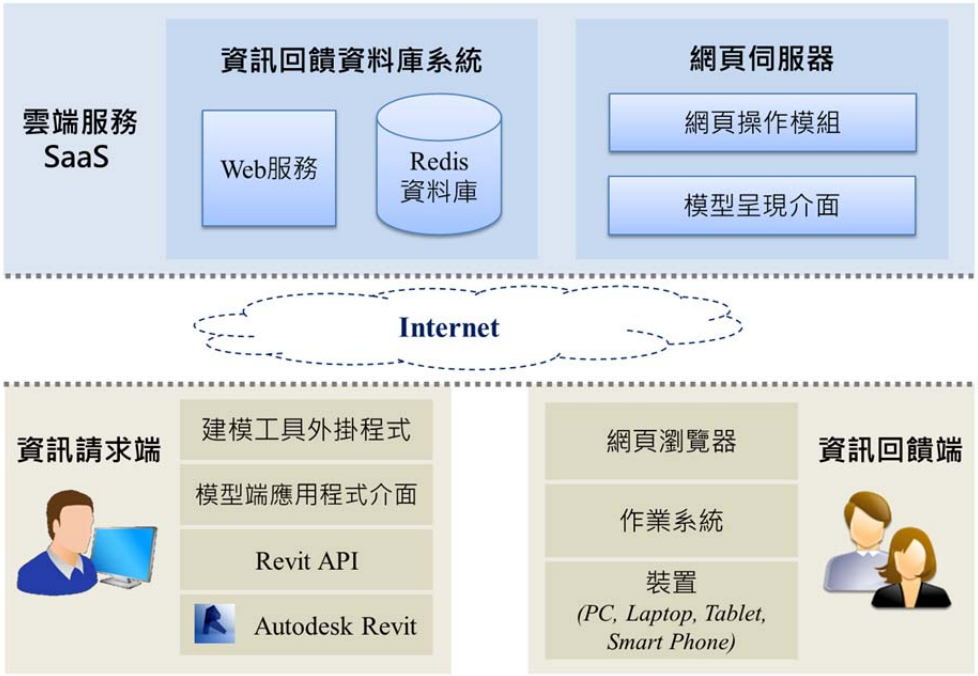


圖 5：BIMFeeD 系統架構圖

圖 5 顯示 BIMFeeD 的使用者分為資訊請求端和資訊回饋端，如 1.2.1 節所述，前者為操作建模工具 Autodesk Revit 之模型管理者，後者為使用各種不同裝置、作業系統和網頁瀏覽器的專案利害關係人。資訊請求端和資訊回饋端皆透過網際網路分別連接資訊回饋資料庫系統和網頁伺服器，就資訊回饋資料庫系統的 web 服務和網頁伺服器的網頁操作模組而言，使用者進行資料交換的方法是一種雲端服務的應用模式，也就是軟體即服務(SaaS, Software as a Service)。

有關 BIMFeeD 的主要運作流程已藉由圖 4 在 3.3 節說明，為使圖 4 中的關鍵任務或事件更具體化，本研究將 BIMFeeD 運作流程重新包裝為商業邏輯，以下根據圖 4 中的任務或事件來說明：

- **(6)輸入請求資訊並送出**在 BIMFeeD 中代表一個任務(task)，此任務根據不同的建模工具而有不同的任務類型(task type)，資訊請求端使用者可在建模工具外掛程式中建立任務(create a task)，一個任務代表含有關連到至少一個模型元件的請求資訊(即**(14)請求資訊之資料結構**)，並賦予此任務一任務名稱(task name)和一任務描述(task description)，同時輸入此任務接收者(task recipients)之電子信箱地址(email)，任務接收者即資訊請求端欲指定的資訊回饋端使用者。
- **(22)輸入回饋資訊並送出**在 BIMFeeD 中代表一個建議(advice)，資訊回饋端使用者可透過網頁操作模組來檢視資訊請求端使用者所發出之任務和其請求資訊，然後根據請求資訊來輸入該任務之建議，此建議必須包含資訊回饋端使用者的姓名(name)、電子信箱地址(email)及建議內容(advice)，即**(24)回饋資訊之資料結構**，而資訊請求端使用者可在建模工具外掛程式中檢視不同任務、不同資訊回饋端使用者之建議。
- 一個任務的初始狀態為待判(pending)，資訊請求端使用者可根據資訊回饋端使用者對於該任務的建議來改變任務的狀態，如**(32)排他閘道**所示，一個任務的狀態可被接受(accepted)或取消(cancelled)，資訊請求端使用者有權限將任務狀態指定為這三種狀態的其中一種。
- 同一個任務可能會有很多個建議，因為資訊回饋端使用者可能超過一位，抑或同一個資訊回饋端使用者多次修改已經送出的建議，每一次送出建議皆會留下記錄，當資訊請求端使用者同意某個建議而將任務的狀態指定為接受時，該建議將顯示為接受的建議，即**(36)判定結果**。

- 從**(1)開始事件**至**(37)結束事件**，必須在同一個模型專案中執行，即任務必須歸屬於某一個模型專案，不同的模型專案應有各自不同的任務。

接下來的 4.2 節至 4.6 節將依照上述操作邏輯來進行系統設計，將 BIMFeeD 各個構成要素的設計原理和結果逐一闡釋。

資訊回饋資料庫管理系統主要包含 Redis 資料庫和 web 服務兩部分，有關資料庫的資料結構設計將在 4.2 節介紹，而 web 服務的主要目的是確保資料傳輸和資料庫的安全性，其資訊傳遞方法的設計原理將在 4.3 節介紹。

資訊請求端的使用者主要透過操作建模工具外掛程式來發出請求資訊，本研究規劃的首個建模工具外掛程式是以 Autodesk Revit 為基礎，利用其 Revit API 和本研究自行開發之模型端應用程式介面來實作，其設計細節將在 4.4 節介紹。

如 3.4 節所述，由於不同的建模工具或有不同的商業邏輯，故建模工具外掛程式必須根據建模工具的資料模型來設計，就 Autodesk Revit 而言，由於其模型元件的參數化屬性分為類型(type)屬性和實作元件(instance)屬性(Autodesk 官方稱為例證屬性)，且模型元件的位置(location)、材質(material)和表面油漆(paint)不一定屬於類型屬性或實作元件屬性，故這五種物件資訊必須分開來處理，BIMFeeD 將以五種不同的任務類型來區分，其設計細節同樣於 4.4 節介紹。

資訊回饋端的使用者主要透過模型呈現介面來檢視資訊請求端使用者所發出之請求資訊，然後藉由網頁操作模組來提供其建議。模型呈現介面和網頁操作模組在 BIMFeeD 的系統架構中屬於網頁伺服器的系統開發範疇，此部分主要工作為開發網路應用程式，其相關設計細節將在 4.5 節詳述。

本研究的重點在於設計一套資訊回饋方法以修改或更新 BIM 模型，然而 BIM 模型的建置和管理包含許多廣泛的應用層面和議題，整個資訊管理的流程和機制若欲完善則需要更多軟體系統的配合(如：專案管理系統)，但為了避免研究重點失焦和減少不必要的資源排擠，本研究將 BIMFeeD 分為兩種版本：入門版(BIMFeeD

Lite)和專業版(BIMFeeD Pro)，本研究僅實作入門版，即專注於開發資訊回饋系統之核心功能，如同 3.2 節對於產品定位和規劃所描述的一些權宜設計方法，至於專業版的主要目的為配合實務上的 BIM 模型管理需求，故必須提供更健全的輔助功能或電腦系統，有關專業版系統功能之需求分析將在 4.6 節說明，本研究將僅探究其設計概念，並嘗試設計一部分的原型系統，而不進行專業版的系統實作。

4.2 資料結構設計

在 3.4 節已提及 NoSQL 資料庫的種類繁多，每種資料庫皆有其特色和擅長處理的資料型態。整體而言，NoSQL 資料庫之所以能在處理巨量資料時保持高效能和高彈性，其中的一個關鍵因素為：使用者可自行定義免綱要式(schema-free or schema-less)的資料模型，即資料庫綱要可根據實際需求隨時調整，因此有利於處理半結構化(semi-structured)資料、非結構化(unstructured)資料和後設資料(metadata)(又稱詮釋資料、元資料) (Tiwari, 2011; Bakshi, 2012)。

就資料模型而言，鍵值資料庫(key-value store)和文件資料庫(document store)是其中兩種 NoSQL 資料庫的分類法，前者以鍵(key)搭配值(value)來儲存資料，值的部分可以是任何資料型態，後者則以 JSON、BSON 或 XML (eXtensible Markup Language)來儲存資料，這兩種資料模型皆非常適合用來記錄物件式資訊(Han et al., 2011; Li and Manoharan, 2013)。

根據 DB-Engines 網站的統計資料(Solid IT, 2012)，其係以熱門度(popularity)作為資料庫的排名根據，自有記錄以來，長期居鍵值資料庫和文件資料庫排行首位的分別是 Redis 和 MongoDB，若將兩者相提並論，MongoDB 的排名一直是 NoSQL 資料庫的第一名，可謂遠遠高於 Redis。在 3.4 節已回顧不少近年來的 BIM 研究係採用 MongoDB，而 BIMFeeD 則採用 Redis，因本研究認為 Redis 較適合作為資訊回饋資料庫系統，原因在 3.4 節已大致說明，由於 Redis 是記憶體資料庫，故其資料存取速度相當快，可滿足資訊回饋的即時性要求，另外 Redis 亦提供豐富

的內建資料結構，有利於根據不同建模工具的資料模型來設計符合資訊交換需求且更具彈性的資料結構，即 BIMFeeD 的資料結構可隨不同的建模工具來擴充調整，雖然 MongoDB 和 Redis 皆是免綱要式資料庫，但 Redis 可自行定義更具彈性、更複雜的資料結構。簡單來講，MongoDB 提供易於操作的資料結構設計方式和資料庫 API，而 Redis 在使用上則不那麼方便，但其資料結構能有比較複雜的設計。

Redis 提供的五種資料結構為：String、List、Set、Sorted Set 和 Hash，本研究有用到的是 String、List 和 Hash，每一種資料結構都有一個鍵和一個值，鍵和值在 Redis 底層皆是以 byte[]來儲存的，以下針對此三種資料結構作說明：

- String 即字串，一個資料結構屬於 String 的鍵存有一個字串，其值若以 byte[] 的角度來看(即 binary-safe)，String 可儲存一般字串、圖片或序列化的物件 (serialized object) (稍後解釋)。
- List 即列表，其可儲存一個有序的字串列表，即一個資料結構屬於 List 的鍵存有多個有序的字串，可由列表兩端推進(push)和彈出(pop)數個字串，抑或直接取得 List 的某一個片段的字串組合。
- Hash 即散列表，又稱雜湊表或哈希表，在 Redis 當中的 Hash 為一種字典結構，其結構分為鍵(key)、欄位(field)和值(value)，一個資料結構屬於 Hash 的鍵可以有多個欄位，每個欄位可以有一個值，這個值可以是 String 或 byte[]，Hash 的資料結構如圖 6 所舉例。

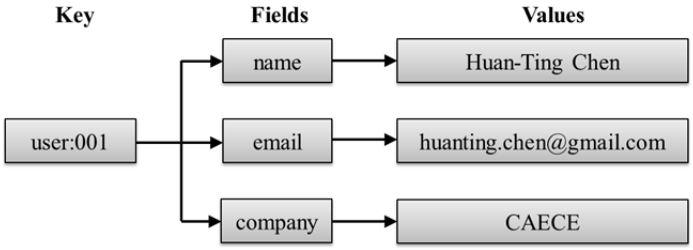


圖 6：Hash 資料結構舉例

從圖 6 便可看出 Hash 的好處，其資料結構非常類似於 BIM 模型元件的物件識別碼、參數化屬性和參數化屬性值的物件結構。

通常鍵會以冒號來分隔以區別不同的識別碼，例如圖 6 中的 user:001 可代表 user id 為 001 之使用者，又或 user:002:task 可代表 user id 為 002 的使用者所管轄的 task，此鍵背後可能是一個 List 結構，基本上鍵可依實際需求自行設計。

由於 Redis 每一種資料結構背後皆是以 byte[]儲存，故 String 的值、List 當中的值和 Hash 的欄位值皆可以透過物件序列化(object serialization)的方式來儲存一個物件，所謂物件序列化指的是將物件的狀態(state)轉化為字串(stringify)的過程，並且能透過語法分析(parse)的方式再轉回物件，被轉化為字串的物件稱為序列化的物件(serialized object)，而將序列化物件轉回物件的過程稱為物件反序列化(object deserialization) (Flanagan, 2011)。

常用的物件序列化格式有 XML 和 JSON，然而 JSON 已被許多研究證明其資料轉換效率明顯高出 XML 許多(Nurseitov et al., 2009; Sumaray and Makki, 2012)，故本研究將採用 JSON 作為物件序列化的格式，也就是說，String 和 List 都可以用來儲存以 JSON 型式存在的序列化物件，而 Hash 因為有欄位的關係，本身就可以當作是物件使用，但其欄位值亦可用來儲存以 JSON 型式存在的序列化物件，等同於將序列化物件存放於一個以 Hash 為資料結構的物件欄位中。

從以上敘述便可看出 Redis 在資料結構設計上的彈性，只要能夠事先知道物件的型別(type)，甚至也能達到物件導向繼承和多型的效果。本研究歸納幾個資料結構的設計原理或原則，但有時這些設計考量會互相衝突，或可依照傳統智慧「兩害相權取其輕」來處理：

- 當一個物件內的成員(member)有不同存取頻率時可用 Hash，反之用 String 即可，原因是 Hash 的欄位值可直接存取，若將序列化物件存放在 String 則尚需要進行反序列化才能存取。

- 承上，若當一個物件內的成員有不同存取頻率，但其中又有部分成員有相同的存取頻率時，則可以將相同存取頻率的資料以序列化物件的方式存放在 Hash 的欄位值。
- 物件直接以 Hash 來儲存，抑或以序列化物件存放在 String，兩者在物件資料量不大時所呈現的效果差不多，但當物件成員非常多時 Hash 的存取將變得沒有效率，因為 Hash 必須將欄位逐一存取，而 String 則是將一次將整個序列化物件存取。
- 若物件的某一個成員有需要進行原子操作時，則必須存放在 Hash 的欄位值，而不能以序列化物件來儲存，例如：物件中有一個成員是計數器(counter)，若將整個物件轉為序列化物件，則當此序列化物件被同時間存取時，便有可能發生計數器錯誤計數的情況。
- 由於 Redis 是記憶體資料庫，故記憶體的用量應該越少越好，故某些不需要永久保存的資料可以設定其生存時間(TTL, Time To Live)。
- 承上，鍵的名稱不宜太長，但也不能太短，以免影響可讀性(readability)。
- 雖然 JSON 的物件序列化和物件反序列化已經相當有效率，但仍需耗費系統運算資源，若有大量重複的資料便不建議以 JSON 中的陣列(array)來儲存，而應該將其存放在 List。

BIMFeeD 所需儲存的資料和建模工具的商業邏輯有關，待 4.4 節再說明，此處先將資料結構之設計結果整理如表 2，其中鍵名或值包含識別碼的部分以{ }來表示，若某個鍵的值或 Hash 的欄位值是序列化物件則以該物件的類別(class)名稱表示，若僅是單純的字串或列舉(enumeration)則以“ ”表示，而序列化物件的資料模型則以 UML (Unified Modeling Language)類別圖(class diagram)在圖 7 至圖 11 中表示。

表 2 中的 taskLite:{taskID}和 taskLite:advice:{taskID}兩個鍵名，分別是以 TaskLite 和 Advice 這兩個序列化物件的多型來儲存，故實際上這兩個序列化物件是以其子類別來儲存，即圖 9 和圖 10 中所示之繼承關係，而這兩個序列化物件的執行期型別資訊(RTTI)則由 TaskLite 物件中 taskInfo 成員的 taskType 成員來判斷。

表 2：BIMFeeD 之 Redis 資料結構鍵值表

鍵名	資料結構	欄位	值
projectLite:{projectID}	Hash	projectInfo	ProjectInfo
		modelManagerInfo	ModelManagerInfo
projectLite:tasks:pending:{projectID}	List	N/A	“{taskID}”
projectLite:tasks:accepted:{projectID}	List	N/A	“{taskID}”
projectLite:tasks:cancelled:{projectID}	List	N/A	“{taskID}”
taskLite:{taskID}	Hash	taskLite	TaskLite
		taskStatus	“TaskStatus”
		projectID	“{projectID}”
		acceptedAdviceIndex	“index”
taskLite:advice:{taskID}	List	N/A	Advice

圖 7：BIMFeeD 序列化物件資料模型之 UML 類別圖(1)

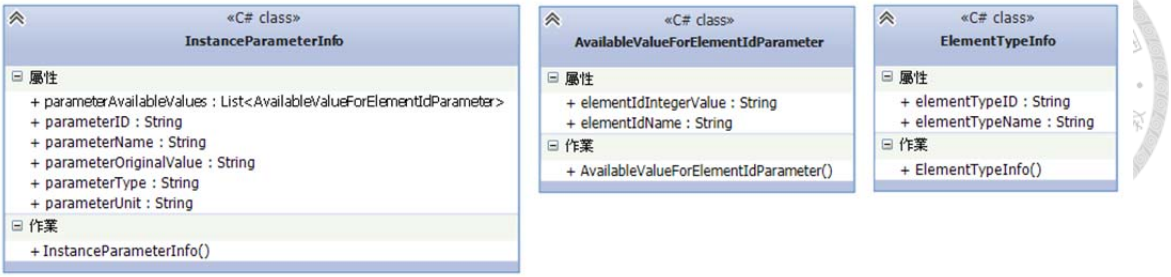


圖 8：BIMFeeD 序列化物件資料模型之 UML 類別圖(2)

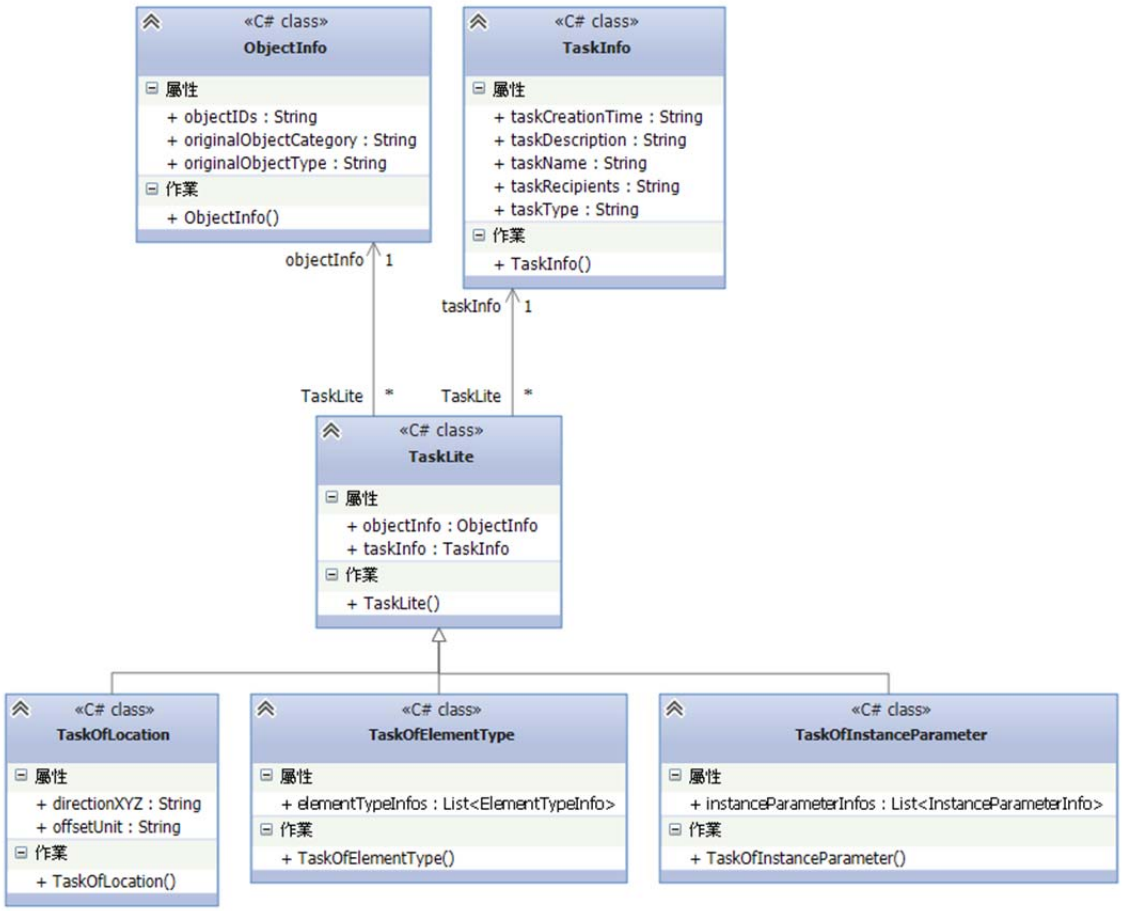


圖 9：BIMFeeD 序列化物件資料模型之 UML 類別圖(3)

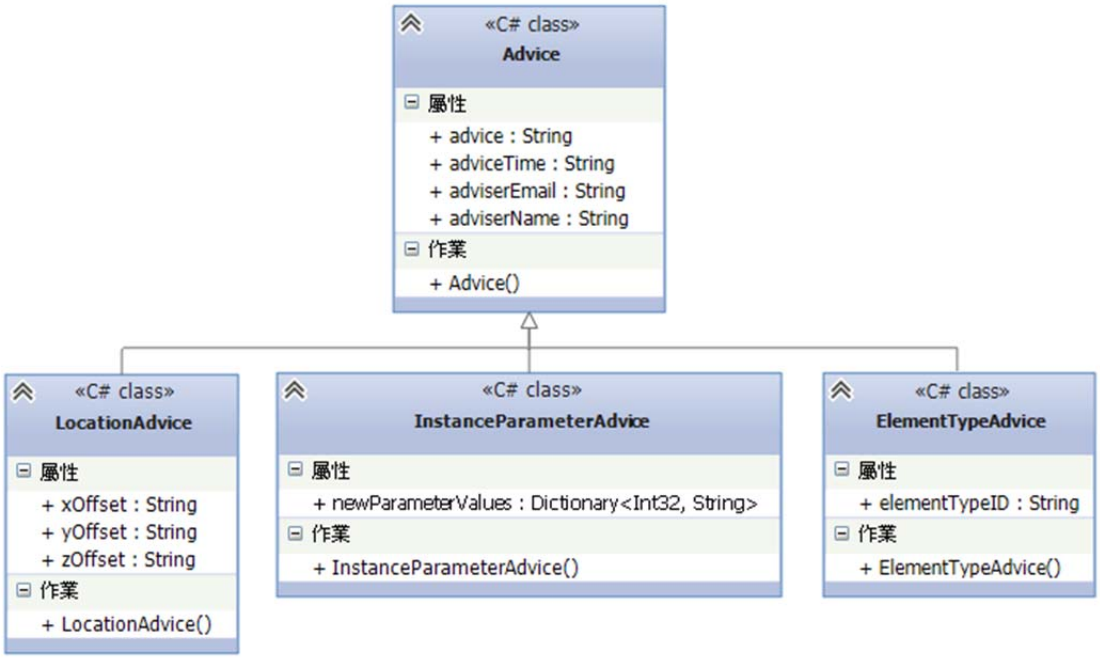


圖 10：BIMFeeD 序列化物件資料模型之 UML 類別圖(4)

圖 11：BIMFeeD 序列化物件資料模型之 UML 類別圖(5)

4.3 資訊傳遞方法設計

在 3.1 節已提及 web 服務的必要性，就資料傳遞的服務而言，必須能夠在不同的應用程式之間分享共同的資料處理邏輯，並能在不同的電腦平台上進行協同作業，一個平台可能是不同硬體、作業系統、軟體框架或程式語言的組合，所以服務應該要在客戶端的外部流程執行，web 服務便是一種整合異質系統的資料交換途徑，其經由 HTTP 通訊協定來操作可重複使用的商業功能，並可將其作為一個完整的應用程式協定，該協定亦可根據語意來定義服務行為(Daigneau, 2012)。

由於 BIMFeeD 的資訊必須透過網際網路傳輸，讓其符合服務導向架構(SOA, Service-Oriented Architecture)是可行的辦法，因服務導向架構可帶來抽象化(即對外部隱藏商業邏輯)、鬆散耦合和資訊封裝的好處。Isikdag (2012)歸納三種在網際網路進行 BIM 資訊共享的設計模式(design pattern)，分別是：BIM AJAX (Asynchronous JavaScript and XML)、BIM SOAP (Simple Object Access Protocol) Façade 以及 RESTful BIM，並比較其異同。

該研究提到 BIM AJAX 較適合模型幾何物件的視覺化，因 AJAX 技術可讓網路應用程式的互動更即時(如：在不更新頁面的情況下存取資料)，就像 Chen et al. (2013)的研究也採用 AJAX 技術來開發 web-based 模型瀏覽介面，但 AJAX 的腳本(script)必須在 client 端的應用程式實作，即 client 端的程式邏輯和資料模型是緊密耦合(tightly-coupled)，如此便不符合服務導向架構。此外，SOAP 技術雖然發展歷史悠久，但使用上卻不如 RESTful 方便，因 SOAP 必須基於正式的介面封裝和查詢規則，所以 RESTful BIM 的資訊交換方式應比較適合 BIMFeeD 的需求。

另一方面，該研究並沒有特別提及資料傳輸格式的效率問題，因為 AJAX 和 SOAP 都必須以標記語言(markup language)來作為資料傳輸的格式，這會使得資料傳輸量較 BIMFeeD 所採用的 JSON 格式高出許多，而 RESTful 則可採用 XML 或 JSON 格式，故 BIMFeeD 的 web 服務採用 RESTful 搭配 JSON 應是最好的選擇。

REST (Fielding, 2000)是一種軟體架構風格，而非一項標準或協定，而符合 REST 架構的系統可稱為 RESTful。由於 REST 將網路上的所有資料或軟體視為資源(resource)，每一個資源皆有和其對應的 URI，也就是網址的概念，對於資源的 CRUD 操作正好可對應 HTTP 通訊協定的四種方法，即 GET、POST、PUT 和 DELETE，而資源該以何種型式呈現則視情況而定，例如：HTML (HyperText Markup Language)、XML、JSON 等。因此，RESTful 的 web 服務應包含三個部分：資源位置、資源傳輸的格式以及對資源的操作方法，例如：欲取得 user id 為 001

的使用者資料，其資源位置可能是 http://example.com/user/001，資源格式可以為 JSON 或 XML，當中記錄有關這位使用者的相關資訊，而對資源的操作方法則為 HTTP 的 GET 方法。

表 3 為 BIMFeeD 之 web 服務 API，其包含 URL 位址、所搭配的 HTTP 方法、所需傳遞的參數及該 HTTP 方法所回應的內容，在 BIMFeeD 之 web 服務當中，所有回應內容(response)的資料傳輸格式皆為 JSON，故回應內容有可能是 JSON 化的序列化物件、JSON 化的序列化物件陣列、JSON 化的字串或 JSON 化的字串陣列，其中包含識別碼的部分以{ }來表示，若僅是單純的字串則以“ ”表示，而序列化物件則以圖 11 中的類別名稱來表示，其資料模型必須符合圖 11 中的物件關係，如同 4.2 節所述，TaskLite 和 Advice 這兩個序列化物件是以其多型來表示。

此外，本研究將 HTTP 方法為 POST 的 API 設計為必須傳遞參數(parameter)，即 HTTP 之 request body 必須包含一參數字串，以將不同類別的序列化物件或字串傳回至資訊回饋資料庫系統，例如：若要新增一個任務，則必須使用表 3 中的第一個 API，即 URL 位址為/api/task，HTTP 方法為 POST，所需傳遞之參數組成的字串為 projectID=xxx&taskLite=xxx&isSendCopy=xxx，其中 projectID 為模型專案識別碼、taskLite 為 TaskLite 之序列化物件、isSendCopy 為是否將此任務資訊傳送給模型管理者的字串(當中記錄一布林(boolean)值)，若此任務成功地在資訊回饋資料庫系統中被建立，則 web 服務會回傳此任務之任務識別碼，即 taskID。

這套 API 為資訊回饋資料庫系統的 web 服務，可供資訊請求端之建模工具外掛程式和資訊回饋端之網頁操作模組來使用，若爾後要將此套 API 開放給其他系統開發者(developer)使用，出於安全性考量，則必須引入使用者驗證機制(如：HTTP basic authentication 或 OAuth)，同時根據不同的資料處理邏輯回傳不同的 HTTP 回應(HTTP response)和狀態代碼(status code)。

表 3：BIMFeeD 之 web 服務 API

URL	HTTP 方法	參數	回應內容
/api/task	POST	projectID taskLite isSendCopy	“{taskID}”
/api/project_id	GET	N/A	“{projectID}”
/api/project_info	POST	projectID projectInfo	N/A
/api/model_manager_info	POST	projectID modelManagerInfo	N/A
/api/model_manager_info/{projectID}	GET	N/A	ModelManagerInfo
/api/pending_task_ids/{projectID}	GET	N/A	“{taskID}”陣列
/api/accepted_task_ids/{projectID}	GET	N/A	“{taskID}”陣列
/api/cancelled_task_ids/{projectID}	GET	N/A	“{taskID}”陣列
/api/task_status	POST	taskID taskStatus	N/A
/api/task/{taskID}	GET		TaskLite
/api/accepted_advice_index/{taskID}	GET	N/A	“index”
/api/accepted_advice_index/	POST	taskID acceptedAdviceIndex	N/A
/api/advice/{taskID}	GET	N/A	Advice 陣列

4.4 外掛程式設計

外掛程式依附於建模工具，主要功用是作為資訊請求端使用者的操作系統，模型管理者可透過 BIMFeeD 的建模工具外掛程式來建立任務，並且將此任務發送給指定的資訊回饋端使用者，故建模工具外掛程式主要功能即是建立任務、管理任務和管理模型專案等。

如同 4.1 節所述，BIMFeeD 建模工具外掛程式的 Autodesk Revit (以下簡稱 Revit) 版本有五種不同類型的任務，如圖 11 中的 TaskType 列舉類別，當資訊請求端使用者選取至少一個同元件類型(element type)的模型元件時，其可以透過 BIMFeeD 來建立以下五種任務：

- **Instance Parameter**：即 Revit 中的實作元件參數，BIMFeeD 會將非唯讀(not read-only)的實作元件參數清單提供給使用者，讓其勾選與此任務相關的實作元件參數，以讓資訊回饋端使用者在網頁操作模組上提供建議，即提供新的實作元件參數值，例如：窗戶元件有一個「窗台高度」的實作元件參數，模型管理者可選取某些同元件類型的窗戶元件，並透過 BIMFeeD 建立 Instance Parameter 任務，同時將窗台高度這個實作元件參數勾選，資訊回饋端使用者即可在網頁操作模組上提供窗台高度的建議值。
- **Element Type**：即 Revit 中的元件類型，BIMFeeD 會判斷所選取的元件類型，將可進行替換的元件類型清單提供給使用者，讓其勾選與此任務相關的元件類型，以讓資訊回饋端使用者在網頁操作模組上提供建議，即提供元件類型的更換建議。例如：牆元件有「RC 牆 10 公分」、「RC 牆 12 公分」和「RC 牆 15 公分」三種元件類型，模型管理者可選取某些同元件類型的牆元件，並透過 BIMFeeD 建立 Element Type 任務，同時將 12 公分和 15 公分這兩個元件類型勾選，資訊回饋端使用者即可在網頁操作模組上將牆元件類型更換為「RC 牆 12 公分」或「RC 牆 15 公分」。

- **Location**：即 Revit 中的元件位置，BIMFeeD 會提供 X、Y 和 Z 三個座標方向的偏移(offset)選項，使用者可勾選至少一個座標方向的偏移選項，以讓資訊回饋端使用者在網頁操作模組上提供建議，即提供各座標方向的偏移值，例如：模型管理者可選取某些同元件類型的模型元件，並透過 BIMFeeD 建立 Location 任務，同時將 X 方向和 Y 方向勾選，資訊回饋端使用者即可在網頁操作模組上輸入 X、Y 方向的偏移值，以達到調整元件位置的效果。
- **Material**：即 Revit 中的材質，事實上某些模型元件已有和材質相關的實作元件參數，或是元件類型的更換也可能涉及不同材質的替換，因此 Material 任務主要以整個模型元件的材質來考量，因為同一個模型元件可能由許多細部材料或複層結構所組成，由於這部分的操作邏輯較複雜，故 Material 任務設定為 BIMFeeD 專業版的功能，本研究的入門版將不實作。
- **Paint**：即 Revit 中的油漆，其指的是模型元件表面的材質，基本上和 Material 任務非常相似，Paint 任務則主要處理模型元件表面的材質，基於同樣的考量，Paint 任務設定為 BIMFeeD 專業版的功能，本研究的入門版將不實作。

在上述五種任務當中，BIMFeeD 入門版提供了三種不同類型的任務，其中複雜度較高的當屬 Instance Parameter 任務，因為實作元件參數在程式邏輯上又有許多變數型別，如圖 11 中的 InstanceParameterType 列舉類別，包含：Boolean、Double、ElementID、Enumeration、Integer 和 String，BIMFeeD 的需求和 Revit API 中的分類方法又不盡相同，因此有兩個比較特別的實作元件參數類型要特別處理，以下針對 ElementID 和 Enumeration 作說明：

- **ElementID**：在 Revit 中有些實作元件參數看起來像是字串，但實際上則是以 ElementID 來儲存，例如：樓層可能顯示為 2FL，但背後則對應到一個樓層元件，此時樓層所顯示的 2FL 便不是單純的字串，故 BIMFeeD 需要將同

元件類型的 ElementID 找出來，在此例中即是將所有的樓層元件找出來，以讓資訊回饋端使用者以下拉式選單的方式來指定正確的樓層位置。

- **Enumeration**：在 Revit 中有些實作元件參數看起來像是字串，但實際上則是以列舉的整數值(integer)來儲存，例如：牆定位線可能顯示為牆中心線，但背後則是對應到 Revit API 不一定有的列舉類別，該列舉類別包含所有牆定位線的列舉，即牆中心線、核心線、塗層面: 外部、塗層面: 內部、核心面: 外部、核心面: 內部，每一個列舉皆對應到一個整數值，如：牆中心線的列舉值為 0，此時牆定位線所顯示的牆中心線並非單純的字串，然而此情況不一定能透過 Revit API 來擷取列舉值所代表的字串，故 BIMFeeD 額外新增一個 Enumeration Checker 的功能，讓使用者自行透過選取不同的實作元件參數值，以檢視其背後所代表列舉值，並且必須將這些資訊傳遞給資訊回饋端的使用者知道(可輸入於任務描述中)，因為受限於列舉實際上以整數值來儲存的關係，在網頁操作模組上也只能輸入一整數值。

事實上，在任務建立之前，資訊請求端使用者會需要作兩個前置作業，其一為將模型專案的基本資訊傳送至資訊回饋資料庫系統，由於 Revit 中已有可供建立專案資訊(project information)的功能，故 BIMFeeD 會直接擷取其相關資訊，專案資訊的資料模型如圖 11 中的 ProjectInfo 類別；其二為將模型管理者的資訊傳送至資訊回饋資料庫系統，模型管理者資訊的資料模型如圖 11 中的 ModelManagerInfo 類別。這兩個前置作業的目的為：讓資訊回饋端使用者在網頁操作模組得以檢視每一個任務所關聯的模型專案資訊和模型管理者資訊。

而在任務建立之後，BIMFeeD 必須提供資訊請求端使用者檢視任務清單的介面，如 4.1 節所述，任務可能是 pending、accepted 或 cancelled 三種狀態，如圖 11 中 TaskStatus 列舉類別所示，由於這些任務的狀態資訊是儲存在資訊回饋資料庫系統，為避免一次存取所有的任務資訊，故 BIMFeeD 將三種不同狀態的任務清單分

為三個不同的介面，但事實上大同小異，pending 的任務清單介面可以重新整理，以確認是否有新的建議傳送進來，而 accepted 的任務清單介面可以將被接受的任務資訊匯出為 CSV (Comma-Separated Values)格式。在這三種任務清單介面皆可以顯示與任務相關的模型元件，即 show object，BIMFeeD 會將關聯的模型元件在視圖中亮顯(highlight)並置中。

模型管理者可在 pending 的任務清單介面中判定資訊回饋端使用者的所有建議，其可將一個任務的狀態指定為 accepted 或 cancelled，若指定為 accepted 時，與此任務相關的模型元件會遵照所接受的建議進行修正或更改，並將此任務改存放至 accepted 任務清單；若指定為 cancelled 時，則不進行任何動作，僅將此任務改存放至 cancelled 任務清單。同樣地，accepted 和 cancelled 的任務清單介面也能透過類似的方式來指定任務的狀態，即任務的目前狀態是可以透過 BIMFeeD 的不同任務清單介面來指定為另外兩種任務狀態的其中一種。

BIMFeeD 外掛程式的 Revit 版本主要透過 Revit API 的外部指令(external command)作為不同功能的程式進入點，圖 12 為這些外部指令的 UML 類別圖，每一個類別皆對應到一個特定功能(即本節所述之外掛程式功能)，這些外部指令類別皆繼承自 BIMFeeDExternalCommand 的抽象類別(abstract class)，並由其中的 showUI 抽象方法(abstract method)來啟動不同功能的使用者介面。

就系統設計的角度而言，BIMFeeD 透過不同的命名空間來管理不同性質的類別，如圖 13 所示，BIMFeeD.RevitEntry 代表外掛程式的外部指令類別(即外掛程式的程式進入點)、BIMFeeD.DataModel 代表圖 11 的資料模型類別、BIMFeeD.UI 代表外掛程式的使用者介面類別、BIMFeeD.WebService 代表外掛程式使用 web 服務 API 的類別、BIMFeeD.Utility 代表一些公用程式的類別。

另外，負責透過 web 服務來存取資料的 UML 類別圖如所示。而公用程式的功能則主要包含 Email 驗證、字串驗證、螢幕截圖、時間轉換、單位轉換、檔案上傳、

表單上傳和 HTTP request 等類別，由於並非本研究之重點，故公用程式的 UML 類別圖便不特別展示。

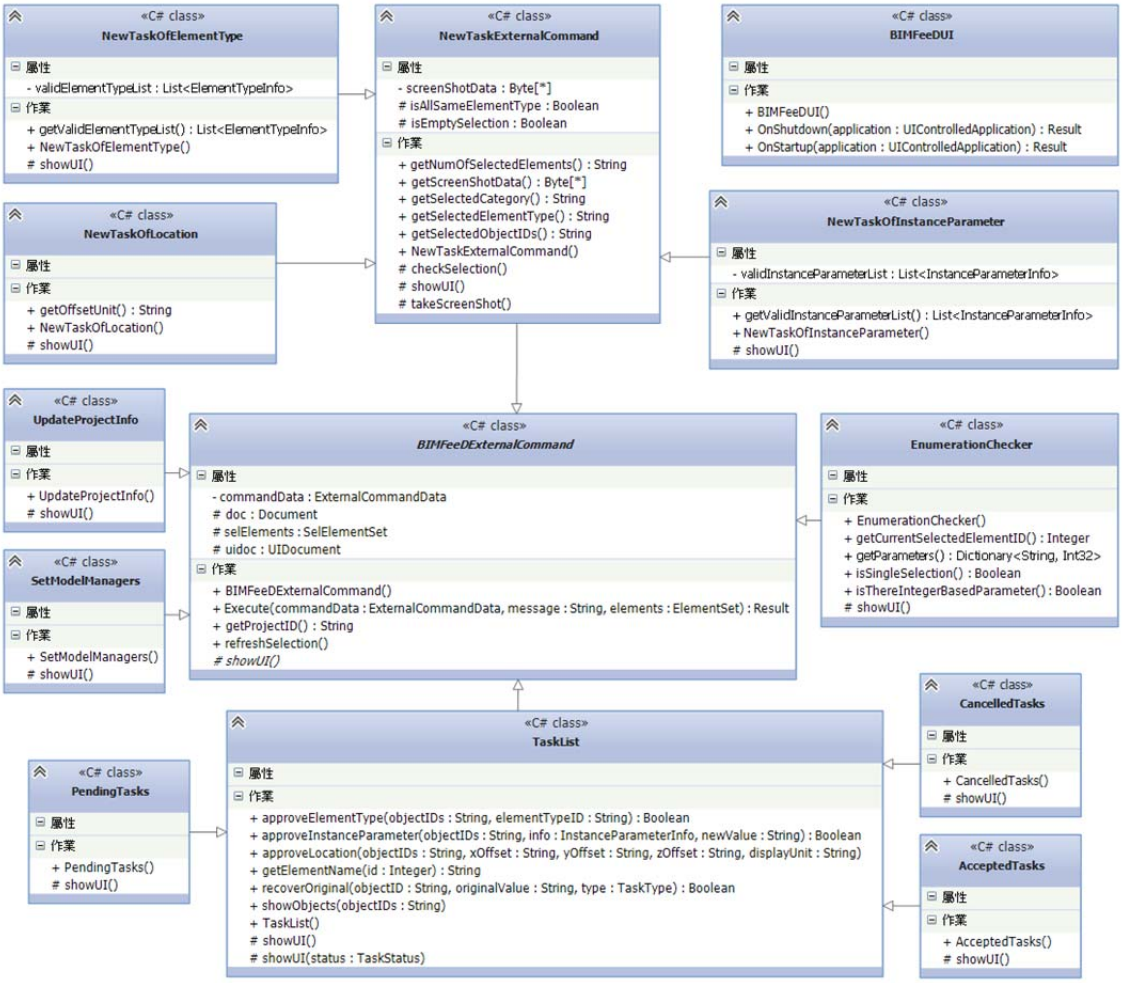


圖 12：BIMFeeD 外掛程式之外部指令 UML 類別圖

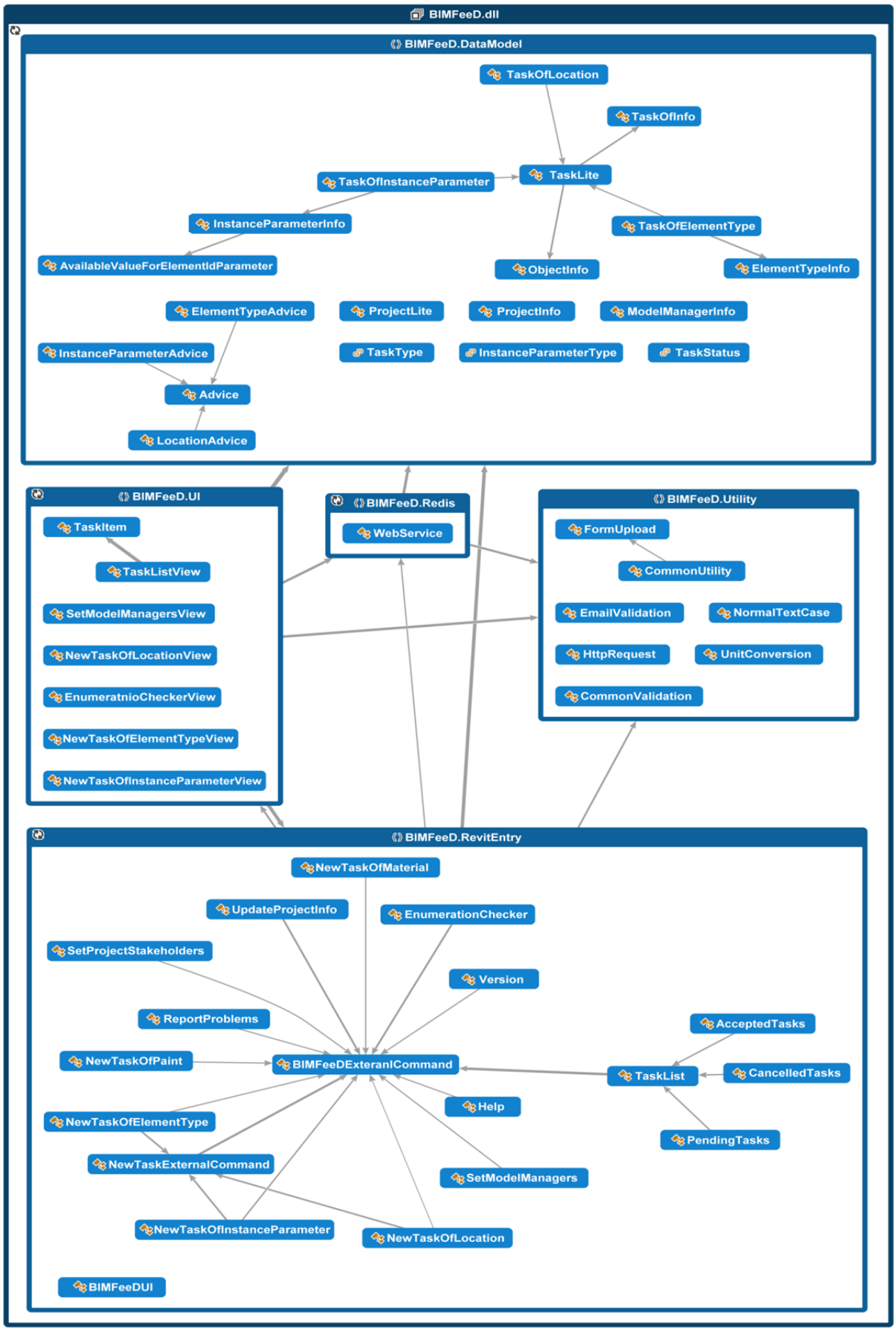


圖 13：BIMFeeD 命名空間及其類別關係圖

4.5 網路應用程式設計

當資訊請求端使用者建立任務之後，BIMFeeD 會發出 email 通知給在任務中被指定的資訊回饋端使用者，接收到通知的使用者可透過 email 中的超連結連至 BIMFeeD 的任務網頁，此任務網頁是由圖 5 中的網頁伺服器所產生，因此必須包含網頁操作模組和模型呈現介面兩部分，網頁操作模組整合模型專案資訊、模型管理者資訊、任務資訊和任務建議等，模型呈現介面在圖 3 中原指基於 IFC 的網頁伺服器，然而 3.2 節已說明將用截圖的權宜之計來取代直接在網頁上瀏覽三維模型，惟 4.6 節仍會實作一個基於 IFC 的網頁模型瀏覽原型系統。

網頁伺服器指的是提供網頁瀏覽或相關服務的電腦設備，而其中的軟體系統應是由網路應用程式所構成，在 3.4 節已提及 BIMFeeD 將利用 Play Framework 作為網路應用程式的開發框架，目前市面上有相當多 web 應用程式框架皆採用 MVC 模式來進行開發，Play Framework 也不例外。

MVC 模式(Model-View-Controller)最早是由 Reenskaug (1979)提出，是一種軟體架構的設計模式，其將軟體系統分為模型(model)、檢視(view)和控制器(controller)三部分，主要目的為增加程式設計的彈性、可擴充性、可維護性和利於專業分工。控制器主要作為模型和檢視之間的中介，負責處理使用者的請求，並操控模型來產生對應的檢視；檢視主要作為使用者介面的功能，負責產生控制器所驅動的圖形化使用者介面；模型主要作為資料管理或資料庫的介面，負責存取和處理軟體系統的資料邏輯。圖 14 顯示 MVC 模式的運作方式。

Yang and Zhang (2006)在設計基於 IFC 的建築物件資訊系統時，同樣採用 MVC 模式來設計其網頁伺服器系統，其中網頁伺服器會先辨別 HTTP request，並轉送至該 URI 路徑所對應的控制器，控制器負責處理對於物件資訊的請求(request)，並操控模型以取得相關資料，最後則產生相應的檢視，即呈現具備物件資訊的使用者介面於網頁瀏覽器。

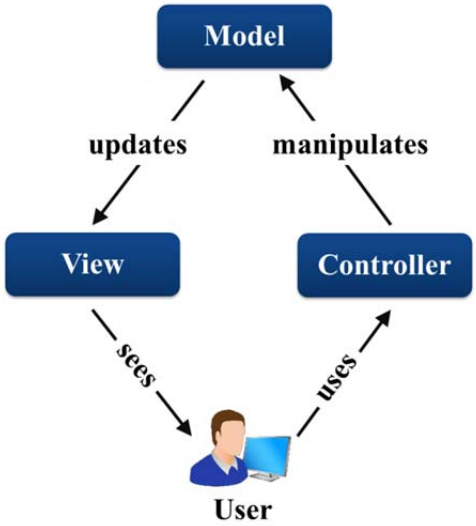


圖 14：MVC 模式的運作方式

BIMFeeD 的網頁伺服器和上述研究的運作流程相似，因此接著以路由(routing)、控制器、模型和檢視的順序分別說明：

- **路由**

  早期的網頁伺服器大多直接對應檔案的目錄結構，然而大部分採用 MVC 模式的 web 應用程式框架皆具備自有的路由功能，且幾乎都採用 RESTful 設計架構，即開發者可指定任意的 URL 位址與任意的控制器對應，而檢視則主要透過樣板系統來產生，如此可獲得極高的設計彈性。因此，當資訊回饋端使用者連接 BIMFeeD 網頁伺服器時，Play Framework 會先辨別 HTTP request 之 HTTP 方法和 URL 位址，並呼叫對應的控制器動作(action)或方法(method)。BIMFeeD 的路由設計如表 4 所示。

表 4：BIMFeeD 路由設計

HTTP 方法	URL 位址	控制器方法
GET	/	controllers.Application.index()
GET	/lite/images/tasks/:file_name	controllers.Application.getImage(file_name: String)
GET	/lite/:taskID	controllers.Application.getTaskLite(taskID: String)
POST	/lite/:taskID	controllers.Application.setTaskLite(taskID: String, recipients: String, modelManagerEmail: String)

- **控制器**

Play Framework 的控制器必須是一個命名空間為 controllers、繼承自 Controller 類別的 Java 子類別，根據表 4 的設計，當資訊回饋端使用者在瀏覽器網址列鍵入/lite/images/tasks/:file_name 的 URL 位址時，表示 HTTP request 為 GET 方法，故路由系統會將此 HTTP request 對應到控制器的靜態(static)方法，即 Application 類別的 getImage 方法，其中:file_name 代表一個型別為 String 的控制器方法引數，此控制器主要負責呈現模型的截圖。

- **模型**

Play Framework 的模型必須是一個命名空間為 models 的 Java 類別，此類別負責處理控制器所轉發的資料處理要求，例如：當 URL 位址為/lite/:taskID 的 GET 方法被執行時，Application 控制器會將 taskID 這個變數傳送給 getTaskLite 靜態方法，而在此靜態方法中必須呼叫對應的模型類別，透過該模型類別的方法來取得所需的資料(即請求資訊)，因此模型類別的方法主要負責向 Redis 資料庫進行 CRUD 操作，並將資料處理結果回傳至控制器，再由控制器產生(render)一個 HTML 頁面的檢視，此頁面包含任務資訊、模型截圖、網頁操作模組等，以傳送至瀏覽器供資訊回饋端使用者觀看或操作，此時 HTTP 狀態代碼也一同被回傳至客戶端。

- **檢視**

Play Framework 的檢視是一個基於 HTML 的樣板系統，當控制器透過模型取得所需資料後，將所取得的資料或整個模型類別傳送至樣板系統，則樣板系統會產生相應的 HTML 頁面，伺服器再將此頁面傳送至客戶端瀏覽器呈現，例如：當資訊回饋端使用者在網頁操作模組提供模型修改建議時，其會送出一個 HTML 的表單(form)，此時 URL 位址為/lite/:taskID 的 POST 方法會被執行，故 Application 控制器的 setTaskLite 靜態方法會被呼叫，在

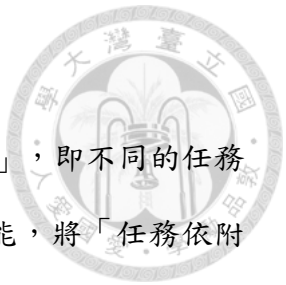
此靜態方法中會將回饋資訊透過對應的模型類別方法傳送至Redis資料庫，並在模型類別方法執行完成後，將更新的資料傳送至指定的樣板系統，由樣板系統 render 一個新的 HTML 頁面，當中包含原本的請求資訊、模型截圖、網頁操作模型以及此次新增的回饋資訊(即建議)等。

4.6 專業版系統功能之需求分析

如同 4.1 節所述，本研究實作的 BIMFeeD 入門版主要目的為凸顯資訊回饋至 BIM 模型的運作機制，而專業版的規劃則旨在引進其他的輔助工具或配套措施，以讓 BIM 資訊交換的流程和 BIMFeeD 所能發揮的功能更加完善，畢竟 BIM 技術在全生命週期的應用本來就有需要整合不同的領域知識(domain knowledge)、使用工具和工作流程(workflow)，以下用條列的方式來補充十個可讓 BIMFeeD 更加健全的專業版功能需求：

- **專案管理系統之整合**

  因為在生命週期中會使用許多 BIM 工具，包含設計工具、分析工具、規範檢核工具等，Singh et al. (2011)提出以模型庫(BIM server)作為多領域(multidisciplinary)協同作業平台的理論架構，從模型管理、設計審查、資料安全、模型庫建置環境等不同層面來分析協同作業平台的技術需求。由此可見，專案管理系統對於 BIM 協同作業而言是基本而必要的(essential)，但專案管理系統涉及的技術層面和研究議題太廣泛，上述研究也僅是以需求分析的觀點來提出理論架構，因此 BIMFeeD 若能站在巨人的肩膀上，以現有的商用軟體或學術研究進行整合，或許也是可行的出路。商用軟體如 Bentley 的 ProjectWise，其自稱為「工程資訊管理和專案合作軟體」；學術研究如郭榮欽 et al. (2013)的「BIM 工程實作雲端服務平台」，其利用 VDI 技術將協同作業平台在雲端環境佈署，將是未來的趨勢。以下九個功能需求亦必須以專案管理系統為基礎，始能達到較佳的協同作業效率。



- **任務依附於(attach to)模型物件**

本研究開發之 BIMFeeD 入門版是「模型物件依附於任務」，即不同的任務可能涉及同一個模型物件，若能做到類似交互參照的功能，將「任務依附於模型物件」的關係也一併建立，對於衝突排除和協同作業將更有效率。

- **即時任務或建議通知**

當有新的任務或建議產生時，本研究開發之 BIMFeeD 入門版是以電子郵件來通知相關人員，若能在專案管理系統或建模工具中直接建置一個訊息通知系統，則能更即時地將問題反映予相關人員。

- **定時自動提醒**

一個模型專案隨生命週期發展，或許會有非常多的任務待處理，當中又有輕重緩急之分，或能配合專案管理系統或排程系統(如：Google Calendar)，設置定時提醒之訊息通知，則待辦任務和事項較不易被忽略。

- **以 OAuth 輔助權限管理**

如同 3.4 節所述，OAuth 為開放授權的概念，即透過服務提供方的認證機制來進行權限管理，只要使用者同意開放授權，則專案管理系統或 BIMFeeD 皆不需要管理使用者的帳號、密碼等使用者資料，同時又可透過服務提供方取得使用者的基本資料(如：姓名、email 等)，圖 15 為應用 Google OAuth 的範例，使用者可利用其 Google 帳號登入，以在線上瀏覽 IFC 模型。

- **回饋資訊之檔案上傳**

回饋資訊即 BIMFeeD 系統中的建議，資訊回饋端使用者有時較難以文字直接回應資訊請求端所發出的任務資訊，故可能會需要上傳一些檔案(如：圖片或規範)來輔助說明，以讓資訊請求端之模型管理者更加了解資訊回饋端使用者的建議內容。

圖 15：Google OAuth 應用範例

- **IFC 模型線上瀏覽功能**

在 3.2 節已說明基於 IFC 的網頁伺服器將以螢幕截圖的方式來替代，主要原因是模型線上瀏覽的效能不夠健全，尚有一些技術瓶頸需要克服，如 Rasys et al. (2014)的研究也提到此情況。本研究仍實作一個以 IFC 標準為基礎的線上模型瀏覽功能，此原型系統係以 WebGL 技術來開發，由於 WebGL 可跨平台又不需安裝任何外掛程式，是近年來較受期待的網路 3D 圖形呈現技術 (Wei and Ma, 2012)，在利用圖 15 的 Google OAuth 登入之後，本研究設計了一個網頁介面，當使用者點選模型物件時可檢視其可供修改的參數化屬性，如圖 16 所示，有關 WebGL 的技術實作細節將待 5.1 節再補充說明。

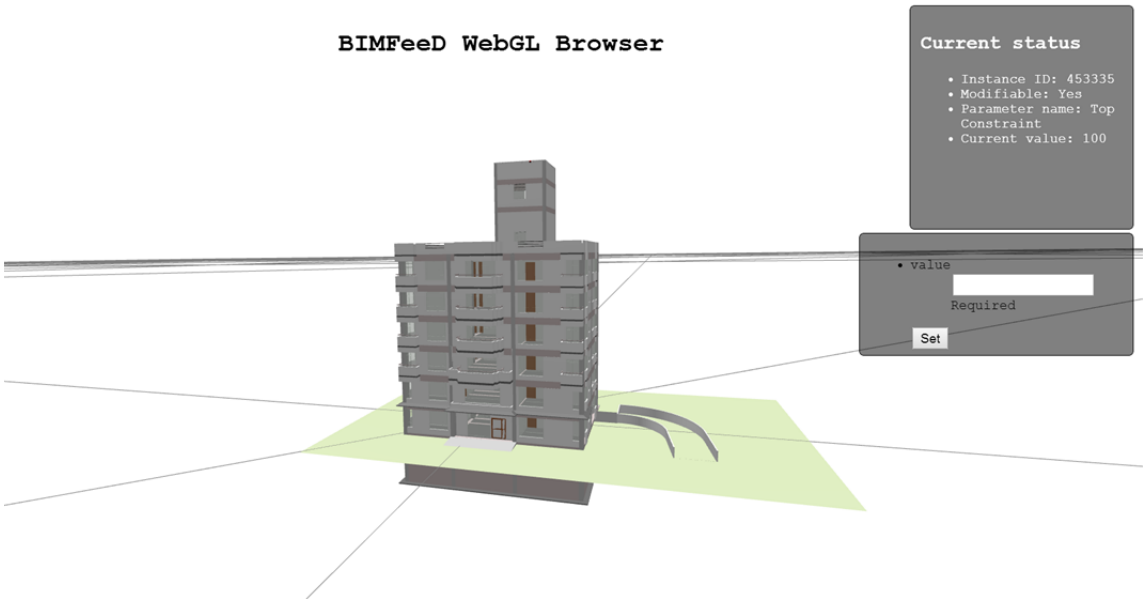


圖 16：以 WebGL 技術開發之 IFC 模型線上瀏覽功能

- **截圖數量與標註功能**

本研究開發之 BIMFeeD 入門版僅能將模型視圖區域截圖一次，若能讓使用者調整不同的模型視圖和觀看角度，並同時產生不同截圖和在其上進行標註，對於傳達任務資訊的清晰程度將更有助益。

- **任務資訊之查詢和檢索**

一個模型專案隨生命週期發展，或許會有非常多的任務，即便 BIMFeeD 將任務分為三種狀態，但仍會有其他的分類或檢索之需求，故若能加入基本的文字檢索或條件查詢功能，對於任務的管理將更有助益。

- **Web 服務 API 對外開放**

本研究開發之 BIMFeeD 入門版並對外開放資訊回饋資料庫系統的 web 服務 API，若爾後能將 web 服務的安全性驗證機制環境建置好，web 服務 API 便可對外開放，對於系統開發者而言將更有應用彈性，並不一定要透過 BIMFeeD 的建模工具外掛程式或網路應用程式來建立和管理任務，如同圖 3 所描述的各種資訊回饋途徑或方法，皆可透過 web 服務 API 來利用資訊回饋資料庫系統，可根據實際需求將使用者介面或應用流程客製化，同時又能利用 BIMFeeD 的資訊交換機制。

第五章　系統實作與展示

5.1　系統實作工具與技術

本節主要介紹 BIMFeeD 在系統實作階段所採用的工具和技術。有關物件導向的軟體架構設計，三層架構(three-layer or three-tier architecture)是常見的軟體分層方式，目的為將不同屬性的軟體功能或構成要素區分開來，這三層架構包含：

- **展示層(presentation layer)**：前端(front-end)使用者之操作介面。
- **商業邏輯層(business logic layer)**：軟體作業流程控制及相關資料處理。
- **資料存取層(data access layer)**：透過通訊管道或資料庫存取資料。

(Mishra, 2011)

基本上，這和 4.5 節所提及之 MVC 模式相當類似，惟三層架構主要闡述軟體功能取向的分層概念，MVC 則是軟體運作的設計模式。本研究考量 BIMFeeD 具有不同的使用者平台和資訊管理模式，特別將客戶端層(client layer)和資料層(data layer)也一併考慮進來，並將 BIMFeeD 之軟體分層技術架構圖繪製如圖 17，雙向箭頭代表技術之間的資訊傳輸或程式互動，單向箭頭代表檔案格式的轉換步驟，接著逐層介紹其中的工具、技術或操作環境：

- **客戶端層**

  客戶端層主要顯示使用者之操作工具或環境，對於資訊請求端的使用者而言，主要操作工具即為建模工具 Revit，再透過 Revit 啟動 BIMFeeD 之建模工具外掛程式，而資訊回饋端的使用者則使用網頁瀏覽器來連接 BIMFeeD 之網頁伺服器，若欲執行 4.6 節所述之 IFC 模型線上瀏覽功能，則網頁瀏覽器必須支援 WebGL，目前較新版本的幾個主流網頁瀏覽器皆已廣泛支援 WebGL，甚至部份行動裝置的瀏覽器亦可支援 WebGL。

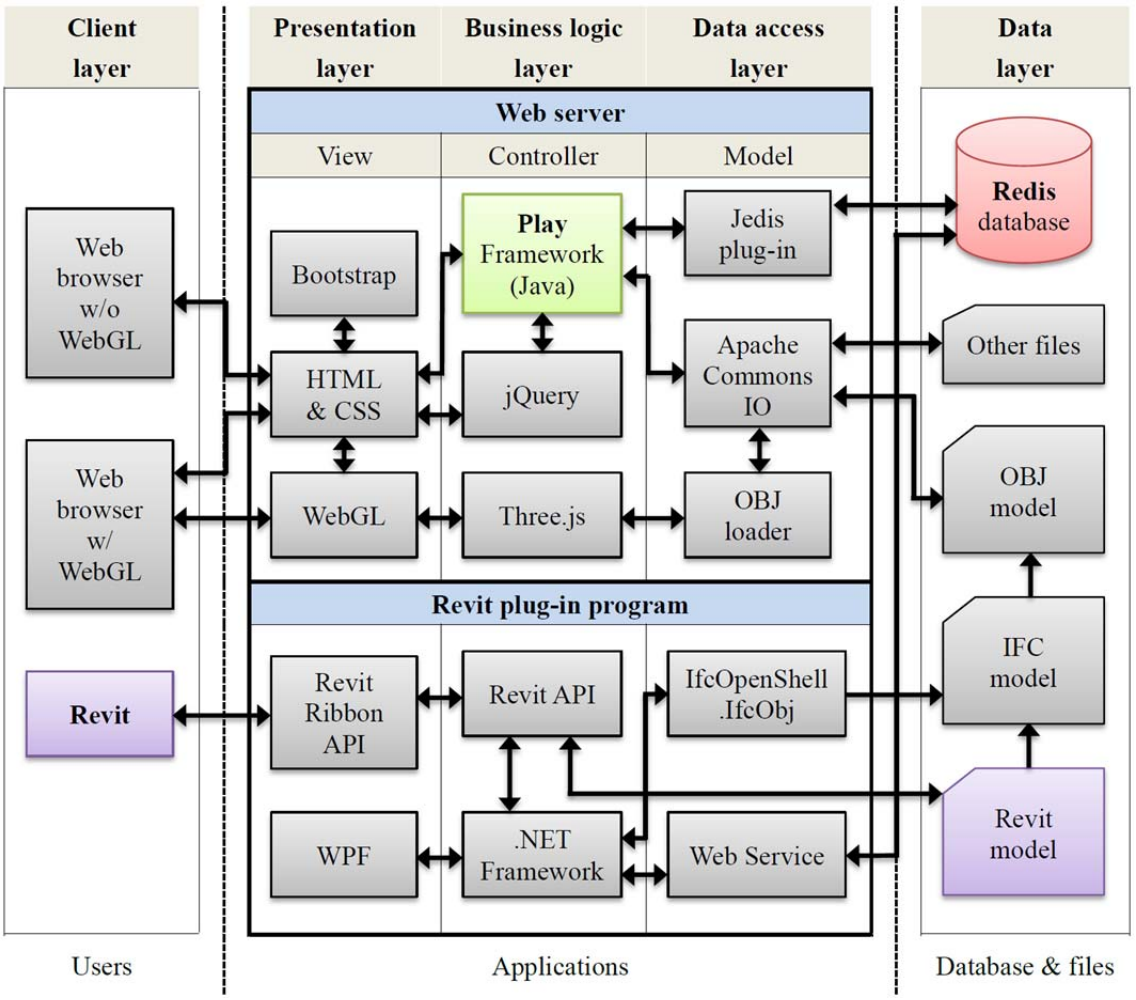


圖 17：BIMFeeD 之軟體分層技術架構圖

- **展示層(網頁伺服器部分)**

網頁伺服器的展示層主要對應到 MVC 模式的檢視，網頁的內容呈現主要以 HTML 為基礎，搭配 CSS (Cascading Style Sheets)作為樣式表，而前端的設計樣板(design template)和使用者介面則採用支援響應式設計的 Twitter Bootstrap 框架，可節省自行設計網頁元件和編排頁面的時間。此外，IFC 模型線上瀏覽功能採用 WebGL 技術，即使用 HTML5 的 Canvas 元素來進行圖像渲染(render)。

- **展示層(外掛程式部分)**

外掛程式的展示層主要分為兩部分：其一為 BIMFeeD 外掛程式在 Revit 使用者介面中的功能區，即藉由 Revit Ribbon API 來設計 BIMFeeD 在 Revit

使用者介面中之頁籤(tab)、面板(panel)和按鈕(button)等；其二為透過按鈕來啟動 BIMFeeD 不同功能之使用者介面(如：建立任務、檢視任務清單等)，此部分採用 WPF (Windows Presentation Foundation)技術來進行使用者介面的開發，WPF 係 Mircrosoft .NET Framework 之展示層開發框架。

- **商業邏輯層(網頁伺服器部分)**

網頁伺服器的商業邏輯層主要對應到 MVC 模式的控制器，由於 BIMFeeD 的網頁伺服器採用 Play Framework 來開發，故伺服器端(server-side)的程式邏輯主要由 Java 的控制器類別來處理，而客戶端(client-side)的程式腳本則採用 jQuery，係一套跨瀏覽器的 JavaScript 函式庫。此外，由於 WebGL 僅提供底層(low-level)的渲染和計算函式，故本研究另外採用 Three.js，係一套解析 3D 檔案格式和處理 3D 互動行為的 JavaScript 函式庫。

- **商業邏輯層(外掛程式部分)**

外掛程式的商業邏輯層主要使用兩種類別庫：有關 Revit 模型的資訊擷取和物件修改必須基於 Revit API 來設計程式邏輯；而資料結構封裝和其他公用程式部分則使用.NET Framework 之類別庫。

- **資料存取層(網頁伺服器部分)**

網頁伺服器的資料存取層主要對應到 MVC 模式的模型，可分為三個部分：其一為資訊回饋資料庫系統的資料交換，由於 Redis 資料庫和網頁伺服器處於同一個內部網路環境，故直接採用 Redis 的 Java 客戶端 Jedis 較為方便，且 Play Framework 亦有 Jedis 的外掛程式可用；其二為檔案的上傳與後續處理，即包含各式模型檔案和圖片檔案等，由於伺服器為 Linux 作業系統，故採用 Apache Commons IO 類別庫來處理檔案的 IO (Input/Output)；其三為 IFC 模型線上瀏覽功能的檔案格式解析，由於 Three.js 無法直接解析 IFC 格式，故可先轉為 OBJ 格式，再由 Three.js 之 OBJ loader 來解析 OBJ 格式。

- **資料存取層(外掛程式部分)**

  外掛程式的資料存取層主要包含兩部分：其一為與資訊回饋資料庫系統進行資料交換的 web 服務；其二為將 IFC 模型轉為 OBJ 格式，係採用 IfcOpenShell 的 IfcObj 模組。

- **資料層**

  資料層即包含資料庫管理系統和各式檔案，前者為 Redis 資料庫，後者則包含 Revit 模型檔案、IFC 模型檔案、OBJ 模型檔案、模型截圖檔案以及其他必要檔案等。

5.2 外掛程式操作演示

本研究實作的 BIMFeeD 外掛程式指的是建模工具 Revit 的外掛程式，在 Revit 中稱為增益集(add-in)，根據 4.4 節的設計方法和 5.1 節的技術原理，BIMFeeD 外掛程式的操作邏輯可分為兩大步驟：先由 Revit 的 Ribbon 功能區執行 BIMFeeD 的不同功能選項，再由 BIMFeeD 的 WPF 使用者介面來操作相關功能。BIMFeeD 外掛程式的 Ribbon 使用者介面如圖 18 所示，接著針對各功能和其使用者介面逐一說明操作邏輯和使用方式。

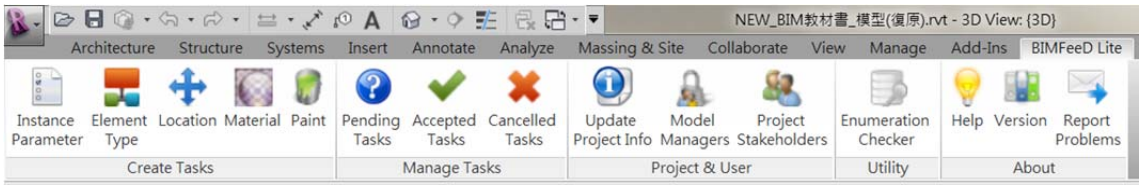


圖 18：BIMFeeD 外掛程式之 Ribbon 使用者介面

圖 18 顯示 BIMFeeD 外掛程式在 Revit 中的主要功能介面，在此 Ribbon 使用者介面中，包含一個名為 BIMFeeD Lite 的頁籤，以及五個不同功能屬性的面板，以下針對五個面板以條列式說明：

- **建立任務(Create Tasks)面板**：如 4.4 節所述，在此面板中有五個按鈕，分別可建立不同任務類型的任務，惟 Material 和 Paint 在 Lite 版本並未實作。

- **管理任務(Manage Tasks)面板**：在此面板中有三個按鈕，分別可啟動不同任務狀態的任務管理介面。
- **專案與使用者(Projects & Users)面板**：在此面板中有三個按鈕，其功能分別為：Update Project Info 功能可將專案資訊同步至 BIMFeeD 的資訊回饋資料庫系統、Model Managers 功能可設定管理此模型專案的模型管理者資訊、Project Stakeholders 功能則計畫與專案管理系統整合而尚未實作。
- **公用程式(Utility)面板**：此面板僅有一個按鈕，即 Enumeration Checker，如 4.4 節所述，本功能可讓使用者檢視實作元件參數的列舉值。
- **關於(About)面板**：此面板主要提供 BIMFeeD 的應用程式基本資訊，使用者可檢視 BIMFeeD 之操作手冊、版本資訊，或者是回報問題。

在開始建立和管理任務之前，資訊請求端的使用者可先設定模型管理者資訊，如圖 19 所示，並且將此模型的專案資訊上傳至資訊回饋資料庫系統，如圖 20 所示，則資訊回饋端使用者可在網路應用程式檢視模型管理者資訊和專案資訊。

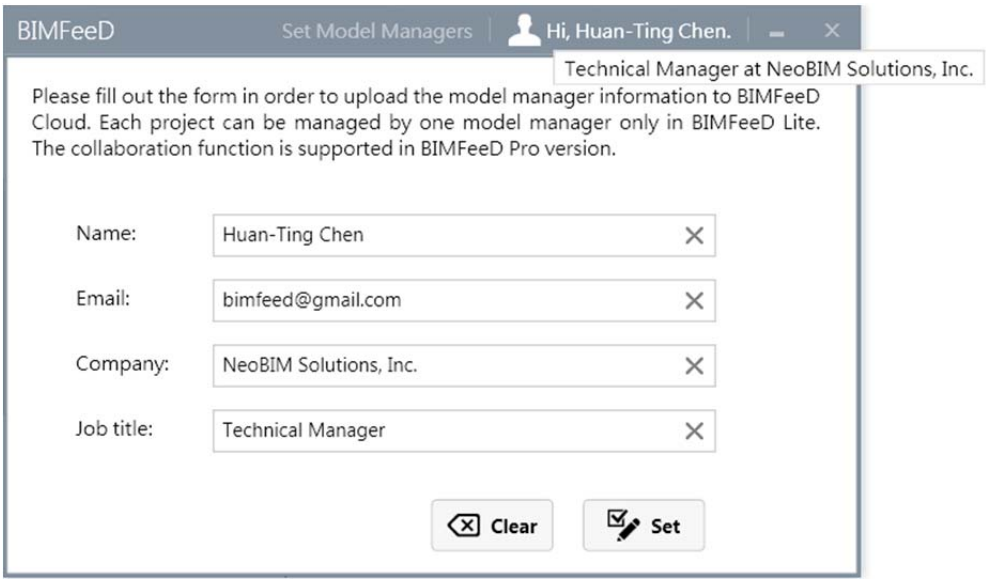


圖 19：設定模型管理者資訊

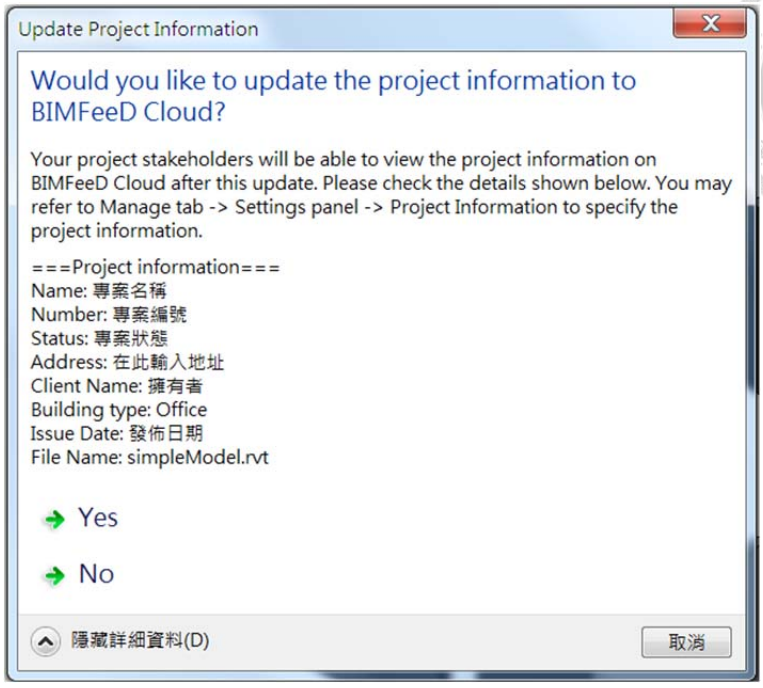


圖 20：上傳專案資訊至資訊回饋資料庫系統

接著，資訊請求端的模型管理者可透過 BIMFeeD 外掛程式來建立任務，並將任務指定給不同的任務接收者(task recipients)，建立任務的使用者介面基本上大同小異，如圖 21 至圖 24 為在 Revit 中建立任務的操作介面，左上半部為模型截圖預覽，右上半部為所選取元件的基本資訊，中間部分則可輸入任務名稱、任務接收者之 email 和任務描述，左下半部為兩個核取方塊(checkbox)，可選擇建立任務之後是否要啟動瀏覽器以連接 BIMFeeD 網路應用程式，以及選擇是否將任務通知發出副本給模型管理者，右下半部則為清除按鈕和送出按鈕，分別可將已輸入內容清除和將任務分派出去。

在 Lite 版本中，任務又分為三種不同的任務類型，圖 22、圖 23 和圖 24 分別為建立這三種任務的操作介面，接著說明建立這三種不同任務類型之差異。

- **建立 Instance Parameter 任務：**圖 22 為建立 Instance Parameter 任務之操作介面，使用者可勾選與此任務相關的實作元件參數，以讓資訊回饋端使用者透過 BIMFeeD 網路應用程式指定新的實作元件參數值。

- **建立 Element Type 任務**：圖 23 為建立 Element Type 任務之操作介面，使用者可勾選與此任務相關的元件類型，以讓資訊回饋端使用者透過 BIMFeeD 網路應用程式指定新的元件類型。
- **建立 Location 任務**：圖 24 為建立 Location 任務之操作介面，使用者可勾選 X、Y 或 Z 方向的偏移，以讓資訊回饋端使用者透過 BIMFeeD 網路應用程式指定所選取元件在 X、Y 或 Z 方向的偏移量數值。

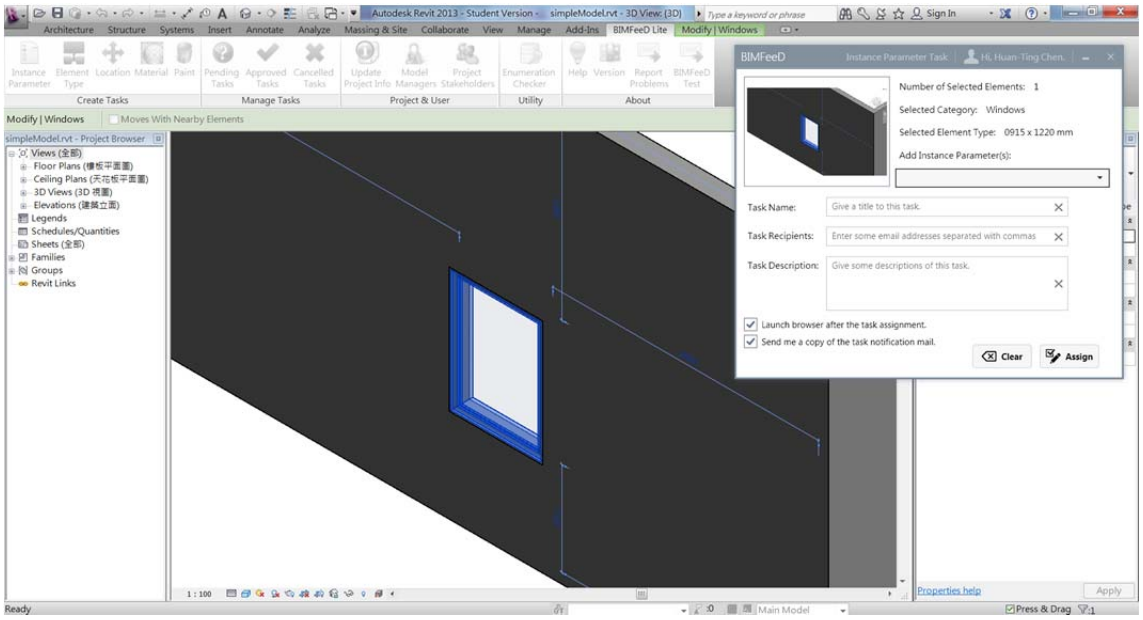


圖 21：在 Revit 中執行 BIMFeeD 外掛程式之建立任務畫面

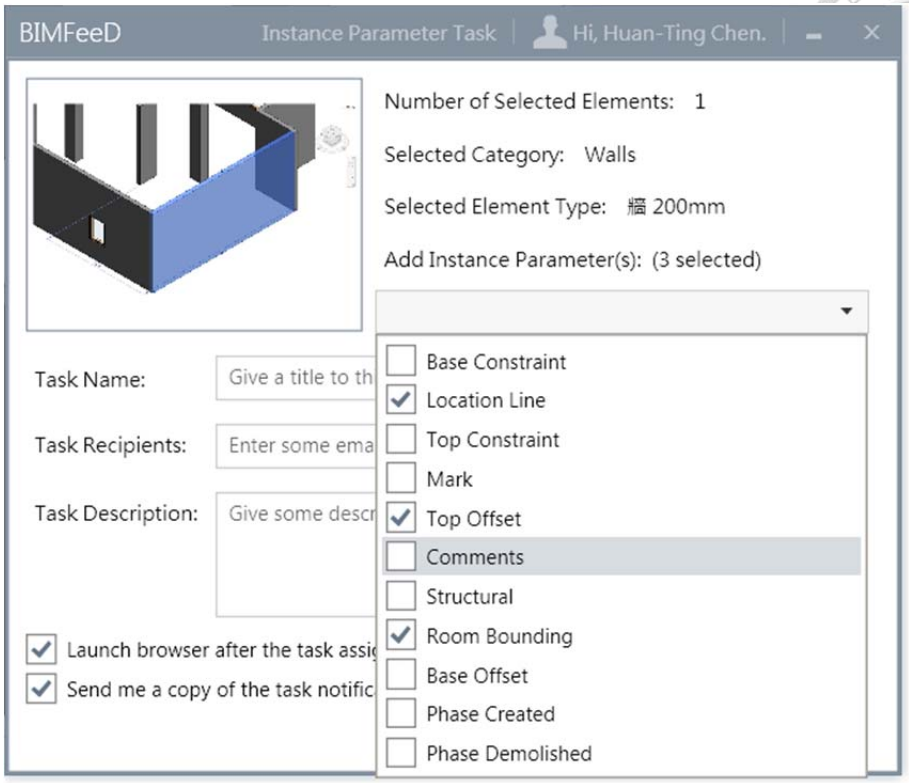


圖 22：建立 Instance Parameter 任務之操作介面

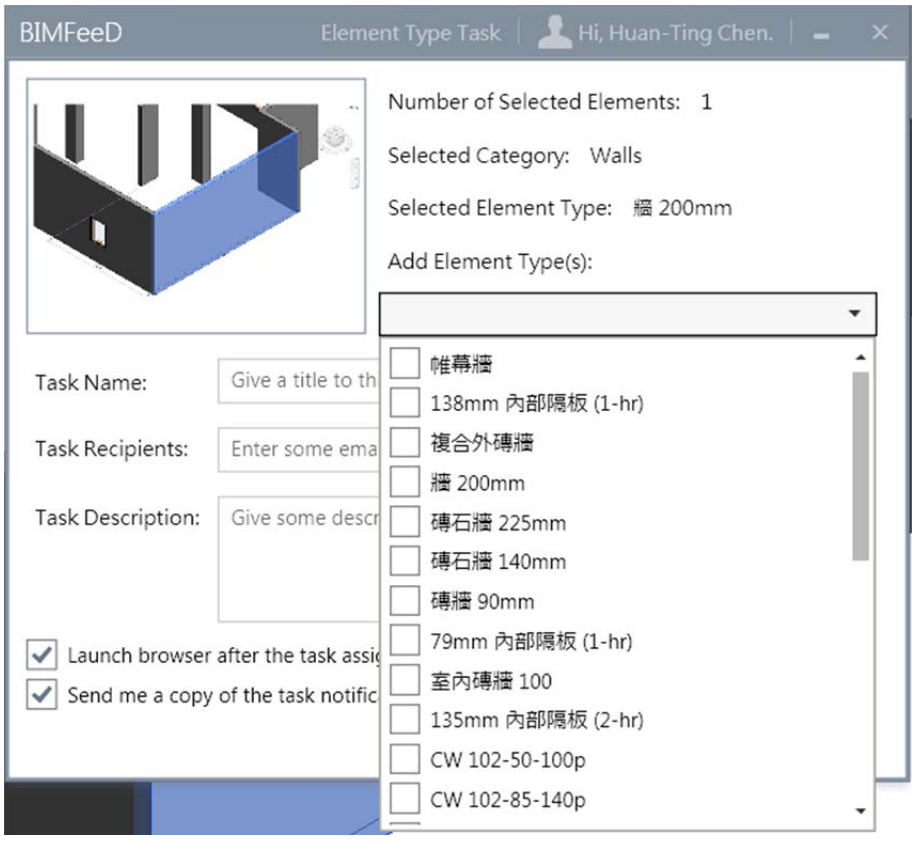


圖 23：建立 Element Type 任務之操作介面

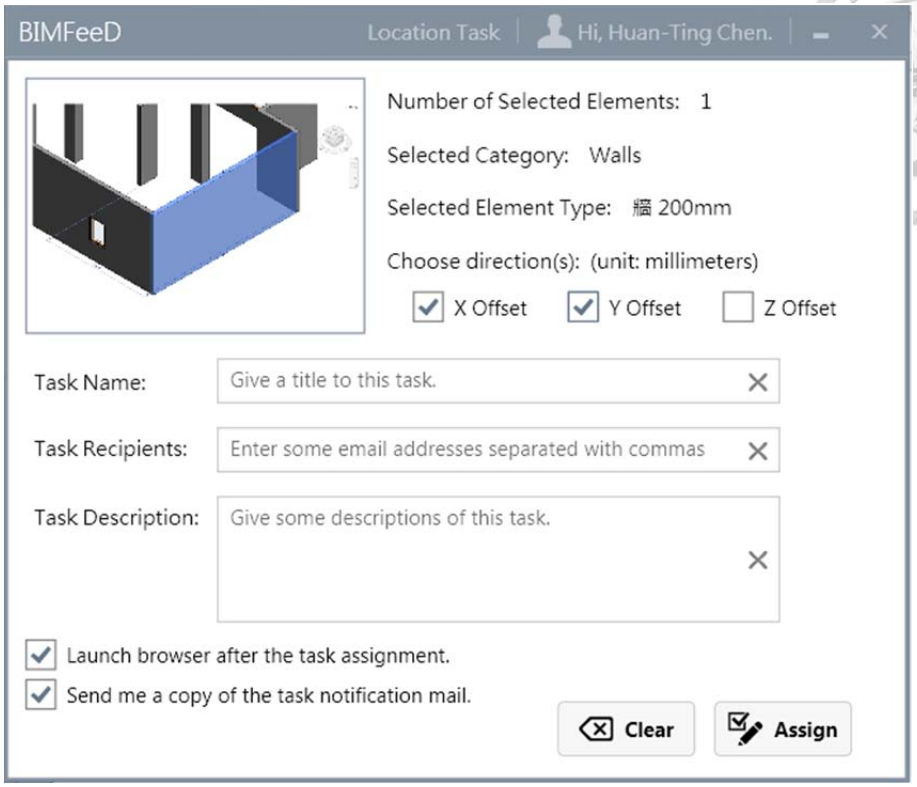


圖 24：建立 Location 任務之操作介面

除此之外，在建立 Instance Parameter 任務時，若欲選擇的實作元件參數型態屬於列舉類別的整數值，則可利用 Enumeration Checker 來檢視其背後的列舉值，如圖 25 所示，藉由手動設定不同的參數值以檢視其列舉值。

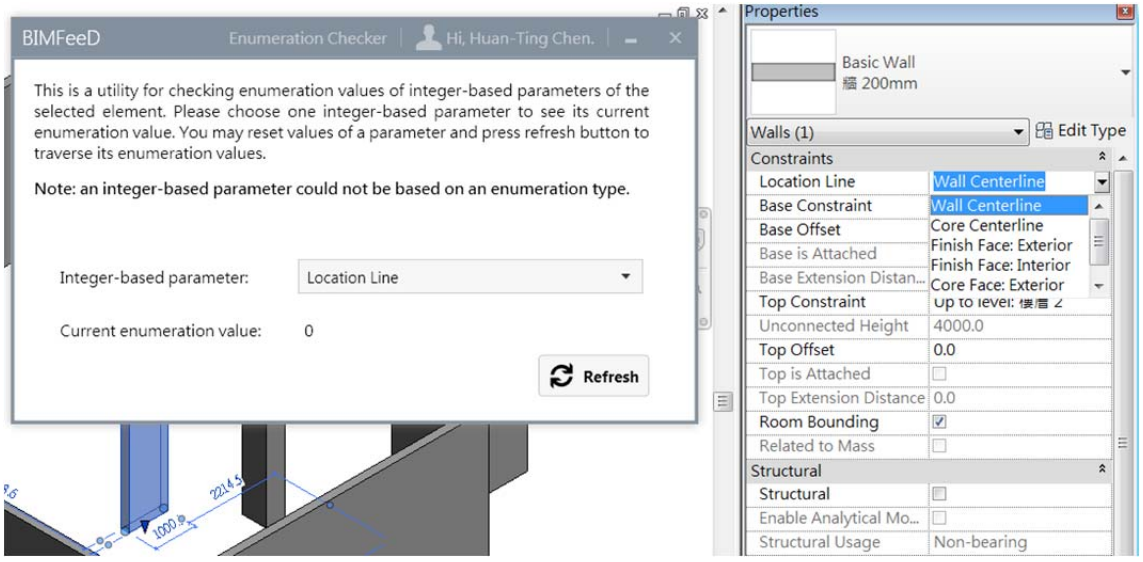


圖 25：Enumeration Checker 操作介面

在任務被建立之後，資訊請求端的模型管理者可透過 BIMFeeD 外掛程式來管理任務，圖 26 為待判任務清單之操作介面，使用者可篩選不同類型的任務，並可從清單檢視任務識別碼、任務名稱、任務建立時間和建議，若某任務已具有資訊回饋端使用者所提供之建議，建議欄會顯示 V，否則為 X，模型管理者可點擊右上角的重新整理按鈕，若資訊回饋端使用者有新的建議則會更新。

圖 26：待判任務清單

當使用者從圖 26 的待判任務清單點擊其中一個任務時，便會出現該任務的詳細資訊和建議，如圖 27 為下拉式建議清單，從此下拉式建議清單可檢視資訊回饋端使用者在不同時間點所提供之建議，模型管理者可從下拉式建議清單來選擇檢視其中一個建議，如圖 28 為某個建議之詳細資訊，其中包含建議內容描述和與此任務相關的修改資訊。

舉例：圖 28 為一個確認窗台高度的任務，模型管理者詢問資訊回饋端使用者窗台高度應為 45 公分或 50 公分，而其中一個資訊回饋端使用者表示應為 55 公分，故其在 BIMFeeD 網路應用程式上輸入 Sill Height 為 55，模型管理者接著可先點選

show object 以在模型中檢視相關的元件，若點選 more details 則會連結至 BIMFeeD 網路應用程式，在經過模型管理者的人為判斷後，其可以決定將此任務 accept 或 cancel，若此任務被接受，則 BIMFeeD 會自動修改與此任務相關的模型元件，即窗台高度會被自動更新為 55 公分，同時將此任務移至接受任務清單，若此任務被取消，則 BIMFeeD 僅會將此任務移至取消任務清單。

當任務被模型管理者接受後，其可開啟接受任務清單，如圖 29 所示，當某個任務被點擊時，該介面將顯示任務詳細資訊和被接受的建議，模型管理者可將此任務狀態重新改為待判或取消。此外，圖 29 右上角有一個匯出的功能，可將被接受的建議清單和其詳細資訊匯出為 CSV 格式。

當任務被模型管理者取消後，其可開啟取消任務清單，其操作介面和圖 29 幾乎相同，故此處不再重複圖示，惟模型管理者可將被取消的任務重新改為待判或接受之任務狀態。

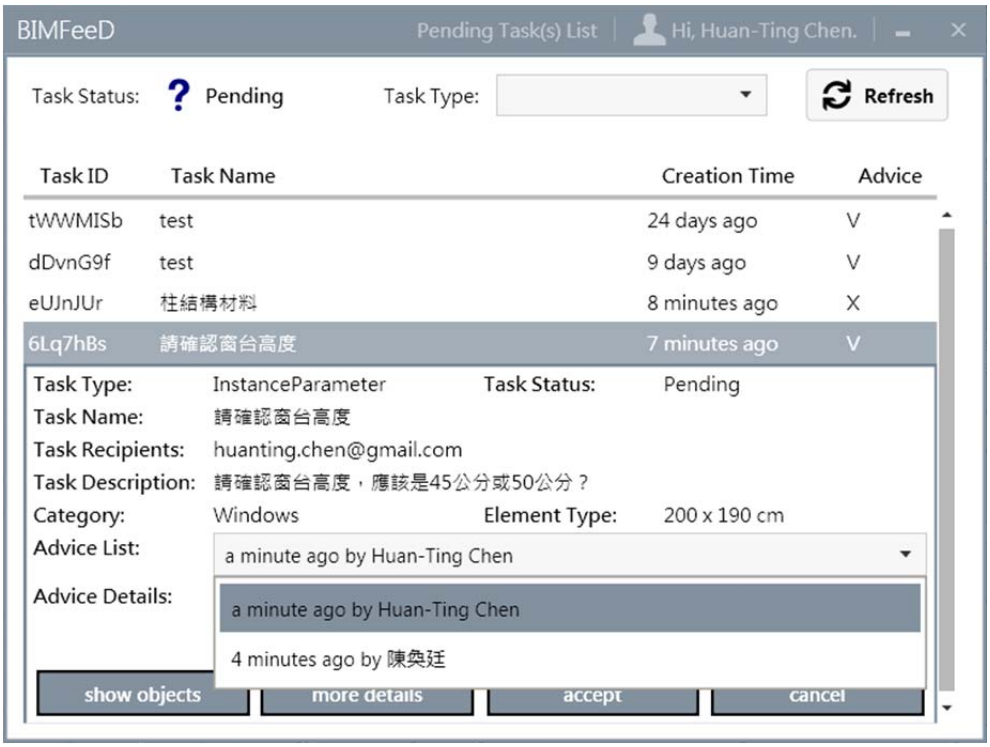


圖 27：待判任務之下拉式建議清單

圖 28：待判任務中某個建議之詳細資訊

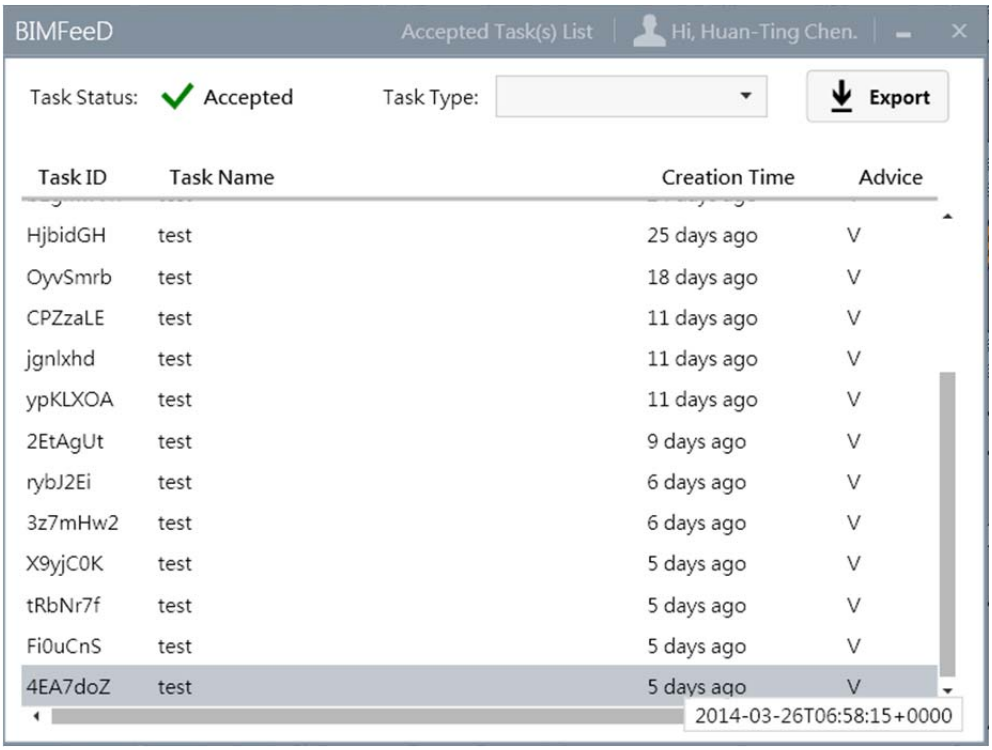


圖 29：接受任務清單

5.3 網路應用程式操作演示

BIMFeeD 的網路應用程式如圖 5 和 4.5 節所描述，主要包含網頁操作模組和模型呈現介面，前者針對不同的任務類型提供不同的回饋介面，後者在 Lite 版本中則以模型截圖的方式來替代。另外，BIMFeeD 的網路應用程式尚須具備發出 email 通知的功能，當任務被建立後必須發出通知給被指定的任務接收者，而當建議被送出後必須發出通知給所有與此任務相關的人員，包含模型管理者、任務接收者和所有曾經給予建議的人員，如圖 30 和圖 31 分別為 BIMFeeD 網路應用程式所發出的任務通知 email 和建議通知 email。

當任務接收者收到任務通知 email 後，即可點擊 email 中的超連結以連結至 BIMFeeD 網路應用程式，由於每個任務皆有獨一無二的任務識別碼，故 BIMFeeD 網路應用程式會產生專屬的操作介面，如圖 32 所示，資訊回饋端使用者可在此網頁介面檢視模型截圖、專案資訊、模型管理者資訊、任務資訊和相關操作說明等，模型截圖可點擊以放大檢視，例如：圖 32 為確認窗台高度的任務，資訊回饋端使用者可直接在此網頁介面上輸入正確的窗台高度參數值。

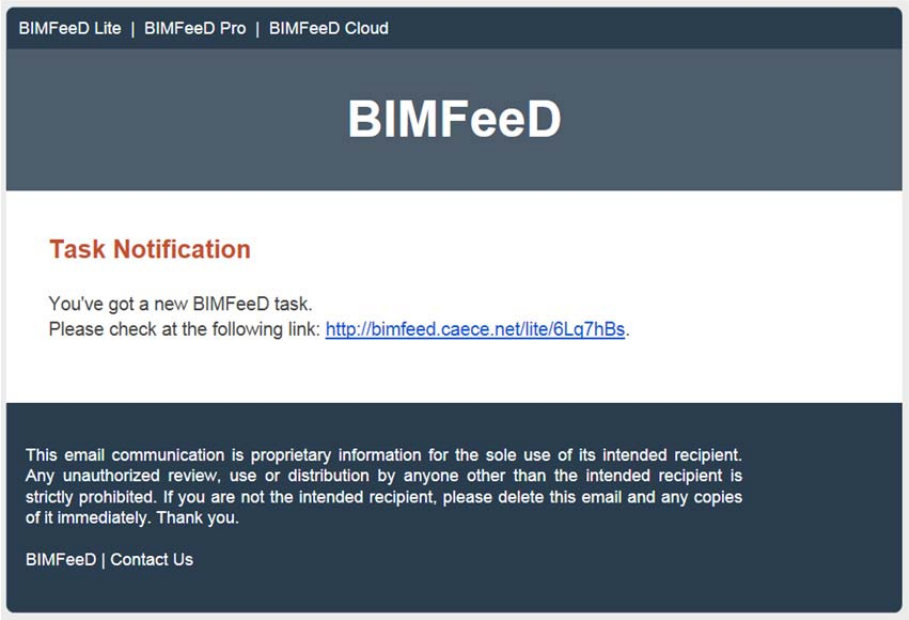


圖 30：BIMFeeD 網路應用程式所發出之任務通知 email

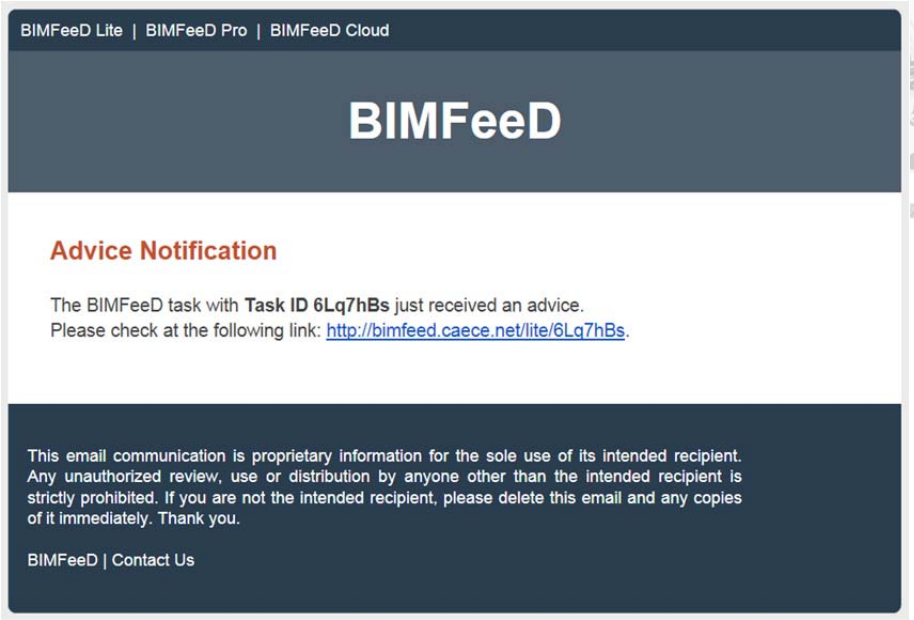


圖 31：BIMFeeD 網路應用程式所發出之建議通知 email

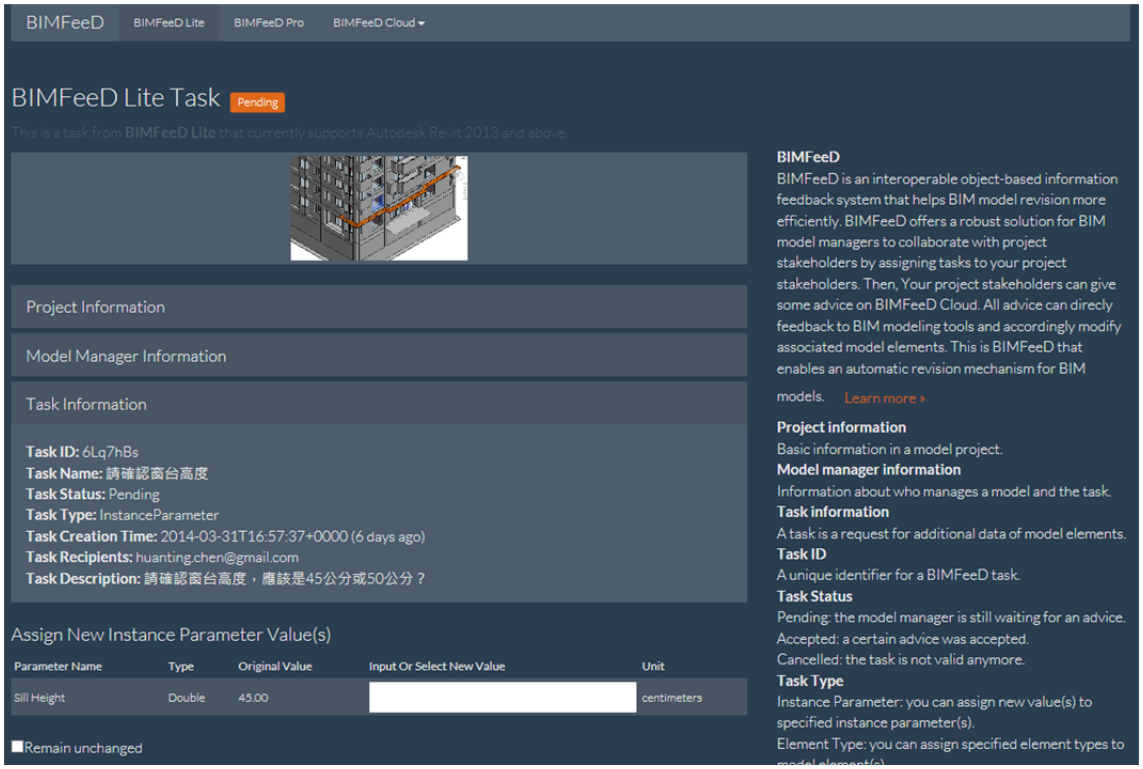


圖 32：BIMFeeD 網路應用程式之主要操作介面

BIMFeeD 網路應用程式會根據不同任務類型而產生不同的網頁操作模組，如圖 33、圖 34 和圖 35 所示，其分別為 Instance Parameter 任務、Element Type 任務和 Location 任務之網頁操作模組，圖 33 之操作介面可輸入或選擇新的實作元件參數值，圖 34 之操作介面可選擇新的元件類型，圖 35 之操作介面可輸入元件位

置在 X、Y 或 Z 方向的偏移量數值。另一方面，若資訊回饋端使用者認為此任務之相關模型元件不需要作任何更動，則可勾選 Remain unchanged。

圖 33：指定新的實作元件參數值之操作介面

圖 34：指定新的元件類型之操作介面

圖 35：指定元件位置在 X、Y 或 Z 方向偏移量之操作介面

當資訊回饋端使用者完成與任務相關的元件修改設定後，其可藉由圖 36 之操作介面來輸入對於此任務之建議內容，同時留下姓名和 email，待此建議送出後，BIMFeeD 網路應用程式會發送此建議通知 email 給所有相關的人員，而資訊請求端的模型管理者則可在 BIMFeeD 外掛程式之任務管理介面檢視此建議和進行後續處理，若資訊請求端的模型管理者已接受某個建議，當此任務網頁被使用者重新整理後，則會顯示建議記錄和已被接受的建議，如圖 37 所示。

圖 36：輸入建議資料之操作介面

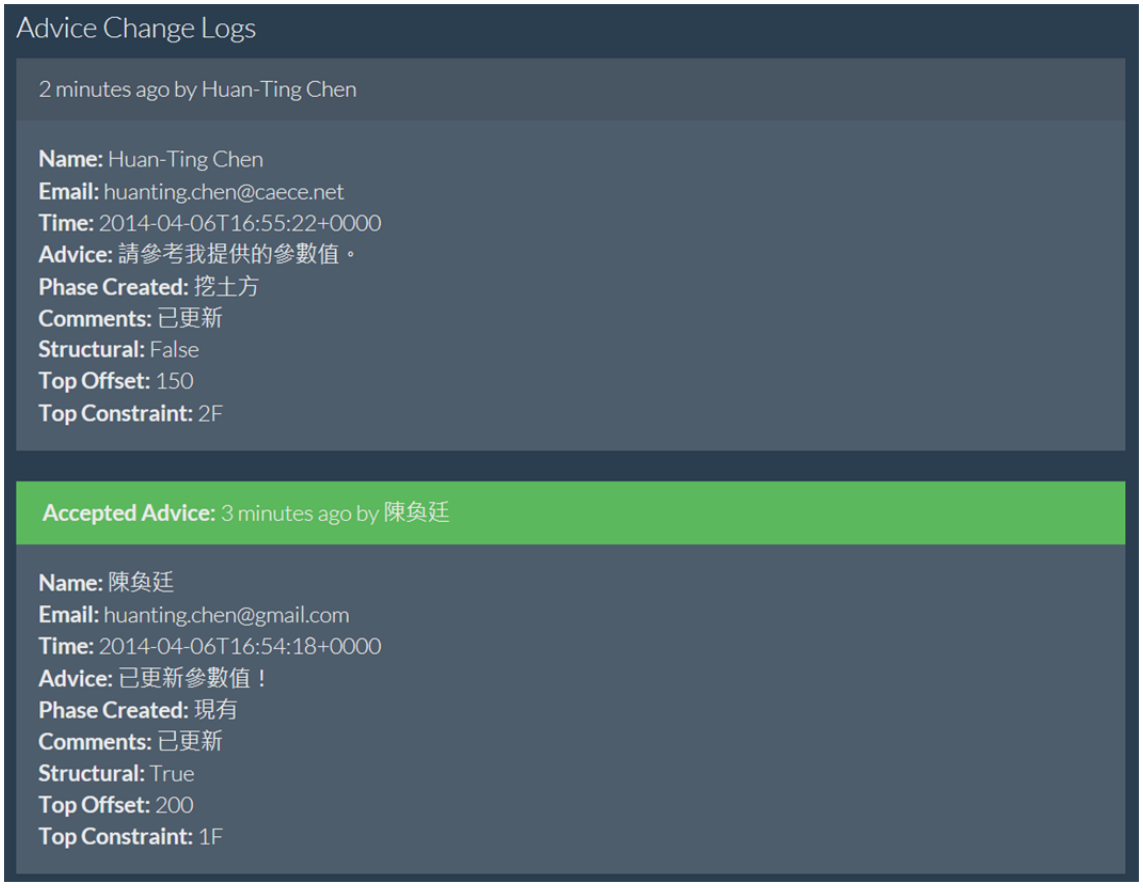


圖 37：任務之建議記錄和被接受之建議

5.4 專家訪談之系統測試與驗證

如同 2.5 節之專家訪談，本研究在系統開發後向同一批專家們徵詢意見，由於 BIMFeeD 屬於較一般化之解決方案，而非針對特定問題或應用情境所設計，故並無進行大量的使用者測試，而是冀望借助專家們應用 BIM 技術多年的經驗，從專家意見來評定 BIMFeeD 能否適用於 2.2 節至 2.4 節所述之十二種應用情境。

系統測試與驗證的方式為：首先向專家們講解 BIMFeeD 之操作邏輯，並示範完整的系統操作方法，再以十二種應用情境逐一詢問專家們之意見。專家意見主要分為三部分：BIMFeeD 是否適用於各應用情境、最適合採用 BIMFeeD 之應用情境和整體改善建議。

整體而言，有關 BIMFeeD 能否適用於十二種應用情境，專家們對於 BIMFeeD 的資訊回饋概念和系統新穎性普遍表示認同，惟實務上有許多複雜度更高的例外狀況，或需額外針對不同生命週期階段的工作流程進行客製化。以下針對專家們認為 BIMFeeD「不適用」或「僅某些情況適用」之應用情境來條列說明，其中應用情境編號與 2.2 節至 2.4 節相同，專家代號則與 2.5 節相同：

- E 認為應用情境(1)需待細部設計定案後才適用，即本研究於 2.1 節所強調之前提，BIM 模型的完整度必須足夠，BIMFeeD 可用於模型修改而非建模。
- E 和 H 皆認為應用情境(2)不完全適用，因為能源分析需耗費時間進行模擬，資訊回饋端並不一定具有能力可直接即時給予模型修改建議。
- A 和 E 皆認為應用情境(3)不適用，因為營造端的 RFI 通常涉及複雜問題或茲事體大，但 A、B、F 和 H 則認為在設計端的 RFI 或可適用，惟 RFI 在不同公司或有不同的名稱，例如：F 和 H 任職之顧問公司稱為工作聯繫單。
- A 認為應用情境(4)不適用，因為模型重新建置或整合的複雜度太高。
- F、G、H 和 I 皆認為應用情境(5)不適用，因行政機關或業主並不完全了解工程專案的細節，故無法承擔模型修改之責任。

- F 和 H 皆認為應用情境(7)不適用，因為 F 和 H 所任職公司之軟體資源較不虞匱乏，這和 2.5 節敘述 A 認為無此需求之理由相同。
- F 和 H 皆認為應用情境(10)不適用，因其認為物業管理團隊的資訊請求端和資訊回饋端係屬同一性質的工作人員，故協同作業的需求較低。
- E 認為應用情境(11)不完全適用，因其認為現場量測數據透過 BIMFeeD 之任務和建議來修改 BIM 模型元件時，並不一定會比其他方式(如：紙本抄寫)的效率更高。
- E 認為應用情境(12)不完全適用，若能搭配影片錄製效果較好，而非單純只有模型視圖的螢幕截圖。而 F 和 H 則認為應用情境(12)不適用，因為建模初學者經常有很多問題，若以任務的型式來呈現，建模教學者必須提供的建議數量便會相當多而不堪負荷。
- B 和 C 認為在某些較單純化、牽涉層面較少的情況下，所有的應用情境或可在某部分情況下適用。

就上述專家意見而言，由於每位專家來自不同專業領域，專家意見的一致性並不是很高，BIMFeeD 在各種應用情境下的適用性也存在不同的意見，若直接詢問專家們「BIMFeeD 最適用於哪些應用情境？」，根據其所選擇最適用的應用情境，專家們的共同交集也不是很高，顯見 BIMFeeD 若欲符合各應用情境的一般化情況仍有更加完善的空間，必須針對不同應用情境再更深入設計符合其實務工作流程的操作邏輯和功能，表 5 為專家們認為 BIMFeeD 最適應用情境之統計表。

表 5：BIMFeeD 最適應用情境之統計表

專家代號	領域	1	2	3	4	5	6	7	8	9	10	11	12
A	營造端	●	●				●						
B、C										●	●	●	●
E										●	●		
D	設計端						●						
F、G、H、I		●	●	●	●								

在整體改善建議方面，茲條列如下：

- A 認為本研究對於資訊請求和資訊回饋的思考模式是正確的，但其建議在網路應用程式上將模型修改結果即時反映予資訊回饋端參考。
- B 和 C 認為本研究在營運維護階段的應用情境較有發揮空間，BIMFeeD 可作為資訊的更新和補充，就資訊交換的需求而言，由於不同專業領域的語意資訊不同，故模型資訊若能進一步包裝，即讓使用者自訂參數化屬性的群組名稱或口語化名稱，並制定如 IDM 的資訊交付清單，將有助於降低資訊請求端和資訊回饋端雙方的溝通隔閡。
- D 認為 BIMFeeD 可作為現有工作流程的輔助工具，即原有工作流程不需改變，而 BIMFeeD 可在有需求時和原有工作流程並行使用。
- E 認為生命週期越後面的階段不確定性越低，故 BIMFeeD 可發揮的空間較大，若能加強網路應用程式的視覺化功能(如：多張截圖一起比較、部分模型修改的即時展示)，有助於使用者透過模型視圖作為溝通媒介。
- F、G、H 和 I 認為 BIMFeeD 較適用於資訊請求端和資訊回饋端雙方存在資訊認知落差時，同時 BIMFeeD 有助於提高變更設計時的資訊透明度，亦可用於流程的確認和記錄，但其建議必須釐清使用者的工作權責和所扮演的角色，即協同作業時對於模型修改的權限管理，若能配合專案管理系統應會獲得較佳的管理效率，例如：資訊回饋端使用者或有主動提出模型修改之需求，而非僅由資訊請求端之模型管理者來觸發模型修改的流程。

5.5 應用案例之流程比較

在 4.6 節已提及 BIMFeeD 與專案管理系統整合之必要性，惟系統整合在實務上並非易事，且資訊回饋端使用者必須具備專案管理系統之權限以回覆模型修改建議，若專案管理系統無法跨平台操作則限制更多且更加不方便。

若為可延伸(extensible)之專案管理系統,則可利用其 API 搭配 4.3 節 BIMFeeD 之 Web Service，客製化開發與 BIMFeeD 整合之應用程式和作業流程；若為不可延伸(non-extensible)之專案管理系統，較可行的做法為以附加超連結之方式，將 BIMFeeD 網路應用程式之 URL 位址藉由專案管理系統傳送給專案參與者。

由於系統整合與客製化程式開發的成本極高，即便是具備 API 的商業套裝專案管理系統，在實務應用時仍以夾帶檔案或附加超連結作為權宜之計較常見。

為了比較傳統 BIM 模型修改和採用 BIMFeeD 之作業流程，茲以一實際案例作說明。本案例為一集合住宅大樓於施工階段始導入 BIM 之實例，主要目的為藉由 BIM 模型來進行施工檢討，即營造廠必須根據已完成細部設計之 CAD 圖面作為參考底圖，並以 CAD 圖面之座標和標註尺寸來建置 BIM 模型，因此當 CAD 圖面標示不清或圖面之間相互衝突的情況產生時，建模單位必須藉由 RFI 來聯繫設計單位以獲取正確的圖面資訊，再進一步修改 BIM 模型。

由於BIM模型之傳遞和瀏覽皆相當耗費時間,故以附帶截圖之方式來傳送RFI較為可行,若有必要始檢視 BIM 模型。本案例分別採用 Autodesk Revit 和 Autodesk Buzzsaw 作為建模軟體和專案管理系統,故建模單位需先將截圖檔案上傳至專案管理系統，並在建模軟體中建立明細表以作為清單，同時在明細表上附帶截圖檔案之超連結(即專案管理系統所產生之特定 URL 位址，如圖 38 所示)，在設計單位先不必檢視 BIM 模型的前提下，建模單位必須將檢查明細表匯出為 Microsoft Excel 格式,再交付設計單位藉由 Excel 檔案中之超連結進入專案管理系統檢視截圖與回覆問題(如圖 39所示),爾後再由建模單位根據設計單位之回覆手動修改BIM模型,整體流程非常繁複及冗長，如圖 40 之 BPMN 流程圖所示。

標檢查表

Family	Type	參考樓層	備註	需檢查	已檢查	已確認	流水號	時間	問題URL
M_混凝	1500x300	B3F		☑	☑	☑			
M_混凝	400x700	B1F	參考哪張圖?	☑	☑	☑			
M_混凝	300x600	B1F	尺寸與圖面不符	☑	☑	☑			
M_混凝	400 x 400	1FL	尺寸與圖面不符	☑	☑	☑			
M_混凝	400 x 400	1FL	尺寸與圖面不符	☑	☑	☑			
M_混凝	300x600	1FL	尺寸與結構景觀圖不	☑	☐	☑			
M_混凝	250x600	1FL	尺寸與結構景觀圖不	☑	☐	☑			
M_混凝	750x750	1FL	卡進牆壁	☑	☑	☑			
漸變標族	500x1000	1FL	無折梁對照圖待修正	☑	☑	☑			
M_混凝	700x1300	1FL	另做特殊尺寸漸變梁	☑	☑	☑	2013032500	20130425	https://projectpoint.buzzsaw.com/client/NTU_BIM_Center/H%26H/RFI/URL%20RFI/201303
M_混凝	500x1000	1FL	無折梁對照圖待修正	☑	☑	☑			
M_混凝	800x2000	1FL	沒有與樓板貼齊	☑	☑	☑			
M_混凝	700x1300	1FL	另做特殊尺寸漸變梁	☑	☑	☑			https://projectpoint.buzzsaw.com/client/NTU_BIM_Center/H%26H/RFI/URL%20RFI/201303
M_混凝	700x1300	1FL	另做特殊尺寸漸變梁	☑	☑	☑			https://projectpoint.buzzsaw.com/client/NTU_BIM_Center/H%26H/RFI/URL%20RFI/201303
M_混凝	700x1000	1FL	另做特殊尺寸漸變梁	☑	☑	☑			https://projectpoint.buzzsaw.com/client/NTU_BIM_Center/H%26H/RFI/URL%20RFI/201303

圖 38：附帶截圖檔案超連結之檢查明細表

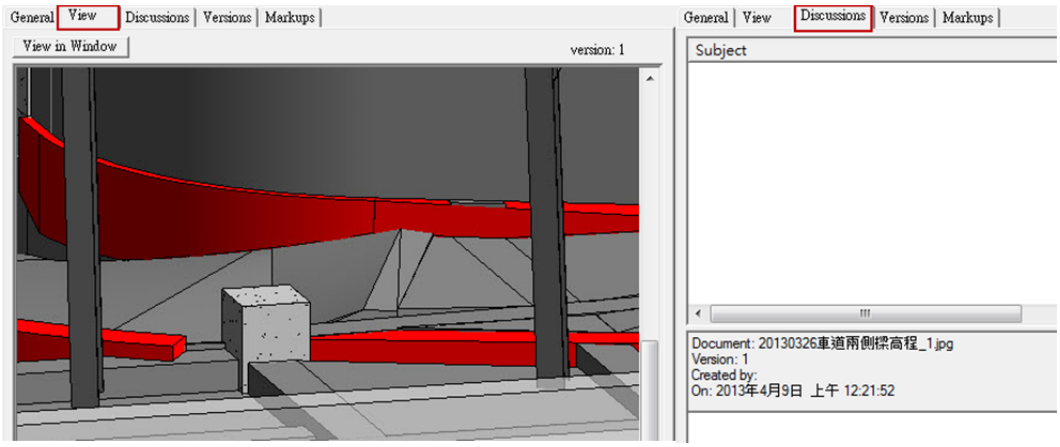


圖 39：在專案管理系統中檢視截圖與回覆問題

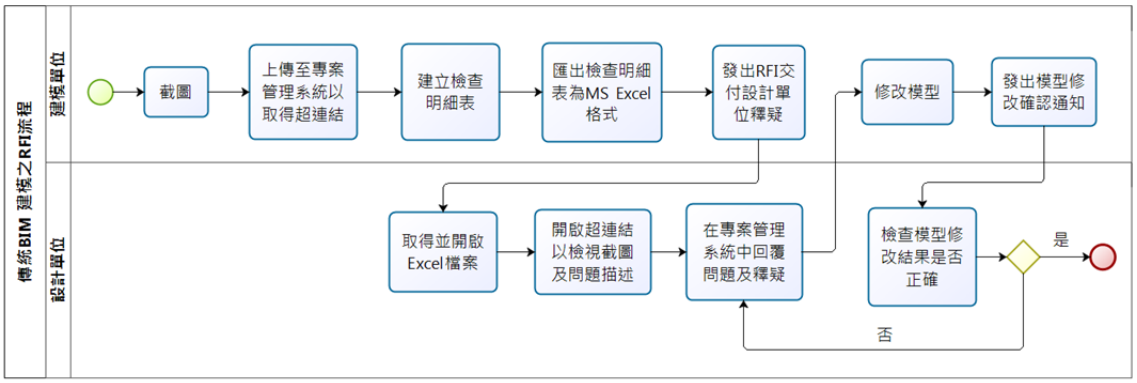


圖 40：傳統 BIM 建模之 RFI 流程

若採用 BIMFeeD 作為 BIM 模型修改之 RFI 流程，則整體流程簡化許多，因 BIMFeeD 從截圖、產生超連結、發出通知，乃至修改 BIM 模型皆是自動化的流程，如圖 41 所示。

以本案例之其中一個 RFI 為例，由於有一標號為 b4 之梁斷面尺寸與 CAD 結構圖面所標示的尺寸並不完全相同，如圖 42 被選取之元件所示，故建模單位可透

過 BIMFeeD 建立 Element Type 任務(因梁斷面尺寸係由元件類型所定義)，如圖 43 所示，在設計單位收到通知後即可根據建模單位對於任務之描述，直接在 BIMFeeD 網路應用程式上選擇正確的梁斷面尺寸，如圖 44 所示，而建模單位僅需透過 BIMFeeD 外掛程式選擇「接受」設計單位之模型修改建議，該 b4 梁元件即自動完成修改，整體流程較採用單一專案管理系統的應用模式更加有效率。

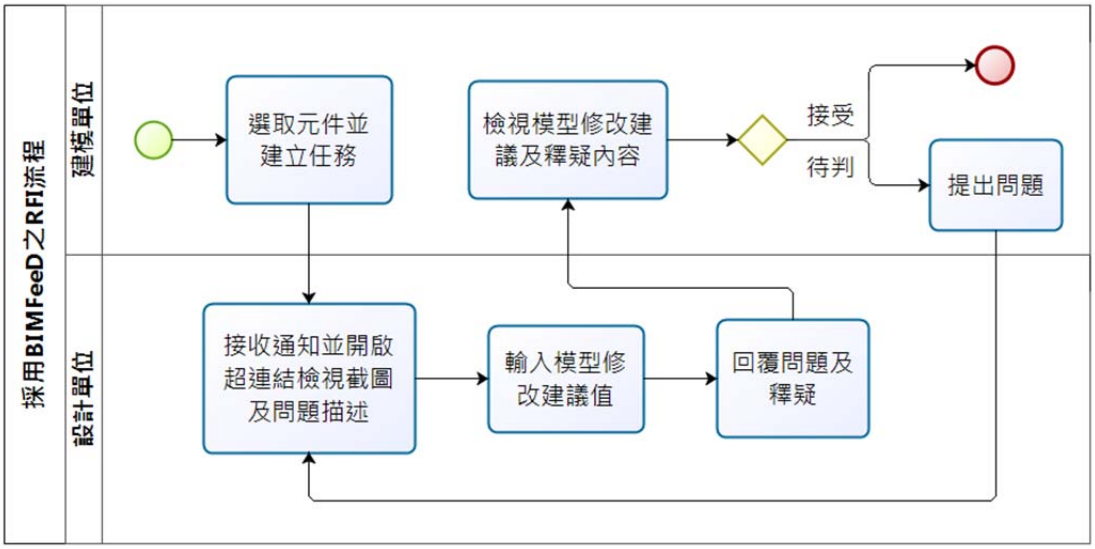


圖 41：採用 BIMFeeD 之 RFI 流程

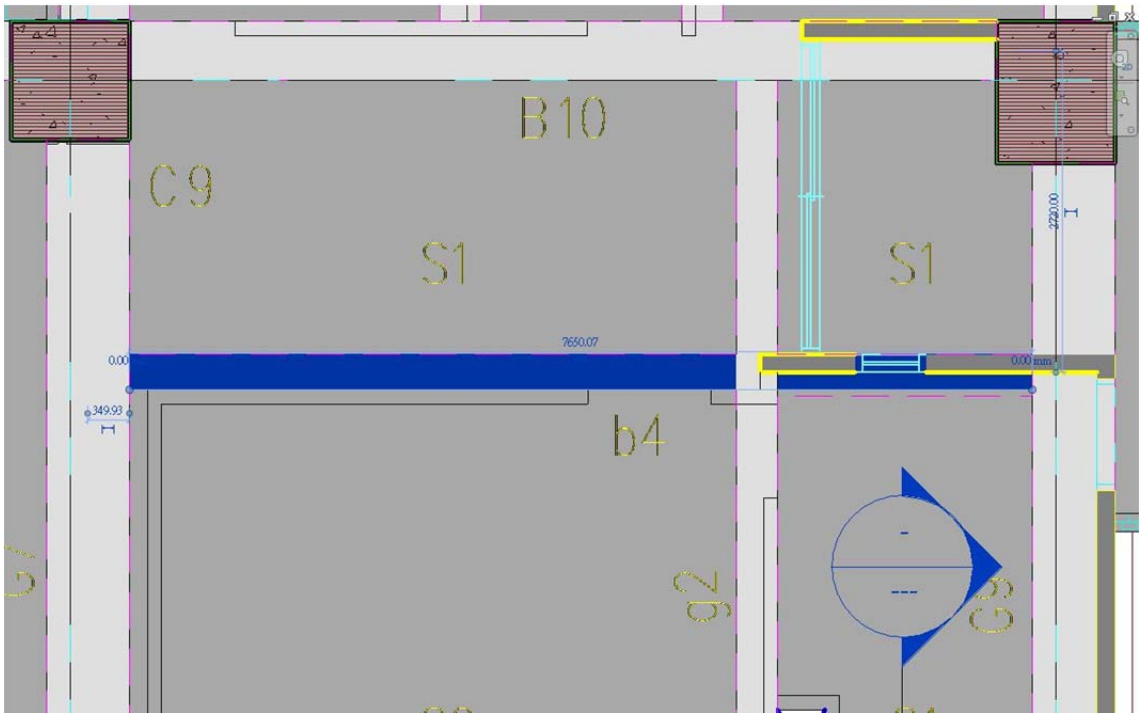


圖 42：標號為 b4 之梁尺寸與圖面不合

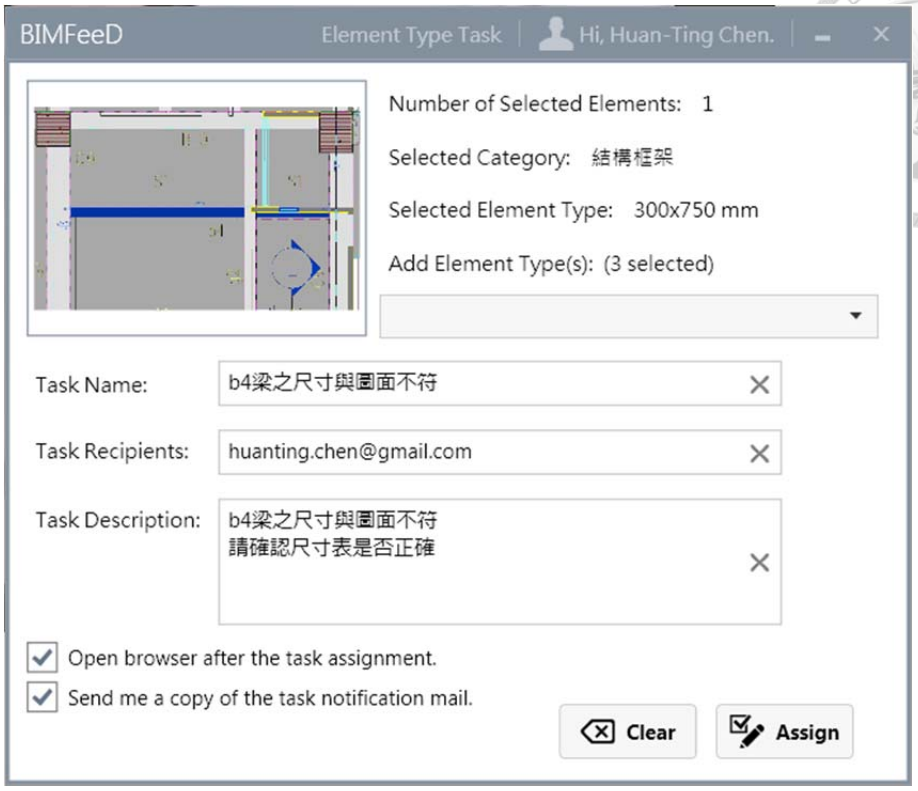


圖 43：建模單位建立 BIMFeeD 之 Element Type 任務

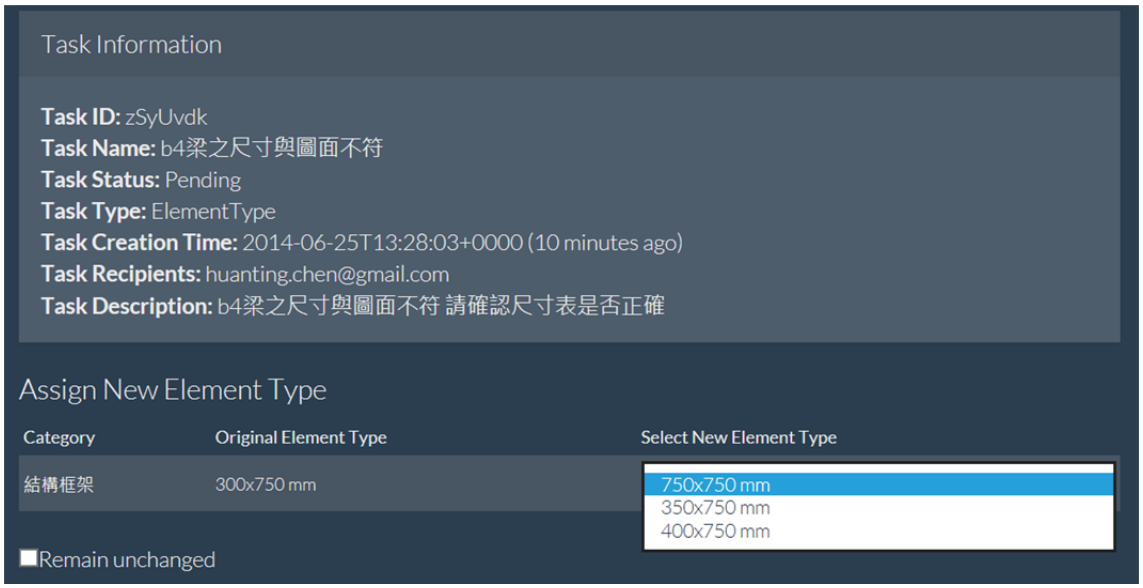


圖 44：設計單位可在 BIMFeeD 網路應用程式直接選擇正確的梁斷面尺寸

第六章　結論

6.1　整體貢獻

有鑑於 BIM 模型修改之異質協同設計需求，本研究著眼於資訊回饋至 BIM 模型的解決方案，所謂修改乃意謂 BIM 模型已具備一定完整度之前提，故模型元件和其參數化屬性的修改或更新即是一種資訊回饋的過程，本研究先於 1.2 節針對資訊交換的方法和相關研究進行回顧，並於第二章分析生命週期中各種 BIM 模型修改之應用情境，在徵詢專家意見和歸納現有相互操作性之解決方案後，本研究於 2.7 節提出「資訊管理之應用架構」，以資訊回饋資料庫管理系統來整合資訊回饋的需求。

第三章的系統分析將上述需求更加明確化，首先針對不同需求提出「應用情境導向之資訊管理框架」，使資訊回饋至 BIM 模型的可能性擴及到不同的途徑或方法，再將本系統的開發範圍收斂在合適的產品定位，同時分析其運作機制和操作邏輯，最後於 3.4 節從不同面向來考量系統開發的技術細節。

第四章的系統設計包含使用者的操作系統和系統內部的設計模式，前者包含資訊請求端之建模工具外掛程式和資訊回饋端之網路應用程式，後者包含免綱要式資料結構和資訊傳遞方法之設計。免綱要式資料結構乃配合 NoSQL 資料庫技術進行開發，在生命週期中可能遭遇不同專業人員、不同領域知識、不同應用軟體、不同操作環境、不同實作技術、不同發展程度和不同時空條件，免綱要式資料結構可隨時根據需求來調整資料庫之資料綱要，此即佛法中之「法無定法」。

第五章將系統設計具體實作出來，並於 5.1 節逐一說明所採用之工具與技術，再示範外掛程式和網路應用程式的系統操作，最後則進行使用者訪談與驗證。

從 2.5 節和 5.4 節的專家意見中可得知：BIM 模型修改的需求確實有許多不同的應用情境和工作流程，本研究開發之系統對於較一般化和單純化之問題有所助益，對於資訊回饋至 BIM 模型的協同作業效率亦能提升，而本研究之目的本來就

不在於追求一個完全的解決方案，若 BIMFeeD 在某些較單純的情況下得以輔助進行 BIM 模型修改，則本研究欲展現之資訊回饋機制和操作系統已有初步的貢獻。

本研究開發一資訊回饋至BIM模型之使用者操作系統,藉由此系統可跨平台、跨時空進行非同步的 BIM 模型資訊交換作業，資訊請求端可向資訊回饋端發出請求資訊之「任務」，而資訊回饋端可根據資訊請求端所提供之任務資訊，進而回報正確合宜的回饋資訊，即模型修改之「建議」。

本系統之任務乃基於 BIM 模型之物件層級進行資訊交換，實能反映資料存取之實際需求，而為了因應生命週期中各種資訊交換需求，所採用之 NoSQL 資料庫技術和所設計之免綱要式資料結構甚具彈性，以物件導向架構為基礎的 BIM 模型可充分利用於資料儲存、交換和整合。

總結而言，當修改 BIM 模型需要透過協同作業來取得外部修改建議時，本系統可協助模型管理者根據回饋資訊自動化修改模型，透過資料庫模式而非透過檔案傳輸來進行資訊交換，主要考量是為了提升跨平台作業和非同步作業之效率，前者因協同作業者所使用之設備、平台、環境可能不盡相同，又或協同作業者可能不具備操作BIM設計工具之能力;後者因協同作業者可能身處不同時間或空間，因而無法進行同步作業。

本研究屬於較一般化的整合型解決方案,相較於 1.2.1 節所述之四種 BIM 模型之資訊交換單一方法，具有操作容易、應用彈性高、相互操作性佳等優點，在自動化修改模型元件時能完全符合 BIM 設計工具之商業邏輯，同時結合檔案管理模式和資料庫管理模式之優勢，而能滿足 BIM 模型資訊交換之跨平台作業和非同步作業之「異質協同設計」需求，更甚者，每一筆任務描述、建議回覆、模型修改之記錄皆會被系統保存下來，若爾後發生爭議時可作為呈堂證供。

6.2 未來展望

4.6 節所敘述之專業版系統需求為本研究在未來可持續發展之方向，尤其專案管理系統之導入絕對有其必要，而本研究則主要呈現資訊回饋至 BIM 模型之資料交換機制。BIM 技術欲於全生命週期能充分應用，必須有許多不同工具和環境的配合，本研究利用雲端服務之概念來介接 BIM 模型檔案和 NoSQL 資料庫系統，創造了爾後更廣泛應用的可能性，除了使用者操作模式外，可基於「資訊管理之應用架構」和「應用情境導向之資訊管理框架」來發展不同應用需求的資訊交換系統，若資訊回饋資料庫系統能提供更多延伸性的解決方案(如：針對不同應用情境之客製化 web 服務 API)，則未來可發展為資訊交換的技術平台。

就專家意見對於 BIMFeeD 的整體改善建議而言，主要可分為三個層面：語意資訊的包裝、網路應用程式視覺化功能的加強、配合專案管理系統的權限管理。語意資訊包裝的前提是資訊請求端能了解資訊回饋端所期待的語意資訊，在系統設計層面可加入參數化屬性的自定義名稱或群組功能。網路應用程式視覺化功能的加強則仍有一些技術瓶頸,透過網路應用程式操作BIM模型的性能仍有待提升。配合專案管理系統的權限管理則屬於另一個範疇的管理模式，若專案管理系統的管控機制可和本研究之資訊回饋機制相互配合，則對於 BIM 模型修改和模型版次的管理皆能相得益彰。

另一方面，隨著硬體設備和網際網路的效能越來越高，巨量資料和雲端運算的發展趨勢有助於解決更複雜的資訊交換問題。以微觀角度而言，若資料模型和其傳輸方式能更具彈性，BIM 模型元件之間的互動行為和關係可由使用者根據需求來定義，則 BIM 的資訊整合可發展為法則式(rule-based)的檢核系統。以巨觀角度而言，若 BIM 模型隨生命週期發展的整合模式能被記錄下來，則其資料交換模式亦有機會形成一種知識管理的專家系統(expert system)。

參考文獻

ABANDA, F., ZHOU, W., TAH, J. & CHEUNG, F. 2013. Exploring the relationships between Linked Open Data and Building Information Modelling. Proceedings of the Sustainable Building Conference, Coventry University, UK, 3-5 July 2013.

AKCAMETE, A., AKINCI, B. & GARRETT, J. H. 2009. Motivation for computational support for updating building information models (BIMs). Proceedings of the 2009 ASCE International Workshop on Computing in Civil Engineering, 2009. 523-532.

AZHAR, S. 2011. Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadership and Management in Engineering*, 11 (3), 241-252.

BAKSHI, K. 2012. Considerations for big data: Architecture and approach. Aerospace Conference, 2012 IEEE, 3-10 March 2012. IEEE, 1-7.

BEETZ, J., VAN BERLO, L., DE LAAT, R. & VAN DEN HELM, P. 2010. BIMserver. org–An open source IFC model server. Proceedings of the CIP W78 conference, 2010.

BIMFORUM 2013. Level of Development Specification. AGC BIMForum.

CHEN, H.-M. & HOU, C.-C. 2014. Asynchronous online collaboration in BIM generation using hybrid client-server and P2P network. *Automation in Construction*, 45, 72-85.

CHEN, H. M., HOU, C. C. & LIN, T. H. 2013. A SYSTEM PLATFORM FOR BIM WEB SERVICES WITH INTEGRATION OF CLOUD COMPUTING AND WEB3D TECHNOLOGY. Proceedings of the 1st International Conference on Civil and Building Engineering Informatics (ICCBEI 2013), 7-8 November 2013, Tokyo, Japan. 333-338.

CHEN, Y. W. & HSIEH, S. H. 2013. A BIM ASSISTED RULE BASED APPROACH FOR CHECKING OF GREEN BUILDING DESIGN. Proceedings of the 13th International Conference on Construction Applications of Virtual Reality (CONVR 2013), 30-31 October 2013, London, UK.

CHENG, Y. M. & WU, I. C. 2013. PARAMETRIC BIM OBJECTS EXCHANGE AND SHARING BETWEEN HETEROGENEOUS BIM SYSTEMS. Proceedings of the 30th International Symposium on Automation and Robotics in Construction (ISARC 2013), 2013, Montréal, Canada. 329-336.

DAIGNEAU, R. 2012. *Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services*, Addison-Wesley.

EASTMAN, C., JEONG, Y.-S., SACKS, R. & KANER, I. 2009. Exchange model and exchange object concepts for implementation of national BIM standards. *Journal of Computing in Civil Engineering*, 24 (1), 25-34.

EASTMAN, C., TEICHOLZ, P., SACKS, R. & LISTON, K. 2011. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, Wiley.

FIELDING, R. T. 2000. *Architectural styles and the design of network-based software architectures.* University of California.

FLANAGAN, D. 2011. *JavaScript: The Definitive Guide*, O'Reilly Media, Incorporated.

HAMIL, S. 2012. *Building Information Modelling and interoperability* [Online]. National Building Specification, UK. Available: https://www.thenbs.com/topics/BIM/articles/bimAndInteroperability.asp [Accessed 15 April 2014].

HAN, J., HAIHONG, E., LE, G. & DU, J. 2011. Survey on NoSQL database. Proceedings of the 6th International Conference on Pervasive Computing and Applications (ICPCA 2011), 26-28 October 2011. IEEE, 363-366.

HIETANEN, J. & LEHTINEN, S. 2006. The useful minimum. *Tampere University of Technology, Tampere*.

HSIEH, S. H., KANG, S. C. & CHEN, H. T. 2013. TOWARD AN EXPERIENCE-ENHANCED BIM APPROACH FOR LIFECYCLE MANAGEMENT OF PUBLIC LIVING SPACE. Proceedings of the 1st International Conference on Civil and Building Engineering Informatics (ICCBEI 2013), 7-8 November 2013, Tokyo, Japan. 349-356.

HUANG, Z., HE, F., LI, X., CAI, X. & LIU, H. 2009. Exchange of Heterogeneous Feature Data in Concurrent Engineering and Collaborative Design Environments. *Global Perspective for Competitive Enterprise, Economy and Ecology*. Springer.

ISIKDAG, U. 2012. Design patterns for BIM-based service-oriented architectures. *Automation in Construction,* 25, 59-71.

JEONG, Y.-S., EASTMAN, C., SACKS, R. & KANER, I. 2009. Benchmark tests for BIM data exchanges of precast concrete. *Automation in construction,* 18 (4), 469-484.

KIVINIEMI, A. 2006. Ten years of IFC-development–Why are we not yet there. Keynote lecture at the 2006 Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montreal, Canada, 2006.

LI, M., GAO, S. & WANG, C. C. 2007. Real-time collaborative design with heterogeneous CAD systems based on neutral modeling commands. *Journal of Computing and Information Science in Engineering,* 7 (2), 113-125.

LI, Y. & MANOHARAN, S. 2013. A performance comparison of SQL and NoSQL databases. Proceedings of 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM 2013), 27-29 August 2013. IEEE, 15-19.

LIN, Y. H., LIU, Y. S., GAO, G., HAN, X. G., LAI, C. Y. & GU, M. 2013. The IFC-based path planning for 3D indoor spaces. *Advanced Engineering Informatics,* 27 (2), 189-205.

LIU, R. & ISSA, R. 2012. Automatically Updating Maintenance Information from A BIM Database. Proceedings of the 2012 ASCE International Conference on Computing in Civil Engineering, 17-20 June 2012. 373-380.

MA, Z. 2012. A BIM-based approach to reusing construction firm's management information. *Australasian Journal of Construction Economics and Building,* 12 (4), 29-38.

MAILE, T., FISCHER, M. & BAZJANAC, V. 2007. Building energy performance simulation tools-a life-cycle and interoperable perspective. *Center for Integrated Facility Engineering (CIFE) Working Paper*, 107, 1-49.

MISHRA, J. 2011. *Software Engineering*, Dorling Kindersley.

MOHAMED, S., TILLEY, P. & TUCKER, S. 1999. Quantifying the time and cost associated with the request for information (RFI) process in construction. *International Journal of Construction Information Technology*, 7 (1), 35-50.

MONTEIRO, A. & MARTINS, J. P. P. 2012. SIGABIM: a framework for BIM application. XXXVIII IAHS World Congress, 2012.

NOUR, M. 2009. Performance of different (BIM/IFC) exchange formats within private collaborative workspace for collaborative work. *Journal of Information Technology in Construction*, 14, 736-752.

NURSEITOV, N., PAULSON, M., REYNOLDS, R. & IZURIETA, C. 2009. Comparison of JSON and XML Data Interchange Formats: A Case Study.

PARK, Y. H., CHO, C. Y. & LEE, G. 2011. Identifying a subset of BPMN for IDM development. Proceedings of the CIB W78-W102 2011: International Conference, 2011, Sophia Antipolis, France. 26-28.

PAZLAR, T. & TURK, Z. 2008. Interoperability in practice: geometric data exchance using the IFC standard. *Journal of Information Technology in Construction*, 13, 362-380.

PORWAL, A. & HEWAGE, K. N. 2013. Building Information Modeling (BIM) partnering framework for public construction projects. *Automation in Construction*, 31, 204-214.

RASYS, E., HODDS, M., DAWOOD, N. & KASSEM, M. 2014. A Web3D Enabled Information Integration Framework for Facility Management. Australasian Journal of Construction Economics and Building-Conference Series, 2014. 1-12.

REENSKAUG, T. M. H. 1979. The original MVC reports.

SANFILIPPO, S. & NOORDHUIS, P. 2010. *Redis* [Online]. Available: http://redis.io/ [Accessed 27 Mar. 2014].

SASAKI, T. 2011. *NoSQL DATABASE FIRST GUIDE*, SHUWASYSTEM Co., Ltd.

SAWHNEY, A. & MAHESWARI, J. U. 2013. Design Coordination Using Cloud-based Smart Building Element Models. *International Journal of Computer Information Systems and Industrial Management Applications,* 5, 445-453.

SHAFIQ, M. T., MATTHEWS, J. & LOCKLEY, S. 2013. A study of BIM collaboration requirements and available features in existing model collaboration systems. *Journal of Information Technology in Construction (ITcon),* 18, 148-161.

SHEN, W., HAO, Q., MAK, H., NEELAMKAVIL, J., XIE, H., DICKINSON, J., THOMAS, R., PARDASANI, A. & XUE, H. 2010. Systems integration and collaboration in architecture, engineering, construction, and facilities management: a review. *Advanced Engineering Informatics,* 24 (2), 196-207.

SINGH, V., GU, N. & WANG, X. 2011. A theoretical framework of a BIM-based multi-disciplinary collaboration platform. *Automation in Construction,* 20 (2), 134-144.

SOLID IT. 2012. *DB-Engines* [Online]. Solid IT. [Accessed 29 Mar. 2014].

STEEL, J., DROGEMULLER, R. & TOTH, B. 2012. Model interoperability in building information modelling. *Software & Systems Modeling,* 11 (1), 99-109.

SUMARAY, A. & MAKKI, S. K. 2012. A comparison of data serialization formats for optimal efficiency on a mobile platform. Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, 2012. ACM, 48.

TANG, P., HUBER, D., AKINCI, B., LIPMAN, R. & LYTLE, A. 2010. Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in Construction,* 19 (7), 829-843.

TAUSCHER, E. & THEILER, M. 2013. *Java Toolbox IFC2x3/IFC4* [Online]. Available: http://www.ifctoolsproject.com [Accessed 28 Mar. 2014].

TIWARI, S. 2011. *Professional NoSQL*, John Wiley & Sons.

WEI, Z. & MA, Z. 2012. Utilizing freeware tools to develop a 3D graphics interactive module for non-authoring BIM-based applications. Proceedings of the 14th

International Conference on Computing in Civil and Building Engineering, 27-29, June. 2012, Moscow, Russia.

WHITE, S. A. 2008. *BPMN modeling and reference guide: understanding and using BPMN,* Future Strategies Inc.

YANG, Q. & ZHANG, Y. 2006. Semantic interoperability in building design: Methods and tools. *Computer-Aided Design*, 38 (10), 1099-1112.

YENERIM, D. & YAN, W. 2011. BIM-Based Parametric Modeling: A Case Study. Proceedings of MSV'11-The 2011 International Conference on Modeling, Simulation and Visualization Methods, 2011.

郭榮欽 2013. BIM 應用於建管行政之現況. *營建知訊*, 370, 58-67.

郭榮欽, 簡添福 & 謝尚賢. 2013.「BIM 工程實作雲端服務平台」之基礎建設與系統初步規劃探討. 2013 年電子計算機於土木水利工程應用研討會論文集, 2013.

蔡孟君. 2013. *導入 BIM 於工程協同作業的衝擊與效益之個案研究*. 臺灣大學.

謝尚賢 2013. 建築資訊模型該如何準備與交付？. *營建知訊*, 365, 60-63.