以軟體工廠理論快速開發研討會管理系統
Using Software Factory for Rapid Development of Conference Management System

李俊瑩* 廖峻鋒† 張景堯† 李蔡彥†
Chun-Yin Li*, Chun-Feng Liao†, Jiing-Yao Chang†, and Tsai-Yen Li†
*國立政治大學資訊科學研究所,†國立政治大學電子計算機中心
*g9311@cs.nccu.edu.tw, †{try,jychang,li}@nccu.edu.tw

摘要

傳統的軟體開發通常由需求分析、系統分析、系統設計、系統實作到系統測試循序漸進的瀑布式軟體製程（Waterfall Process）[2]。為了突破單一反覆週期（Iteration）的軟體製程中，變動系統結構所需成本居高不下的困境，近年來，將整個開發流程切割成數個反覆週期的軟體製程（Iterative Process）已成主流，例如 UP(Unified Process)與 XP(eXtreme Programming)。雖然反覆式軟體製程可以降低系統的變動成本，但在設計同質性高的軟體時，這些反覆的過程會浪費大量的人力及成本。軟體工廠（Software Factory）理論[3][4]，是一種透過重用既有的軟體資產（Software Assets），快速生產出同質性高的軟體製程技術。在本篇論文中，我們將討論如何基於一個軟體工廠的理論，藉由可擴充的軟體工具與高度的軟體資產重用，快速建構一個具有完整功能的研討會管理系統，並以 2004 年的 TAAI 研討會管理系統為例，說明以軟體工廠建立出來的研討會管理系統的開發過程。

關鍵詞：研討會管理系統、軟體工廠、軟體產品線、軟體製程

Abstract

Traditional software development usually takes a Waterfall Process [2] approach, in which software is developed in the order of requirement analysis, system analysis, system design, system implementation, and system testing. In order to deal with the high cost associated with the changes of system requirements during the development, iterative processes that divide the development process into several iterative parts, such as UP (Unified Process) and XP (eXtreme Programming), have become widely accepted solutions. An iterative software process involves many repetitive steps such as requirement analysis, system analysis, system design, and system test at each round, and some of these steps may be redundant when we develop highly similar software. On the other hand, Software Factory is a software development concept that reuses software assets to produce similar software with a more efficient and cost-effective manner. In this paper, we will discuss how to build a complete system by reusing software assets with extendable tools based on Software Factory. We will use the conference management system for TAAI2004 conference as an example to illustrate the idea and feasibility of rapid software development and deployment process.

Keywords: Conference Management System, Software Factory, Software Product Line, Software Process

1. 前言

近年來軟體開發技術不斷進步，藉由自動化開發建構工具與程式碼產生技術（Code Generation）的使用，使軟體的開發成本得以降低。傳統的軟體製程（Software Process）[2]依序是需求分析、系統分析、系統設計、實作與測試等等循序漸進的瀑布式軟體製程（Waterfall Process）。這種製程的缺點在整個流程只進行一次，無法回頭進行修正，導致整合上面臨相當大的風險。相對而言，反覆週期的軟體製程（Iterative Process）將製程分割為數個週期，以改善傳統的瀑布式軟體製程（Waterfall Process）。但這些軟體製程中沒有明確定義如何有系統地重用故有的軟體資產（Software Assets），例如軟體元件，軟體樣式（Software Patterns）或應用程式框架（Application Framework）等。軟體工廠（Software Factory）[3][4]的概念，是透過重用既有的軟體資產（Software Assets），快速產生出同質性高的軟體製程改善技術。本文的目的在以客制化研討會管理系統為例，討論如何基於軟體工廠理論的概念，定義軟體工廠綱要（Software Factory Schema），以一些可延伸的建構工具整合軟體工廠樣板（Software Factory Template）與軟體工廠綱要，來快速開發研討會管理系統這類同質性高的軟體。

學術研討會往往耗費大量人力在研討會論文投稿、論文審查、參與者註冊等各項業務。近年來大部份的研討會都採用 Web 的方式與論文投稿者及論文審查者互動。建置研討會管理系統的目的是希望能透過 Web 的機制，方便研討會的論文投稿者、參與者與論文審查者於此網站中進行各種工作，以提升系統的可及性(Accessibility)，並減輕管理人力的消耗。網路上雖然已經有越來越多此類軟體的開發[8]，但不是所費不貲就是能滿足研討會整

體需求（包含投稿、審稿、註冊及管理等）並能考慮不同研討會之需求者並不多。為達到使用方便且節省人力的目的，研討會管理系統的功能常較為繁雜。一個功能完備的研討會管理系統的至少包含：

- 公告會議的最新資訊：包括舉行的地點與時間、飯店的安排、議程的排定、徵求論文的期限、論文的錄取與否等事項。
- 一般與會者的線上報名：如此可知道與會的人數，及食宿的安排。
- 論文投稿者的管理：包括其個人的基本資料設定、上傳論文、論文資料的編修、查看論文是否被錄取、審查者對其論文的建議、議程設定的安排等，如此可方便論文投稿者對於此次研討會的參加。
- 論文審查及錄取的管理：包括對每一篇論文的審查者安排、線上審查計分、線上給予該篇論文建議、爭議論文等設定，目的在使審查的過程簡單化、公平化，冀望能將所有審查過程均透過網路完成。
- 系統本身的管理：此項是希望系統管理者有絕對的權力去修改、新增、刪除以上各個事項，並處理例外的情況。

雖然研討會的類型與領域眾多，但由於研討會的主要運作模式與功能需求大致相同，差異的部份較少，是屬於高同質性的軟體系統，相當適合套用軟體工廠的理論來達成快速開發的目的。本研究主要目的即在於應用軟體工廠的概念，基於軟體工廠綱要(Software Factory Schema)結合既有的軟體資產建構出軟體工廠樣板(Software Factory Template)，以快速開發出適用性高的研討會管理系統，達到節省人力成本的目的。

2. 研究背景與文獻探討

2.1 軟體產品線(Software Product Line)

軟體產品線（Software Product Line）[5]是針對特定的領域，以既有核心資產開發而來同質性高的軟體系統集合（Software Products Family）。開發分成三步驟，分別是軟體產品線的分析、設計與實作。軟體產品線分析產生產品線的規格表，描述哪些產品名列於軟體產品線可生產的行列；軟體產品線設計與實作則分別是定義產品的生產流程並實作這些流程。

產品線分析的目的在於決定產品線需開發何種產品。分析過程中需先定義問題領域，例如開發的產品中應有的角色或物件、產品特徵、產品功能等等，都必須在此階段定義清楚。產品線分析決定需求，產品線設計針對這些需求提出解決方案，其中包括系統架構、需求與功能對應、開發系統的流程等等。產品線實作根據產品線設計後的系統架構及開發流程實作軟體資產，產出物為可用的軟體資產。

2.2 軟體工廠(Software Factory)

傳統軟體製程至少要經過幾個步驟：需求分析、軟體設計、軟體實作及軟體測試。新的軟體就要從分析、設計及實作等步驟重新開始，從目前的情況看來，軟體開發的速度是緩慢、昂貴且錯誤率高的。對軟體開發人員，軟體工廠正是一種可通過重複的開發過程，生產出效率高、成本低且質量好的方案。

軟體工廠是為了讓某種類應用程式可以快速開發所配置的開發環境；它是一種使用軟體工廠樣板（software factory template）且建築在軟體工廠綱要（software factory schema）的軟體產品線，組織一些可延伸的工具（例如 ANT[1]）和程序，以及可重用現有的元件，快速開發同質性高的軟體。

軟體工廠中的兩個關鍵元素為軟體工廠綱要（software factory schema）與軟體工廠樣板（Software Factory Template）。其中軟體工廠樣板是根據軟體工廠綱要所設計，兩者有密切關係。有了軟體工廠綱要及軟體工廠樣板後，便可以參考軟體工廠綱要，再根據軟體工廠樣板裡有的資產，最後可利用可延伸工具自動快速生產符合我們需要的軟體。

在軟體工廠中，重要的元素之一為軟體工廠綱要。軟體工廠綱要描述軟體工廠中需要的原料（例如程式碼與 SQL 檔等）、定義這些原料間的關係、也說明這些原料如何組成新的同質性軟體。軟體工廠綱要透過軟體產品線分析及軟體產品線設計來建置；而軟體產品線實作，則是實作軟體工廠綱要中定義的物件及其關係。軟體工廠產生同質性的軟體時，也必須參考這份綱要。

軟體工廠綱要列出軟體工廠所需要的所有原料，而接下來則是實作這些原料，讓他們變成可用的資產。軟體工廠樣板是實作這些原料過程中樣式（patterns）、框架（framework）與 DSL（Domain Specific Language）等的集合。軟體工廠樣板包括了固定（fixed）與可變（variable）兩類資產，做這樣的分類可以讓軟體工廠生產出來的軟體更加彈性。

固定資產指的是軟體工廠產生出來的同質性軟體都不會改變的元素；例如在我們的研討會管理系統的投稿者註冊論文中，要求投稿者填寫論文資料這個功能被我們定義為固定資產。每個研討會管理系統軟體工廠產生出來的軟體，論文投稿者投稿時，都必須填寫這些資料。可變資產指的則是軟體工廠產生出來的同質性軟體不一定會相同的元素；例如資料庫連結，每個軟體連結的資料庫來源都不盡相同，可變資產的適用度我們可在部署軟體時做適合的調整。

2.3 ANT 建構工具

大小型專案開發過程中，通常會涉及許多檔案，檔案間又會以不同方式互相關聯；如果修改到其中一個檔案，便必須重新編譯，再重新連結。開發過程中會不斷地進行這些建構動作，而 ANT[1]便是可以化簡這類建構程序的一種工具。

ANT 是一種自動化的軟體建置工具，它能建置、測試和部署大大小小的專案，就像是類似 make（make-like）的工具。然而大部份類似 make 的工具，如 gnumake、nmake 等，都是在 shell 的環境使用的，使用這些工具其實也代表已經被限制在某些特定的作業系統，例如 UNIX，Linux 等。ANT 是一個基於 XML 的建構工具，所以有跨平台的優點，不限制於特定的作業系統。ANT 的配置文件基於 XML 文件，所以具有可攜帶、語法簡單及可讀性高等等優點，讓開發者很容易地製作動態的建置文件。本文的實際案例中，將以 ANT 來部署研討會管理系統。

3 軟體工廠之設計與建構

重新建構一個研討會管理系統通常耗時耗力且在短期內軟體品質不易穩定。但值得注意的是，一般研討會管理系統的功能與執行流程大多類似，不同的研討會之間可能只在如開會時程、收費方式與評分規則有所差異。因此，若能對一個既有的研討會管理系統做有系統的重用，可以達成快速開發高品質且功能完備的研討會管理系統的效果。本章節的主要目的即在於討論如何建立一個研討管理會系統的軟體工廠，達成快速設計與生產其他研討會管理系統的目標。

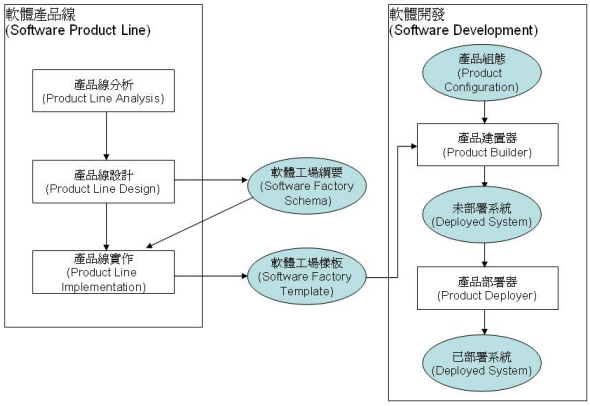

圖 1 軟體工廠建構產品過程

圖 1 係參考[3][4]所繪，顯示本文以軟體工廠理論建構一個研討會管理系統軟體工廠，並實際開發軟體的過程。矩形為開發步驟，橢圓形為產出物。首先，在軟體產品線的產品線分析及產品線設計中，建構出軟體工廠綱要；在產品線實作步驟中實作軟體工廠綱要中所列舉之原料；實作後所產生的軟體資產構成軟體工廠樣板，如此便完成軟體工廠的基本要素。

接下來我們將簡述如何以此軟體工廠開發軟體。產品建置器以軟體工廠樣板為原料，根據產品組態可組織出未經過部署的系統，在經過產品部署器將其部署完成後，即可產生出一個軟體工廠所開發的軟體。

3.1 建構軟體產品線

軟體產品線的目的是列舉軟體工廠所需的軟體資產及建構這些軟體資產提供軟體工廠所用。為此目的，需以產品線分析、產品線設計及產品線實作來達成。下面章節我們將說明研討會管理系統軟體工廠的軟體產品線。

3.1.1 產品線分析（Product Line Analysis）

經過產品線分析的步驟可充分了解需求，以開發出適合使用者的產品。經由初步的需求分析可發現，一般研討會的流程大致相同，依時間先後順序是接受論文投稿者投稿、論文審查者審查論文及大會參與者報名註冊。圖 2 為研討會管理系統經需求分析後所得之使用案例圖（Use Case Diagram），論文作者使用論文投稿系統進行投稿作業；議程委員以論文審查系統分派論文給匿名審查人，讓匿名審查人進行審查作業；會議參與者透過會議註冊系統完成會議註冊作業；而系統管理員可透過管理系統來檢查投稿、審查及註冊狀況。

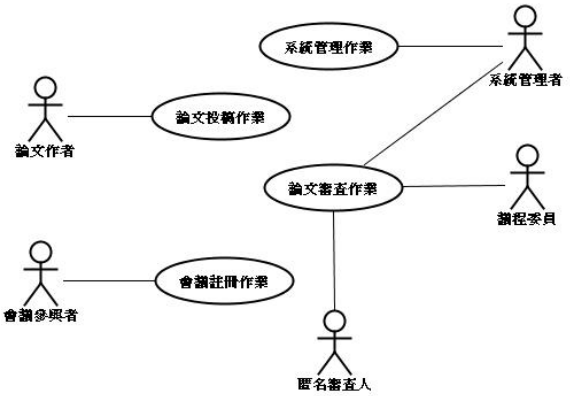

圖 2 研討會管理系統案例圖

3.1.2 產品線設計（Product Line Design）

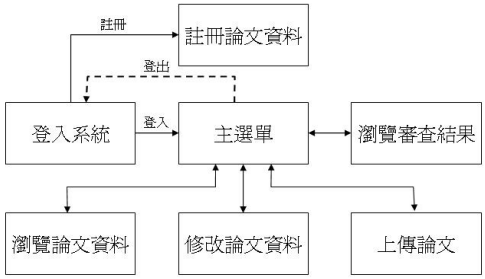

圖 3 論文投稿系統架構

經過產品線分析確認需求後，產品線設計根據分析後所得到的需求來設計系統架構。圖 3 為經過產品線分析後，針對論文作者投稿作業所設計的論文投稿系統功能架構。論文作者註冊投稿論文資料後，可以使用登入系統登入論文投稿系統，進行多項論文相關業務。在產品分析的步驟中，我們也會針對論文審查作業、會議註冊作業及系統管理作業設計系統架構。產品線設計最後將衍生出軟體工廠綱要，3.2 節將描述本研究所設計的軟體工廠綱要。

3.1.3 產品線實作（Product Line Implementation）

軟體工廠綱要列出所有必須的軟體資產，產品線實作的目的就是將這些軟體資產實作，成為軟體工廠樣板。我們在 3.3 節中將會進一步敘述研討會管理系統的軟體工廠樣板。

3.2 軟體工廠綱要

	業務面 (Business)	資訊面 (Information)	應用程式面 (Application)	實作技術面 (Technology)
Conceptual	•系統願景 •審稿評分政策 •收費政策 •論文分類 •重要時程與事件	•重要業務面術語定義 (Business Entities)	•子系統劃分 •會議業務流程 •定義共享服務	•QoS政策 •安全政策 •備份與還原政策
Logical	•工作流程 •系統角色定義 •使用案例 (Use case)	•領域模型 •資料模型 •物件-關連對應	•應用程式框架與服務	•系統保護機制設計 •系統負擔分散機制設計 •備份與還原機制設計 •佈署機制設計
Physical	•使用案例文件 •UML •使用案例圖與活動圖	•領域模型的UML類別圖 •資料庫綱要	•領域模型的類別程式碼 •Web應用程式 •應用程式框架程式碼 •應用程式設定檔	•應用程式伺服器設定 •備份腳本 •安全性過濾器 •建構與佈署腳本 •壓力測試工具

圖 4 研討會管理系統軟體工廠綱要

在產品線設計的步驟，針對產品分析所得的需求，設計出系統架構後產生研討會管理系統軟體工廠綱要。圖 4 為以 Zachman Framework[5]表示研討會管理系統軟體工廠的軟體工廠綱要。越上層的列代表越抽象的等級，由上往下越底層代表越接近實作，每一欄則代表不同概念的內容。概念層（Conceptual）代表使用者提出實務上的需求或定義；例如在業務面，研討會使用者會有系統願景，包括系統提供論文投稿功能及論文審查功能等等。在資訊面需求完整定義業務面術語，諸如投稿者、審查者、論文及與會者等等。邏輯層（Logical）中描述系統模型設計；例如在概念層資訊面中，我們描述了重要業務面術語定義，在邏輯層中就以資料模型來表現它。圖四為研討會管理系統軟體工廠中，「論文」的資料庫模型。物理層（Physical）實作在邏輯層中的設計，例如邏輯層中描述了「論文」的資料庫模型，在物理層中進一步將其精煉為資料庫綱要（Database Schema）。

3.3 軟體工廠樣板

研討會管理系統的軟體工廠樣板，即實現軟體工廠綱要中的軟體資產。這個步驟耗費大量人力成本，在實現這些元素時，要注意它們在軟體工廠綱要中是否可以拼湊起來，而不會產生空隙。為了讓軟體工廠彈性較大，我們將實現後的資產中變動率較高的資產抽取出來，將它修改成可變資產。

我們在研討會的軟體工廠中將以下系統中的元素抽取出來，當做我們的可變資產：

- 資料庫連結
- 研討會程序的時程
- 論文分類方式
- 系統管理員 E-Mail、研討會名稱
- 系統管理員權限
- 與會者註冊資料
- 與會收費規則

3.4 產品開發

建構好軟體工廠綱要及軟體工廠樣板後，研討會管理系統軟體工廠便準備好了，接下來的步驟是開發我們需要的產品，其中包含以產品建置器（Product Builder）及產品部署器（Product Deployer）建置及部署軟體。為了讓產生出來的軟體更適合不同的研討會，研討會管理系統軟體工廠提供產品組態（Product Configuration）給使用者修改。產品建置器便根據產品組態修改程式碼，重新編譯系統。例如不同的研討會所使用的資料庫來源都不同，經過產品建置器，產生的研討會管理系統便可連結到正確的資料庫來源。

經過產品建置器所產生出來的系統是未經部署的，透過產品部署器可以將未部署的系統移動至正確的路徑。

本研究中，研討會管理系統軟體工廠建置器及部署器的主要功能包含：

- 建立系統所需的資料庫表格
- 設定網路服務環境、開啟網路服務應用程式及新增副檔名應對應的執行檔路徑
- 更改資料庫連結檔，使系統連結至正確的資料來源
- 更改使用者介面文字，使其適用於新的研討會
- 重新編譯所有的原始碼
- 部署系統至正確位置

4 應用案例研討:TAAI 研討會管理系統

在 TANET2003 的研討會[6]中，我們在.NET Framework 上開發了一個完整的研討會管理系統。在 2004 年的人工智慧與應用研討會[7]中，我們以

既有的軟體資產，將其中變化較大的部份取出修改成可變資產，再以我們所建構的研討會管理系統軟體工廠來產生適用於 2004 年的人工智慧與應用研討會的軟體。本章節以 2004 年的人工智慧與應用研討會的研討會管理系統為例，介紹如何建構研討會管理系統。

圖 5 為本研究中以研討會管理系統軟體工廠，建置一個研討會管理系統的流程。首先我們參考 SQL 指令文件及組態文件，建置研討會管理系統的資料庫綱要（Database Schema）。SQL 指令文件以 SQL 語法描述需建立之資料庫表格；組態文件則描述連結資料庫來源等。接著 Web 伺服器設定模組根據組態文件設定 Web 伺服需開啟的應用程式服務及副檔名所對應的執行檔。最後研討會管理系統建置模組則依據組態文件建置研討會管理系統。建置過程中，我們亦依據組態檔的設定修改可變資產。

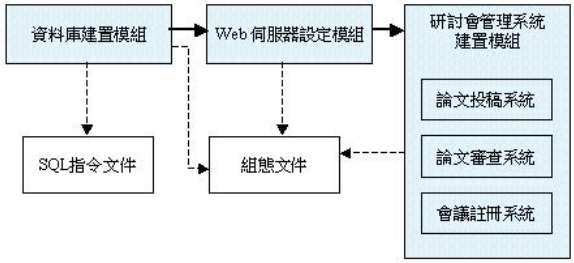

圖 5 研討會管理系統建製過程

4.1 可變資產（variable assets）

為了讓研討會管理系統軟體工廠更加彈性化，我們萃取出在各研討會間變動較高的軟體資產，將它們修改成可變資產。在 3.3 節中我們列出了本研究中的可變資產。以研討會程序的時程為例，不同的研討會在程序上的時程都不一樣。圖 6 為研討會管理系統設定各程序的時程，研討會中論文註冊、取消論文、查詢審查結果、會議註冊及 early bird 註冊等等時程都可部署後使用此系統更改設定。

圖 6 研討會時程表

4.2 維護組態文件

可變資產讓研討會管理系統軟體工廠更加彈性化。為了能在建置及部署系統之前修改可變資產，我們以一份組態檔設定此變化參數；在建置過程中參考此組態檔產生適用性高的程式碼，再加以編譯。圖 7 為一段組態檔，conference.paper_reg.start 為論文註冊開始時間，conference.paper_reg.end 為論文註冊結束時間。建置後系統將參考這些時間參數設定來決定啟用功能時程。

```
homedir=C:\\conference_builder
webdir=C:\\taai2004
DBadministrator=sa
DBADMPWD=a14d932004
DBname=test
DBsource=ge.cs.nccu.edu.tw
website=ge.cs.nccu.edu.tw
ConName=TAAI2004
EMailAdd=s8842@cs.nccu.edu.tw
conference.paper_reg.start=2004/6/15
conference.paper_reg.end=2004/7/15
conference.paper_can.start=2004/6/15
conference.paper_can.end=2004/6/15
conference.commnet_rev.start=2004/6/15
conference.commnet_rev.end=2004/6/15
conference.cam_rdy.start=2004/6/15
conference.cam_rdy.end=2004/6/15
```

圖 7 組態文件片段

4.3 系統建構

```
<sql
  driver="${DBdriver}"
  url="${urlheader}://${DBsource}:${websiteport};
  DatabaseName=${DBname}"
  userid="${DBadministrator}"
  password="${DBADMPWD}" >
  INSERT INTO c_due
  (action, start_date, end_date, description)
  VALUES
  ('paper_reg', '${conference.paper_reg.start}',
   '${conference.paper_reg.end}',
   'paper registration interval')
</sql>
```

圖 8 建構研討會日程腳本片段

```
<replaceregexp
  file="${webdir}/paper/SqlConnectionFactory.cs "
  match='static string DATASOURCE = "(.*)"'
  replace='static string DATASOURCE = "${DBsource}"'
  byline="true" />
<replaceregexp
  file="${webdir}/paper/SqlConnectionFactory.cs "
  match='static string DBNAME = "(.*)";'
  replace='static string DBNAME = "${DBname}";'
  byline="true" />
```

圖 9 建購資料庫連結檔腳本片段

組態文件寫作完成後，便可開始建置系統。其目的在以既有的軟體資產，根據組態檔作修改，產生符合研討會使用者所期望的原始碼，再加以編譯成為適用性高的軟體系統。圖 8 為系統建置片段，sql 為 ANT 的任務（task）節點，它是一組連結資料庫下指令的命令結合。sql 包含多項屬性，如描述使用何種資料庫驅動程式的 driver 屬性、資料庫來源位置的 url 屬性等等。圖 8 中的腳本片段即是描

寫如何將組態文件中定義的論文註冊開始及結束時間，新增至決定論文註冊日程的資料庫中。圖 9 為建構資料庫連結檔腳本片段，replaceregexp 為 ANT 定義的任務節點，它主要的任務在於以字串替換文件中以常規表示式（regular expression）表示的文字。圖 9 中腳本描寫將資料庫連結檔中的資料庫來源及資料庫名稱，修改為組態文件中所指定的資料庫來源及資料庫名稱。

4.4 規劃部署

圖 10 為研討會管理系統部署圖（Deployment Diagram），研討會管理系統分割成論文投稿系統、論文審查系統及會議註冊系統。透過 ADO.NET，存取系統資料庫，備份主機中則分別備份研討會管理系統資料庫及研討會管理系統，以備不時之需。

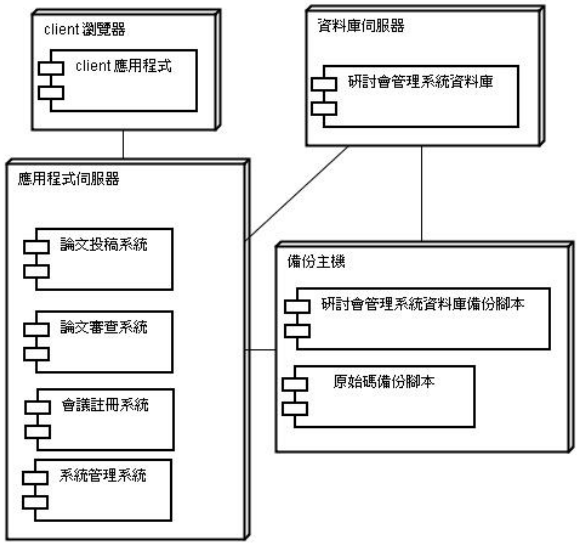

圖 10 研討會管理系統部署圖

4.5 部署系統

研討會管理系統軟體工廠透過簡單的步驟，便可建置出新的研討會軟體。首先我們需安裝開發工具，如 JVM、.NET framework 及 ANT，建立開發環境；接著我們需維護組態文件，因這份設定文件將決定產生出來之軟體的適用性；最後經過簡單的指令或直接點選執行檔就可建置研討會軟體並部署完成。

圖 11 為以 ANT 建製部署系統的流程，當使用者執行建置檔後，便開始建置資料庫綱要、設定 IIS 組態、建立網路目錄及建製研討會管理系統。圖 10 中實線代表參考文件，如建製資料庫綱要的 sql.xml 參考 init.sql，設定 IIS 的 iissetup.xml 參考了 build.properties，paper.xml 及 paper_adm.xml 分別建置部署論文投稿系統及論文審查系統，其他系統亦以相同的方式建置。

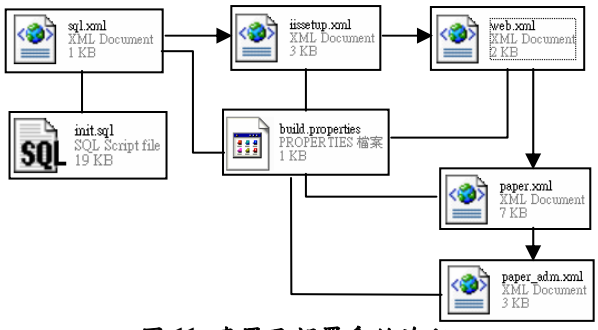

圖 11 建置及部署系統流程

5 結論與未來延展方向

本研究基於軟體工廠理論，建構軟體工廠綱要及軟體工廠樣板，藉由可擴充的工具開發研討會管理系統，並且將開發出來的研討會管理系統使用於 2004 年人工智慧與應用研討會，提供一個改善反覆式軟體製程的可行方法。本研究獲得結論為：對於同質性高的軟體，經由軟體工廠理論開發，確可減少浪費人力成本，快速建構高品質軟體。

軟體工廠中雖然採用的建置及部署工具為跨平台的 ANT，但主要的軟體資產採用.NET framework 開發，在不同的作業系統上無法啟用。若將既有的軟體資產以跨平台工具（如 Java）重構（Refactoring），未來研討會管理系統軟體工廠便可生產跨平台研討會管理系統。

目前研討會管理系統軟體工廠彈性尚有不足之處。以審查系統為例，各研討會審查規則皆有所不同，目前審查規則尚屬固定資產，使用者若需更改審查規則，則需重構論文審查系統，故擴充可變資產是未來需加強重點之一。

參考文獻

[1] Erik Hatcher, Steve Loughran, 邱忠文, “Ant 實作手冊”, 博碩, 2003.

[2] Ian Sommerville, ”Software Engineering,” 6th edition, MA: Addison-Wesley, 2001

[3] Jack Greenfield, “Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools,” available at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/softfact3.asp. .

[4] Jack Greenfield, Keith Short, “Software Factories,” Addison-Wesley, 2004.

[5] The Zechman Institute for framework advances, http://www.zifa.com/

[6] 台灣區學術網路研討會(TANET2003 conference), http://www.nccu.edu.tw/TANET2003.

[7] 台灣人工智慧與應用研討會(TAAI2004 conference), http://taai2004.nccu.edu.tw/

[8] START, http://www.softconf.com/START