摘 要

近年來智慧型手機的應用，不管是應用系統的開發或者是網頁的應用，都發展出各式各樣的系統軟體市集服務(App Store)。然而面對不同設備，擁有不同的作業系統與開發環境，開發人員所面臨的不單只是設備資源(CPU 執行效率、記憶體有限等)的不足，在開發上的問題更是另一項挑戰。如：一個以 iPhone 開發的系統不能執行在 Android 或 J2ME 系統上。原因在於 iPhone 和 Android、J2ME 的作業系統、開發語法都不相同，而導致必須各自開發系統，以符合在 iPhone 和 Android、J2ME 上使用。本研究結合 OMG 的 MDA(Model-Driven Architecture)的開發方法應用在 SOA(Service-Oriented Architecture)上，並採用 SoaML Profile 來進行 SOA 的分析、設計。另外我們也提出應用在 SOA 架構上的 MDA 開發流程步驟及方法應用。以及我們也設計了 MDA 的開發模式，並且與傳統的 RUP 開發模型相做比較。最後，我們也假想一個手持式設備的專案，採用我們所提出的開發流程方法，並且搭配使用 UML、SoaML Profile 及 ESB 工具來分析、繪製、部署此專案。

關鍵詞：SoaML、SOA、MDA、軟體工程

Abstract

In recent years the application of SmartPhones, regardless of application system development or web applications, have developed a variety of system software marketplace services. However, faced with different devices, with different operating system and development environment, developers are facing not only equipment resources (CPU execution efficiency, limited memory, etc.) deficiency. The problem in the development is another challenge. Such as: One can't be running in Android or J2ME system with system that iPhone develops. The reason is that iPhone, Android and J2ME operating systems, development of language are not the same, which led to their development of systems to meet the iPhone, Android and J2ME use. In this research, we combined with OMG's Model-Driven Architecture, MDA development method used in Service-Oriented Architecture, SOA and using SoaML Profile to carry out the analysis and design. Also, we proposed the SOA architecture of the MDA development process steps and methodology. And we have designed the MDA's development model and the RUP with the traditional development model to compare with. Finally, we also assumed a study

case, using the proposed methodology of development process, and with use of UML, ESB middleware and SoaML Profile to analyze, model and deploy for this study case.

Keywords：SoaML、SOA、MDA、Software Engineering

目 錄

圖目錄

表目錄

第1章 緒論

1.1動機與背景

國際市調機構 iSuppli 在預測指出，在樂觀的情況下，智慧型手機(SmartPhone)的出貨量在 2009 年會達到 1 億 9230 萬支，相較於 2008 年的 1 億 7360 萬支手持設備，年成長率達 11.1%，佔全部手機的出貨量達 17.4%，而最悲觀的情形下也有 6%的長成率，佔全部手機的出貨量達 16.6% [2]。另一份資策會 FIND 調查，3G 在 2009 年第二季，已佔台灣行動通信總門號數的 52.0%，達 1,361 萬戶[1]，而 4G 的 WiMax 也隨之而來。再則，各家手持式設備也紛紛的建立起自己的軟體市集服務(App Store)，除原本 iPhone、Microsoft、Android、BlackBerry、Symbian 外，今年韓國大廠 Samsung 也建立自己的軟體市集[3]。面對著手持式行動通訊的普及，和傳輸速度的提昇，以及應用軟體的配合，手持式行動通訊的軟體開發與架構部署，相對的顯為重要，不僅要考慮系統相容性，執行效率也必須滿足使用者，此外在開發上語言的一致性也相對來的重要。

有鑑於手持式設備的普及率逐年提昇，以及傳輸速度的提昇，從早期的 2G GSM 到 3G 和 WiFi，演變至最新的 4G WiMax。加上 2007 年 6 月 iPhone 在美國熱賣，同年 Google 在 11 月和其他 47 家成員號

召成立了「開放手機聯盟」(Open Handset Alliance)[5]，隔年 7 月和 10 月 iPhone Apple Store、Android Market[4]，分別上線讓使用者可以透過 App Store，下載軟體以延伸手機的服務與功能。今年 Microsoft、Samsung 也紛紛的加入 App Store 的市場，來爭取手持設備的商機。另外 Web Services 使用 HTTP、XML、AJAX 技術的成熟，使的 web widget 技術連結了 Twitter、Facebook、Flickr 等社交應用，即時的將訊息傳送到世界各地，以更增加了手持式設備的可用性。

然而手持式設備除了本身資源使用上的限制外，在開發手持式設備時在各家的平台上其實也有相關的門檻，且在傳統的 Web Services 的技術也有相對的問題，進而影響了開發人員的困擾。例如，設備平台所支援的瀏覽器不一樣，導致在 web 上開發所使用的 JavaScript、HTML 和協定會有所不同[17][18]，也因此開發人員必須先建置不同的平台來進行系統測試。加上，如果所開發的應用程式需支援各種手持設備。例如，企業開發應用系統，員工所持有的設備不相同，如：iPhone、Android、Nokia。這又使的開發人員必須建立起不同的開發平台進行系統的修改、測試，這不僅增加了開發成本與時程，也增加了測試的困難度。

近年來 SOA 架構已被許多企業所採用，SOA 支援了分散式的開發，強調了 Component-based 的運用，以及鬆散的界面(Loosely coupled)

運用，做為 SOA 核心的 ESB 更扮演了重要的角色。ESB 做為服務的提供者(Service Provider)與服務的消費者(Service Consumer)的 Middleware，提供了兩者的訊息傳遞，也可以轉換不同的協定，達到相互溝通的目的，更增加了 SOA 的完整性。此外 OMG 的 MDA (Model-Driven Architecture)開發架構，讓開發人員可將系統分成 CIM(Computation Independent Model)、PIM (Platform Independent Model)、PSM(Platform Specific Model)三個觀點來進行分析、開發、轉換、部署。在不同的觀點，開發者只要著重於該觀點的分析與進行，如此也簡化了開發的複雜性。另外使用 UML 來描述系統並配合使用 XMI、OCL，更能讓繪製的模型具有轉換、限制的能力，也讓模型具有重覆使用的能力。

因此本研究將結合 OMG 的 Model-Driven Architecture, MDA 的開發方法，應用在 Service-Oriented Architecture, SOA 的架構上，利用 SoaML Profile 進行 SOA 服務的分析和設計，並提出應用在 SOA 架構上的 MDA 開發流程步驟，運用在我們假想的開發系統上。最後我們透過假想的開發系統，提出了 MDA 的開發模型。

1.2研究目的

上述的觀念及問題，引發本研究的興趣。不同的手持設備有不同的開發界面；不同的手持設備的瀏覽器對 Web 的支援也有不同。也

因此我們的研究目的是希望能夠發展出一個針對 SOA 架構中採用 MDA、SoaML Profile 的開發的流程。使用 MDA 的三個觀點來進行系統分析及功能設計、平台轉換、及部署。其中的分析我們採用 SoaML Profile，來表示整個系統的架構，以及 SOA 中各個服務之間的關連。再將所繪製的 UML 模型來轉換到對映的平台，以達到 SOA 中各個服務的重覆使用性。另外我們將整理目前手持式設備相關平台的技術，以及目前在 Web Services 上開發所面臨的挑戰。

本研究論文分為兩大部份，第一部份主要探討手持式開發人員所面臨的遭遇。我們將再分成兩個子部份來探討。

- ✓ 應用程式開發環境部份：早期手持式設備的開發可分為 Windows Mobile 和 J2ME 的兩大技術。然而現今 iPhone、Android、BlackBerry 的出現，使得開發環境開始有了轉變，不僅各自開發的軟體不相容外，各自也有各自的開發平台。
- ✓ 網頁開發技術部份：使用網頁式環境，大致上都採用 Web Service 的方式連結，原因在於企業擁有異質性的平台環境，以及網頁語言是一個標準語言。在多項系統中，網頁式的系統是目前多數採用的環境，在技術上會使用到 HTML、SOAP、XML、JavaScript、AJAX 等技術。不同設備有不同的瀏覽器，如：Opera、Safari、FireFox、IE 等。但設備上的瀏覽器所支援的網頁技術卻不一致。

在第二章我們將更進一步探討。

第二部份是將手持式設備的開發方式，導入 SOA 的架構並將開發的環境分為四個子部份來探討。

- ✓ Client 端部份：服務消費者(Service Consumer)、手持硬體設備部份。擁有不同的作業系統(Operation system)、CPU 的執行速度、記憶體、瀏覽器。
- ✓ Server 端部份：服務提供者(Service Consumer)，提供各種不同服務讓服務消費者使用。服務也可呼叫其他服務而成新的服務。
- ✓ Middleware 中介部份：Enterprise Service Bus, ESB 與 web service。提供 Client 的呼叫並透過 ESB 的查詢、轉換將服務的描述回傳給 Client 端，再連結到 Server 以取得所需的服務。
- ✓ 通訊協定部份：如同第一部份的開發技術，在 SOA 中通訊也是重要的一環，Web Service 的技術帶動了異質性平台的溝通。XML、AJAX、HTML 等標準與非標準的技術也帶動了溝通的技術。同樣的我們也將在第二章再進一步的探討。

1.3論文架構

本論文各章節架構說明如下：

第 2 章：文獻探討

整理目前手持式設備上所使用的開發界面、語言選擇等，以及在

Web Services 的技術問題。並簡介 IBM 的 SOA、OMG 的 MDA、SoaML Profile 及目前仍為草案的 W3C HTML5。

第 3 章：開發理論與方法

我們提出“SOA 應用 MDA 的開發流程”方法以及架構，將說明各步驟。

第 4 章：SOA 應用 MDA 的開發流程

我們假想一個專案，並且運用我們所提出的“SOA 應用 MDA 的開發流程”，來做為手持式設備的開發流程。

第 5 章：MDA 開發流程比較

我們將利用 MDA 的三個主要觀點，建構出 MDA 的開發模型，並且與 RUP 的四個階段、六個核心工作流來做比較。

第 6 章：研究結論

將對本篇論文做出結論並期許未來工作。

第 7 章：參考文獻

第2章 文獻探討

手持式設備的開發方式，大制上分為應用程式開發(Application development)和網頁的應用開發(Web development)。應用程式開發的軟體，通常必須安裝在設備上才能使用。網頁的開發可分為兩種，一種是將軟體 plug-in 在網頁上使用，並透過設備的瀏覽器來操作使用。例如，Flash plug-in。另一種是直接使用瀏覽器操作一般標準語言的網頁。開發人員面對這兩種的開發模式，都具有相當的困難及困擾，我們將從手持式設備談起到開發環境，以及目前在開發所採用的 Web Services 的通訊協定和近年來企業所推崇的 SOA 架構。底下我們將分成六個小節來討論相關的研究。2.2 節我們研究了手持式設備的硬體、作業系統和瀏覽器部份；2.3 節針對了手持式設備的開發環境來做說明；2.4 節研究了網頁上的技術及 HTML5 所支援的新功能；2.5 節討論了 IBM 的 SOA 的架構和 Web Service、ESB 的關係，以及 OMG 提出的 SoaML Profile 與 SOA 的關係；2.6 節討論 MDA 架構和 MDA 與開發人員的關係；2.7 節為相關研究的總結。

2.1手持設備環境

以目前設備來說，硬體上，如：CPU、RAM、儲存設備等技術，每年都有新的技術提出和儲存容量的提昇，但仍比不上桌上型電腦擴

充來的便利。在作業系統方面，目前較為主流設備的作業系統，如：symbain、iPhone OS X(09 年為 iPhone OS 3.0)、Google 的 Android、Microsoft 的 Windows Mobile、BlackBerry OS、Brew、Plam One 等，各個作業系統都佔有一席之地[9][20]。且近年來 iPhone、Android 的加入，也漸漸的打亂原有市場的作業系統，使原本 Symbin 的作業系統佔有率有下滑趨勢。另外 web2.0 的盛行，促使設備上瀏覽器技術大大的提昇，但設備所採用的瀏覽器也都不相同[21][22]，如：以 webkit 為核心引擎的瀏覽器有 Safari、Android 和 Nokia S60；以 Presto 為核心引擎的瀏覽器有 Opera Mobile、Opera Mini；以 Gecko 為核心引擎的瀏覽器有 Mozilla；以 Trident 為核心引擎的瀏覽器有 Internet Explorer[23]。也因為各種瀏覽器所使用的核心引擎不一樣，造成了對網頁的解析、排版會有不同的結果。另外各瀏覽器對 Flash 的支援程度也不大一樣，如：以 plug-in 為主的 Adobe Flash 技術，目前就無法在 Safari 上執行。各自的設備既然無法使用同一種的瀏覽器，所幸 WHATWG 組織催動了 HTML5 的標準讓各種瀏覽器，可以使用同一種標準語言，讓設備不會因為支援不同瀏覽器而產生不同的網頁結果。2.4 小節，我們將更進一步討論 Web Service 的技術來解決瀏覽器的問題。

2.2 應用程式的開發環境 (Integrated Development

Environment)

開發環境影響了開發人員所使用的開發工具，如果要針對每一個設備建立不同開發環境，那將是相當的麻煩與費時。再則，軟體開發完成之後的測試環境，也都必須建立測試環境，才能逐一做測試，這都會造成開發人員的困擾。

開發語言與環境上，Sun 提供 J2ME 的 Wireless Toolkit 方便開發者可以使用 Eclipse IDE 來進行手持式設備的開發。採用 Java 語言並使用 Eclipse IDE，除了免去因開發不同設備可能架設不同的開發環境，以及造成另一個 IDE 的學習外，另一項原因是 Java 本身的好處是“write once run away”，同一個 Java 所寫的應用程式只要在不同設備上有相同 Virtual Machine 的執行環境，就可以執行 Java 所撰寫的程式。

近年來 iPhone、Android、BlackBerry 的誕生讓原本的 J2ME 更遭到許多的挑戰。iPhone 的開發環境最為嚴謹，必須在 MAC 的平台開發且使用 Objective-C 語言來開發[7]。Android 和 BlackBerry[6]的開發環境，可以在 Eclipse IDE 上開發，但是 Android 卻不支援 J2ME 所開發出來的系統，需將原本 J2ME 所開發的程式碼轉換成 Android 程式碼才能進行後續的開發[12]。反而 BlackBerry 支援 J2ME 的開發[11]方式。由上述看來，應用程式的開發的確會造成開發人員的困擾，不

同的設備有不同的開發環境，但大致上開發的工具，仍可以使用 Eclipse 的開發界面來支援大部份手持式設備的開發。如：J2ME、Android、BlackBerry。Windows Mobile 使用 Visual studio 開發，但利用 CLDC/MIDP 仍可以在 Eclipse 上開發後再發佈到 Windows Mobile 執行。Sun 公司也宣佈的 J2ME 已支援 iPhone 的使用[13]。

2.3網頁程式的開發環境及Web Service

僅管 J2ME 應用了 CLDC/MIDP 和 KVM 的技術解決了平台性的問題，以及標準語言網頁式開發方式，只要設備有瀏覽器就可以執行程式，解決了應用程式開發上的不便。但在網頁上的開發延伸了另一個問題，那就是瀏覽器對於網頁語言的解析問題[17] [18]。web Service 的興起使的開發人員使用大量的 HTML、XHTML、AJAX、XML 等網頁的技術以加快存取的速度。早期使用 AJAX 所運用的 XmlHttpRequest 以及 JavaScript 的系統，不是所有瀏覽器都有支援的。但是近年來由於瀏覽器使用頻繁，促使了各家瀏覽器的技術都支援 AJAX。但另一個 plug-in 的技術，仍有一些瀏覽器不支援的，如：iPhone。這也使得 Adobe 必須發展配合 iPhone 的 Flash professional CS5[15]來支援 iPhone，以服務使用者。加上螢幕大小的限制，一般來說目前手持式設備都是先載入整個網頁，再讓使用者點選要瀏覽的區域使之放大整個區塊瀏覽。這樣的技術影響 HTML 或 XHTML 的

語法的配置，這方面的研究也有相關的技術[16]。也因為在網頁開發上的種種困擾，2004 年由 Apple, Mozilla, 和 Opera 組成了 Web Hypertext Application Technology Working Group (WHATWG)組織並提出 HTML5 草案[42]，此草案也於 2007 年被 W3C 所接納並著手製訂 HTML5 的規格書[10]，以此想解決 HTML4 的不足。早期 Google 開發 gears 的技術發展出網頁離線的操作，運用在 mobile Gmail 和 mobile Calendar[8]。而 Google 在 Android 2.0 也將 HTML5 的離線功能加入，可想 Google 對 HTML5 的重視[43] 。未來 Google 也打算採用 HTML5 的 offline 功能，來取代 Google gears 的技術[32]。因此我們也相信未來手持式設備都將支援 HTML5 的功能，這不僅讓設備不用隨時連上網以節省支出，也減少了頻寬的使用。

HTML4 與 HTML 5 不同的地方[41]，從一開始宣告 HTML 的文件就已經有所不同。HTML4 的宣告是非常攏長而不易記住，在 HMLT5 已變為簡短易記的<!DOCTYPE html>。在新的元素(Elements)上新增了<section>、<header>、<footer>、<nav>、<article>、<canvas>、<video>、<audio>、<embed>等元素，另外也新增、改變了一些元素的屬性(Attributes)。在 API 上增加了 2D 的圖形像<canvas>元素和<video>、<audio>元素的影音播放，以及網頁 offline 的應用和 Cross-document messaging [30]。但是以目前手持式瀏覽器而言，並不

是全部都支援 HTML5 的標準，像 Opera 9.5 不支援 offline 功能但支援 Cross-document messaging 功能[14]，Android 2.0 瀏覽器支援 offline 但不支援 Cross-document[31]。不過以目前的趨勢不難看出，未來不管是桌上型或手持式設備的瀏覽器上，都將支援 HTML5 的語言，這將讓開發人員減少開發時間，也減了測試時間。

2.4服務導向架構Service-Oriented Architecture, SOA

Service-Oriented Architecture, SOA[24]的架構是近年來最受矚目的架構，從企業應用整合(EAI)[25][44]、企業與企業間的整合(B2B)[26]以及企業流程管理(BPM)[27]，大致上都漸漸的採用 SOA 架構而運用在企業、政府、醫療、數位家庭等各式的服務，但如同 IBM Redbooks 所描述的“Service-Oriented Architecture is not a new notion”。在早期 IBM WebSphere MQ 及 J2EE 的 JMS 就是以訊息系統(messaging systems)的傳遞，以達到 service-oriented 的概念。近年來因為 web Services 的使用和 IT 面對企業的分散性架構、異質性系統及不斷需求的改變，導致 SOA 的架構被多數企業所採用。原因在於 SOA 強調低耦合(Loosely coupled)的界面開發，使的開發時可將功能元件化(Component-based)。且現今的企業，大部份都是分散式的架構，來部署企業的內部資源或者使用外在提供的外部資源。使的 SOA 在這樣的環境很容易被用來整合內、外資源及異質性整合的參考架構。加上

web service 技術的成熟，讓客戶端(Client side)可以透過各種標準協定如：WS-I 所定義的 HTTP、XML、SOAP、WSDL、UDDI、JMS 對服務端(Server side)進行服務需求的請求(Request)。底下我們將說明 SOA 的運作方式及 SOA 相關的技術運用。

2.4.1SOA架構說明

圖 2.1 顯示了 SOA 的運作方式。當服務消費者 (Service Consumer)向服務註冊(Service Registry)提出某項服務時，服務註冊(Service Registry)會依照所發佈的服務紀錄中尋找該項服務，如果服務註冊(Service Registry)中有該項服務，則提供該服務的相關訊息及介面，讓服務消費者(Service Consumer)可以連結(Bind)到該服務的提供者(Service Provider)以使用該服務。各自的角色說明如下:

- ✓ 服務消費者(Service Consumer)：本身是一個應用程式、軟體模組或者是另一個服務。
- ✓ 服務提供者(Service Provider)：具有網路位址的實體(Network-addressable Entity)，執行消費者的請求。服務提供者發佈界面和限制條件到服務註冊表(Service registry)中，讓服務消費者尋找服務並可執行服務。
- ✓ 服務註冊(Service Registry)：服務註冊(Service Registry)能讓服務消費者(Service Consumer)尋找所需要的服務，它也負責接受與存

取來自於服務提供者(Service Provider)的合約。

角色之間的操作說明如下：

- ✓ 發現(Find)：服務消費者經由在服務註冊中尋找所需的服務，而在尋找過程中是有合約的限制。
- ✓ 發佈(Publish)：一個服務是經由提供者所發佈到服務註冊中，如此才能讓服務消費者探尋及呼叫使用。
- ✓ 繫結和執行(Bind and Invoke)：在服務註冊中探尋到所需的服務描述後，消費者服務描述的資訊連結到提供者並呼叫執行服務。

服務導向中服務的定義：

- ✓ 服務(Service)：一個服務是透過已發佈到服務註冊並提供界面(Interface)讓消費者取得並使用。
- ✓ 服務描述(Service Description)：一個服務必須由提供者描述該服務的取得路徑，並說明請求和回應的格式，這些的描述可以設定存取條件、使用協定或者服務品質等級(quality of service ,QoS)。

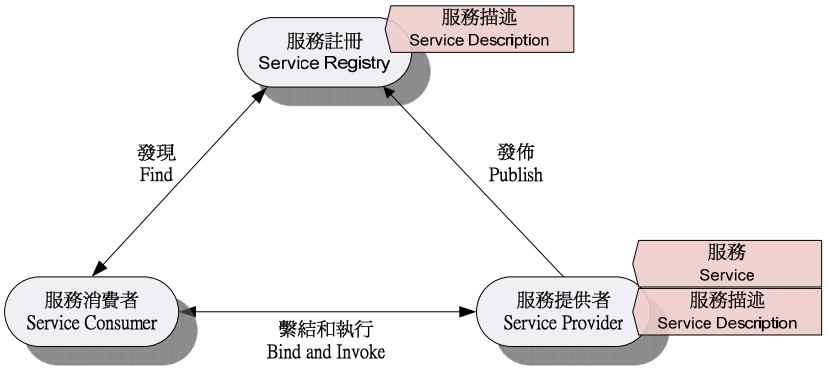

圖 2.1 SOA 的運作

2.4.2SOA 和 Web service

先前我們談過 SOA 並不是一定要使用 Web service 來做訊息的溝通，早期 IBM WebSphere MQ 及 J2EE 的 JMS 都可以做為 SOA 訊息傳遞的方式。面對異質平台以及企業對企業的溝通如果採用 MQ 或 JMS 都較限制在特定的語言上，以致企業採用 Web Service 的方式做為 SOA 架構的溝通。當然如果針對現有系統採用 JMS 或 MQ 的整合，SOA 的 ESB 也需能夠相互的傳遞 JMS 或 MQ 的訊息。圖 2.2 說明 SOA 如何運用 Web service 來運作

(A) 服務提供者(Service Consumer)發佈服務到服務註冊表(Service registry)，而此服務註冊表可為 Universal Description, Discovery and Integration, UDDI 的描述。

(B) 服務消費者(Service Consumer)向服務註冊發出需求的請求。

(C) 服務消費者取得服務的描述。

(D) 服務消費者依服務描述連結到網際網路並透過 Web Server 取得服務提供者的位址。

(E) Web Server 協助服務消費者取得服務提供者的連線並使用服務。

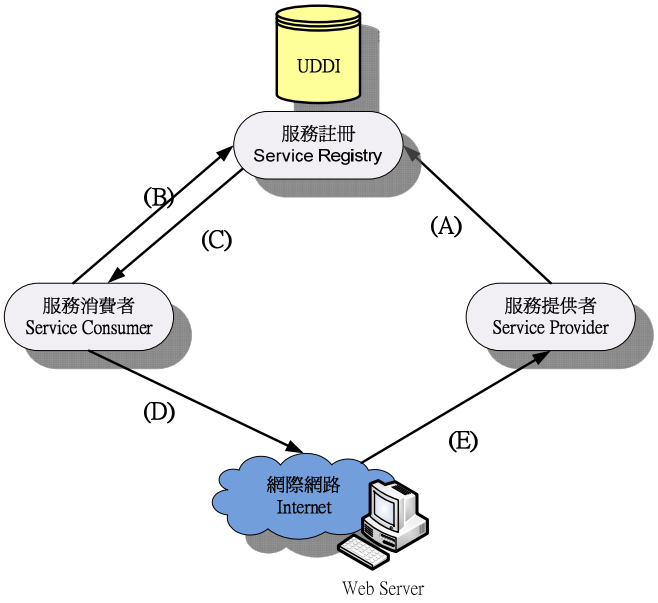

圖 2.2 SOA 與 Web Services 的運作

2.4.3SOA和ESB

SOA 中另一個重要的核心，也就是提供協助客戶端和服務端的 Middleware；Enterprise Service Bus, ESB。ESB 在 SOA 架構下除了，提供客戶端尋找它所需的服務以及將服務端的服務提供給客戶外，另一方面 ESB 也負責了客戶端和服務端傳送訊息的轉換、提供客戶端正確的服務和客戶端與服務端協定的轉換。依照 IBM Readbooks[28]，定訂了 ESB 較完整的所需能力，如：通訊(Communication)、服務的相互影響(Service interaction)、整合(Integration)、(Quality of service)、安全(Security)、服務等級(Service level)、訊息處理(Message Processing)、管理和自主性(Management and autonomic)、建模(Modeling)、基礎建設的思考能力(Infrastructure

Intelligence)。另外也定訂了基本的 ESB 能力如表 1.1，以及圖 2.3ESB 在 SOA 中所扮演的角色。

表 2.1 ESB 的基本能力

	能力	理由
通訊能力	路徑選擇 定址能力 至少要有一組訊息格式 (request / response, pub/sub) 至少要有一個傳輸協定	提供服務位置的透通性 和服務的可替代性
整合能力	服務整合 協定轉換	支援異質性的整合和支援服務的可替代性
服務(Service)的相互影響	服務(Service)界面的定義 服務(Service)訊息模式 服務的可替代性執行	支援 SOA 架構的原則，將應用程式中分離出服務協定和執行方式
管理和自主性	管理能力	管理服務的定址和命名

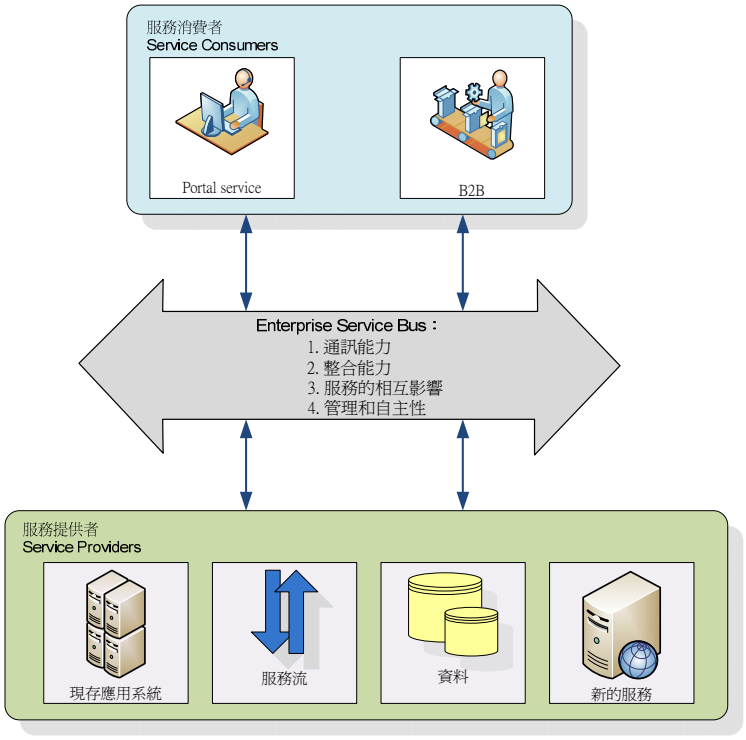

圖 2.3 SOA 與 ESB 的相互作用

ESB 產品的服務大致上也都整合了原有的 web server 的服務，像是 Microsoft 的 BizTalk、IBM 的 WebSphere、Oracle 的 SOA Suite、Sun 的 Java CAPS。Open Source 的有 Mule、Sun 的 OpenESB、Jboss 的 Jboss ESB、Eclipse 的 STP 等。

2.4.4SoaML Profile

SOA 的架構方法，定義了人、系統、組織，三個角色如何去使用一個服務。而 SoaML Profile[33]延展 Unified Modeling Language, UML(Extension UML)，提供一個標準的設計和塑模的方式，來描述服務以支援 SOA 的架構。使用 SoaML Profile 能讓企業導向和系統導向的服務彼此相互合作，以支援企業所需要的系統。也就是說，假如

我們採用 MDA 的方法，目的是要將企業和系統架構分離以進行分析，那麼 SoaML Profile 就是企業和系統的導覽地圖，用來描述和設計企業架構，並且讓開發系統使用，以實現 SOA 的架構。

在軟體開發方式上，我們常採用 Top-down 或 Bottom-up 的方式[49]來進行軟體的開發。Top-down 的方式是將一個完整的系統拆成數個 subsystem，每一個 subsystem 再拆解成最小的元素或元件。Bottom-up 是從最小的元素或元件著手進行組合成子系統，再將子系統相互結合成一個完整的系統。此方式應用在 SOA 上，在 Bottom-up 方法可以將服務定義成是一個基本的服務(base service)，利用 SoaML Profile 來描述服務的規格、運作方式，或者與其他服務相結合。在 Top-down 方法可利用 SoaML Profile 的服務架構模型(services architecture model)，來顯示各個參與者(participant)是如何提供、使用服務，而達到互相工作的企業的流程。

在 SoaML Profile 所描述的基本服務(base service)是一個 UML 的 interface，而讓參與者(participant)使用服務。服務可能由一個或多個所組合而成，因此在多個服務中相互的運作時就會有條件限定(constrain)或協議(protocol)來指定服務的運作方式，進而讓參與者(participant)來實現(implementation)服務。底下我們將介紹 SoaML Profile 中較為重要的 stereotype 說明，未提及的可參考文獻[33]說明。

- ✓ Service：利用 stereotype 為<<interface>>的 UML interface，表示一個服務。Service 提供操作(operations)的能力讓參與者(participants)利用 ServicePoint 或 RequestPoint 來提供服務或需要額外服務需求。
- ✓ ServiceInterfaces：UML 的 interface 或 class。ServiceInterfaces 有附加的一些特性，可以讓提供者和消費者接收或傳送訊息。主要有以下三點：
  - ■ 提供者和消費者的介面：提供者和消費者是經由 Service Interfaces 來被實現或被使用的，而被實現的提供者將提供他的能力給消費者使用。
  - ■ ServiceInterface class：扮演參與者(participant)的角色來實現(realized)服務提供者和使用(use)消費者。
  - ■ 行為的協議：規範了提供者和消費者相互之間的協議(protocol)。
- ✓ participant：可代表提供者、消費者、軟體的元件、組織、系統或人。且 participant 的 servicePoints stereotype 表示真實的服務提供，requestPoint stereotype 來表示需要額外的服務或參與者需求。
- ✓ servicePoint：經由 participant stereotype 提供一個服務到客戶端。
- ✓ requestPoint：經由 participant stereotype 請求另外一個服務，以提

供 participant 的服務需求。

- ✓ ServiceContracts：定義了服務提供者和消費者之間的協議(protocol)約束條件，讓 participant 遵守約束條件。
- ✓ ServiceChannels：在消費者需求和提供者服務間提供一個通訊的路徑。
- ✓ ParticpantArchitectures：一個 high level 的服務架構，定義了內部或外部一組 participants 之間如何使用服務達到相互工作的服務架構圖。

2.5簡介MDA

Model-Driven Architecture, MDA[19]是由 OMG 所提出的架構，該架構可被用在各種不同的領域上像 F-16 噴射引擎軟體、銀行的付款系統、醫療。圖 2.4 顯示了 MDA 使用 UML、MOF、CWM 三項的標準建模工具，並在開放的或私有的不同的平台.NET、Java、Web Services、CORBA、XMI/XML，可以互相的由 PIM 轉成 PSM 或者 PIM 互轉 PIM，而形成新的 PIM 再轉成 PSM，進而在不同平台上執行。依此可應用在跨平台的服務如：Security Services、Directory Services、Transaction Services 等，並且用在不同的領域，如：金融、電子商務、電信、運輸等領域。

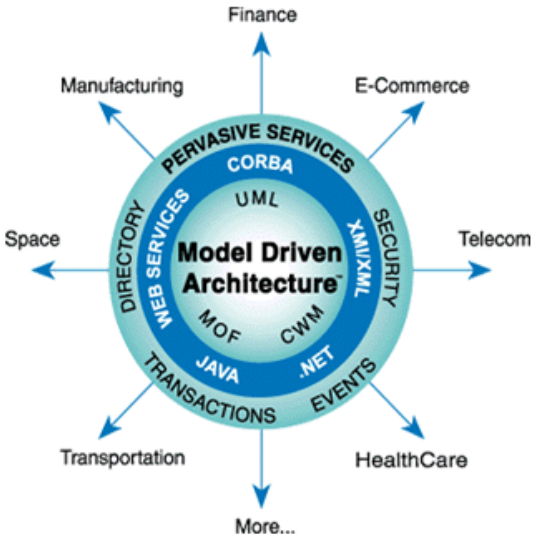

圖 2.4 MDA 架構[34]

MDA 的架構文件，主要在說明將系統可從三個觀點(view)來進行分析，而且三個觀點是相互獨立，也讓系統架構師與程式開發人員可以獨立於專職之上進行分析和開發。而這三個觀點分別如下：

✓ Computation Independent Model, CIM：

著重於系統的環境及使用者的需求，但不詳述系統內部的結構和運作的細節。又稱為領域模型(domain model)，由特定領域的專業人員進行需求的分析，並詳述其需求。

✓ Platform Independent Model, PIM：

著重於系統內部的結構、組成，但不去詳述實作的平台。

✓ Platform Specific Model, PSM：

著重於 PIM 轉換成特定的平台，包含程式碼、平台部署的描述檔或其他設定的檔案。

當使用 MDA 將系統各自獨立成三個觀點(view)，OMG 也提供了各種模型標準工具讓使用者參考：

- ✓ Unified Modeling Language ,UML：利用視覺化的模型來表示系統，以提高系統的可讀性。
- ✓ object constraint language ,OCL：對 UML 模型做描述性限制條件的撰寫，也可以提供在模型轉換時的限制條件。
- ✓ Meta Object Facility ,MOF：提供 metadata 的管理框架並提供模型的互換系統。
- ✓ XML Metadata Interchange ,XMI：提供在 MOF 框架內模型轉換的標準語言且為 XML 格式。此一格式可讓模型在不同的系統上做交換。
- ✓ Common Warehouse Meta-Mode ,CWM：包括 data repository integration、the representation of database model、schema transformation model、OLAP 和 data mining model 等標準，提供資料格式的轉換。

2.5.1MDA 開發模型與人員角色

在 MDA 的架構中，開發人員的角色與傳統上相比，雖然角色的分類大致相同，但是在 MDA 的架構上角色的責任卻有不同。OMG 的另一篇文章中[35]，清楚定義了在 MDA 程序中參與人員所扮演的

角色及責任。

- ✓ 需求分析人員(Requirements Analysts)：以人為驅使(people-driven)的方式進行系統的分析。分析人員必須充份了解需求者的需求，並精確的轉換成 UML 模型以利開發程序上的使用。並且模型也可以用來驗證和測試，以利開發程序不會被誤解。

  需求分析人員可繪製的 UML 圖：activity diagrams 和 use case diagrams。

- ✓ 分析設計人員(Analyst/Designers)：進一步更精細的分析企業邏輯需求，並繪製其模型。而這些模型是將來會被轉換成其他模型，或者是最後可執行的程式碼，因此分析設計人員必須精確的描述以減少後續的修正。

  分析設計人員可繪製的 UML 圖：class diagrams, state chart diagrams。

- ✓ 架構師(Architects)：系統整個結構的決策者，並在整個 MDA 流程中決定採用塑模方式和對映轉換的方式，並可以參考過往專案所使用的經驗，重覆使用於進行中的專案。

- ✓ 分析設計、程式設計師(Analyst/programmers)：該人員著重兩項工作，一是將系統轉換成抽象的程式或者套用現有的 pattern，如：QVT、Gang of four, Gof pattern，而不是專業的寫程式人員。另一

個工作是利用 MDA 工具，開發可轉換的 pattern 以用來對映轉換成目標的程式，並在轉換對映過程中必須做轉換的調整。

另外程式碼的產生，也是程式設計師來撰寫程式產生的樣板(template)而直接產生目標程式碼。經由程式設計師使用的模型樣式和程式產生的樣板，可以整理並加調整規則，以利架構師在專案上適當的選擇利用。

- ✓ 測試人員(Testers)：利用工具產生測試案例進行單元測試、整合測試。並且訂定測試的標準，來評量系統的穩定度。
- ✓ 維護人員(Maintainers)：維護人員對於使用中的系統必須隨時注意版本的更新，以及文件的更新。另外維護人員也必須維護轉換模型和設計的規則，而不是直接修改程式碼，因為當客戶平台改變時，可以利用維護的模型進行對映轉換，以減少直接手寫程式的時間。

由以上可以了解到，傳統開發模式的參與人員和 MDA 方式的人員較大的不同，在於程式設計師人員。以往程式設計師所擔任的是程式的執行，利用目標語言(C#、Java)直接撰寫程式，但以 MDA 的角度，程式設計人員著重在於模型轉換的規則和程式產生的樣板，盡量少與目標語言接觸。這樣的構想於目前手持式設備在開發邏輯性的程式有相當大的幫助。我們可以假想，在分析設計階段經由 class diagram

所設計出來的邏輯性，假如沒有 MDA 轉換的觀念，那麼程式設計師必須將依照 class diagram 直接撰寫目的程式。在早期 Eclipse 上已有轉換成 Java 程式碼的工具，但是，如果對映平台是 C#、VB.Net 那麼程式設計師只好再將此邏輯性寫成另外兩種語言。這樣不僅消耗程式設計師的開發時間，也增加了專案的時程。

2.6研究總結

由以上的研究我們可以知道，SOA 的架構已被多數企業所採用，不管在企業上的部署或者開發人員在系統的開發模式。SOA 的架構可以整合分散式架構，以及透過標準協定提供快速的服務，並且將服務元件化而達到重覆使用的效果。在部署 SOA 的架構 ESB 為其核心，ESB 的使用、及開發的整合如果採用套裝軟體如：Microsoft 的 BizTalk、Oracle 的 BEPL、IBM 的 WebSphere，對企業而言都是一筆可觀的成本，但相對的，支援週邊的服務也較為完整。另一重點為 QoS，經由 QoS 的制訂，提供了服務的等級及安全性，讓使用者可以依此為考量並選擇使用服務。如：Llama 計劃(Intelligent Accountability Middleware Architecture)[29]，採用了 Mule ESB，讓 ESB 可以經由使用者的設定，而達到 ESB 的自行監控、管理並執行在 SOA 的架構上。並且 Llama 也強調了 4 D’s(Detection、Diagnosis、Defusing、Disclosure)經由偵測、分析、去除、持續監控四個方式來讓每個服務可以受到監

控並提供更完整的服務。

在服務的探尋(Discovery)上，也有強調 Lightweight Web Services Architecture，說明在使用 XML 訊息傳遞時將訊息壓縮而不解壓縮，以直接取得壓縮的內容以此減少了頻寬的使用[39]。也有使用類似社群中介(Community Middleware)觀念的 Lightweight application server, LAS，可以管理使用者，在有等級制度下取用可使用的服務。並且 LAS 的方式也經由，透過 ESB 和使用傳統 Web Services 的兩種方式比較，證實了 ESB 的方式會優於傳統的 Web Services[40]。在平台的相異性或手持設備的相異性，MDA 的轉換方式也可以部署在不同平台，產生相對的程式碼[38]。

在 SOA 結合 MDA 方面，SOA 應用了 Component-based 的方式，利用 MDA 的三個觀點來開發，不僅能讓我們在 SOA 所強調的服務消費者和服務提供者各自獨立分析外，也能讓服務提供者端所塑造的模型，以 XMI 格式的交換而和其他模型整合成新的服務[37]。也因此我們在下一章節所提出的開發流程，除了綜合 MDA 開發的三個觀點外，並且也採用 SoaML Profile 來塑模各個服務之間的整合，最後採用 MDA 方式中的 PSM，決定最後部署平台並執行轉換模型到程式碼的方式，來節省同一個應用軟體在不同環境的設備上可能必須重新打造的情形，以節省開發時間並可以使用 SoaML Profile 重覆使用 SOA

上的各項服務而產生新的服務。

第3章 開發理論與流程

這一章節我們將使用 MDA 的方式，並使用 UML2.0 及 SoaML Profile 做為系統分析與設計的參考規範。在塑模的工具上，我們採用 Enterprise Architect, EA UML2.0[45]工具，來繪製各項的 UML、SoaML Profile 等模型圖。並使用 Jboss ESB[48]做為 SOA 架構的 Middleware。在此我們提出預先定義好的 MDA 的開發流程，依照 OMG 提出的 MDA Guide[19]，以及參考文獻[47]所提出的開發流程，我們歸類 MDA 的開發流程如下 6 個步驟，並且配合 SOA 架構，取名為“SOA 應用 MDA 的開發流程”，說明如下：

1. CIM 階段：經由與客戶需求訪談後，匯整需求並做 domain 的分析，並且條列使用者需求清單，以 use case diagram 和 active diagram 來表示企業架構的需求。
2. PIM 階段：經由 CIM 階段後，系統架構著重於系統功能與功能之間的關係，或者系統與其他系統之間的關係，細部的分析使用者需求的各項功能，以模型方式清晰的呈現系統架構，並且可以採用 UML 的 OCL、Tagged values 來表示關係及限制條件。最後可繪製 class diagram、state diagram 來表示系統功能架構。
3. 進階 PIM 分析：經由上一步驟 PIM 設計之後，進階 PIM 分析階段

是轉換成 SOA 設計方式，以萃取出系統或功能中的服務，讓服務可以重覆的使用。

3.1. 服務需求描述：進一步分析系統中的各項功能，抽象的表示出服務需求，包含服務定義和服務所含的操作，或者可從企業中現有的服務，重覆使用服務的架構，並增加其所需額外的操作。

3.2. 服務的規格：利用 ServiceInterface 來表示提供者與需求者，並透過協議(protocol)來約束功能之間的關係運作。Protocol 可用循序圖(sequence diagram)來表示。

3.3. 服務的實現：加入 participant 完整表示服務的實現，並透過 servicePoint 和 requestPoint 來表示提供服務和需求服務。

3.4. 部署服務：將最後的服務轉換成程式碼並部署在對映的 ESB 上。此步驟和步驟 3.3，可採用 MDA 的 PSM 觀點方式來轉換，以對映到不同平台。

4. PIM 轉換 PSM(s)：修正原本 2.PIM 階段中的 class diagram，採用服務的觀點來結合待開發的系統，以產生最後的系統架構，並驗證架構與系統需求是否吻合。最後選擇實現的平台，轉換的 patterns、templates 的選擇及調整，其轉換到對映的平台架構。
5. PSM 轉換原始碼：產生原始碼，並且執行後續專案系統程式的調整。UI 的設計、UI 與服務的接合，以及相關的單元測試、測試案

例產生。

6. 部署系統：將最後開發完成的系統部署於平台上，並紀錄相關的設定、部署文件。
7. 整合測試：針對最終的系統，設定測試評量標準和測試方法，以及進行整合的測試、測試評估，並且產生相關文件。
8. 系統維護：針對系統的問題來做系統、pattern、templates 的調整維護，並且更新原有文件。

上述是我們提出 SOA 應用 MDA 的開發流程，此流程的方法是針對了 MDA 的三個觀點和 SOA 的架構所設計的，其中在“3.進階 PIM 分析”著重在於 SOA 架構的服務分析，採用 SoaML Profile 的方式來分析系統中的服務，以達到 reuse 的目的。因此，在“進階 PIM 分析”階段，我們再將分成四個子步驟執行，且在這四個子步驟的進行，仍可以採用 MDA 的方式進行，因為一個服務的產生，並不是只有一個系統可以使用的，我們應考慮服務的擴充性，也就是在服務的分析時，考慮服務的完整性問題，不應該只為單一系統而分析。這觀念在企業中是必須考量的，否則同一個服務會被複製多種版本，而應用在不同的開發系統中，增加了系統維護的困難。

如同 MDA 三個觀點，對於單一服務的分析設計。在 CIM 階段，著重在服務 domain 的分析，例如，股票系統中股票報價服務。我們

可以很簡單的將服務，再拆解成股票報價的顯示服務和資料庫的連線服務。PIM 階段，對單一的服務進行完整性分析，而不要受限在單一系統中使用，例如，在資料庫連線中，我們可以考量加入其他資料庫連線，而不是受限在單一資料庫連線服務。在 PSM 階段，考量服務部署的平台，可以針對企業習慣的方式來部署，另外，同一個服務也可以部署在不同的平台上。

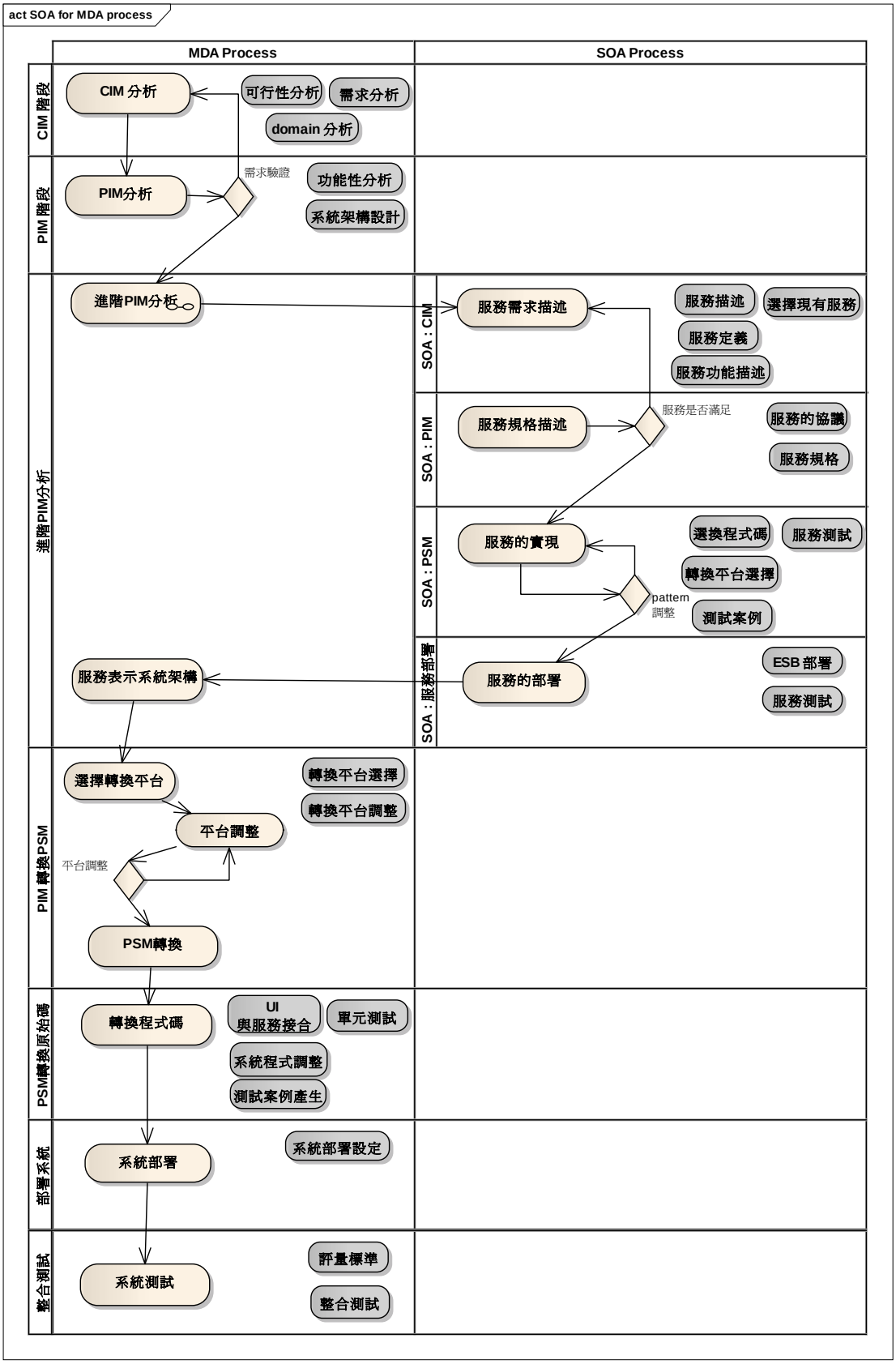

圖 3.1SOA 應用 MDA 的開發流程圖

上圖 3.1 表示了“SOA 應用 MDA 的開發流程”的流程圖，圖中我們將開發軟體分成兩個 Process，一為 MDA Process，另一個為 SOA Process。在 MDA Process 呈現了一般系統開發所套用 MDA 方式的流程。從 CIM 階段，針對系統需求，進行可行性分析、企業領域的分析。PIM 階段，進行了系統上的架構設計，進而到各功能的分析、設計。進階 PIM 分析，是從 MDA Process 轉換到 SOA Process。其目的是將我們要開發在系統中的功能，獨立出來分析而成為服務。因此在 SOA Process 的流程上，服務需求的描述，相似於 MDA 中的 CIM 階段。對服務進行描述和定義，且針對現有的服務，也可以進一步的採用在進行的專案上。服務規格描述，相似於 MDA 中的 PIM 階段，定義了服務的規格，以及服務和服務間如何溝通，透過協議(protocol)來表示。服務的實現，相似於 MDA 中的 PSM 階段。模型的轉換，產生原始碼、測試案例產生。最後的服務的部署，將服務部署在 ESB 上，供服務消費呼叫使用。流程再回到 MDA Process，將開發系統轉換成由服務所構成的模型，以完成整個待開發的系統架構。流程到 PIM 轉換 PSM 階段，進行了平台的選擇，及平台的調整、轉換。PSM 轉換原始碼階段，產生目的程式碼和 UI 結合，並做系統程式面的調整，產出可執行的系統，並且產生測試案例，進行單元測試。部署階段，選擇部署的平台，設定部署參數，使之完整。整合測試階段，設

定系統的評量標準，以進行整合測試。

另外我們也先架構出我們的 SOA 與 ESB 的架構。圖 3.2 說明了我們的架構想法，在整個架構想法上，建構企業內部(Intranet)和企業外部(Internet)的網路，且各自提供服務給不同對象的消費者。

- ✓ 企業內部網路/Intranet：在內部建構 SOA 並提供一個服務池(services pool)存放各式各樣的服務，該服務可能只是程式的框架，或者是一個完整的服務並發佈、註冊在企業內部的 ESB 上，提供系統開發者的使用。
- ✓ 企業外部網路/Internet：在對外建構 SOA，內部開發人員將開發完成的系統發佈、註冊於企業外部的 ESB 上，以提供服務消費者的使用。

服務池(services pool)的想法是來自 Connection pool、Object pool、EJB pool。我們可以將各式各樣的服務，採用適當的管理機制，如 UDDI，最後發佈、註冊在企業內部的 ESB 中。企業內部的 ESB 就像一個 pool，提供的對象是企業內部人員，人員可以依系統所需，在 pool 中擷取所要的服務，產生新的服務。這樣的方式，是需要有系統的管理服務，才不會讓服務氾濫而不受管控。

介於 Intranet 和 Internet 的中間，扮演了服務提供者和服務消費者兩種角色，當開發人員需要從 services pool 中取得服務，就扮演了服

務的消費者。當開發人員從 services pool 中取得服務，產生新的服務，再註冊回 services pool 中或者註冊完成的系統到外部的 ESB 中，則扮演了服務提供者。

下一章節，我們將假想一個系統，來做“SOA 應用 MDA 的開發流程”方式的案例。在案例當中，我們著重在開發流程方式，將不討論測試及維護的問題。

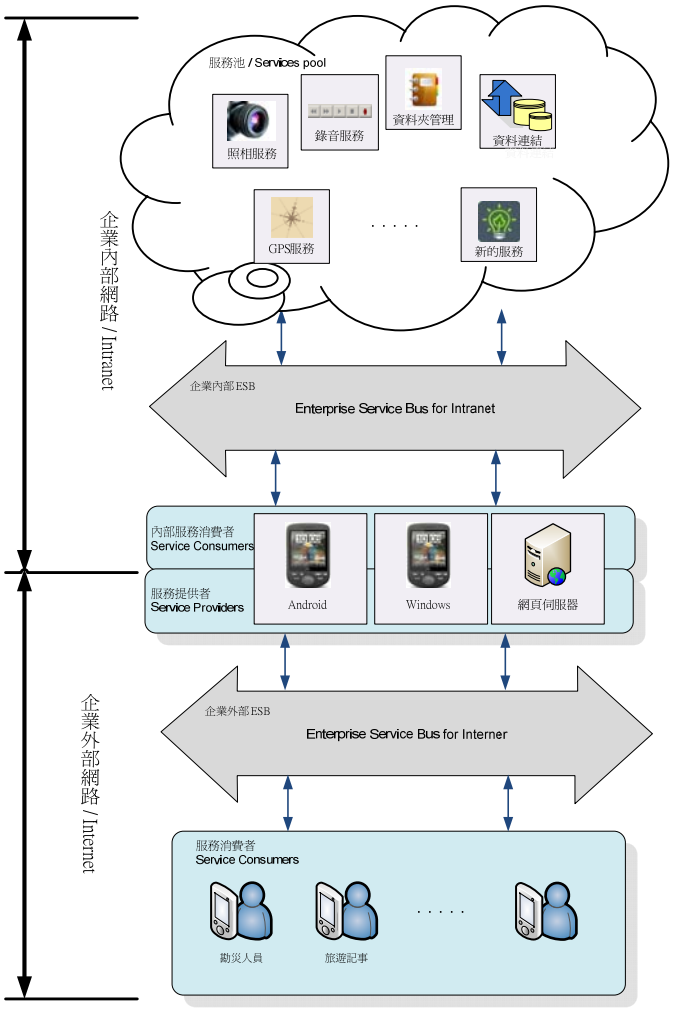

圖 3.2 SOA 的開發架構

第4章 SOA應用MDA的開發流程

4.1CIM階段

“computation independent mode, CIM，不探討系統架構的細節，著重於關心使用者的需求。又稱為領域模型(domain model)，由專業的人員進行系統的訪談。”

在 CIM 階段，我們針對所要進行的專案系統採用假想的方式，專案名稱為“LocationRecord”，專案的需求如下：

1. 使用者打開 GPS 以取得經緯度，並且紀錄該點位的資訊，另外，也可以操作錄音功能和照相功能。
2. 使用者可以經由所收集的點位清單，進行修改及刪除。
3. 使用者透過無線功能，將資料上傳至後端資料庫。
4. 使用者可以利用手機在地圖上瀏覽所匯集的資訊。
5. 使用者也可以透過桌上型電腦在地圖上瀏覽所匯集的資訊。
6. 此專案的需求及功能，我們可以應用在旅遊記事者或者做為勘災人員對現場的紀錄。

圖 4.1 為 LocationRecord 系統的 use case diagram。Survey Actor 可以操作 LocationRecord(位置紀錄)及 MapBrowser(網頁瀏覽)。在 LocationRecord 包含的功能 GPSLocation(GPS 定位)、textRecord(文字

的敘述)、SoundRecording(錄音)、Camera(照相)四個主要的功能。另外 Camera 也使用了 PhotoManage(圖片瀏覽管理)，SoundRecording 使用了 FileManage。在 LocationRecord 和 MapBrowser 兩 use case 都必須擁有一個 Store data 的資料庫在幫助檔案的儲存。

圖 4.2 active diagram 顯示 LocationRecord 的活動。其中 dataProcess 是處理資料的儲存，主要將錄音的聲音檔、照相的圖檔做檔案的儲存，因此我們另外表示了 dataProcess 的 subActive diagram 於圖 4.3 中。

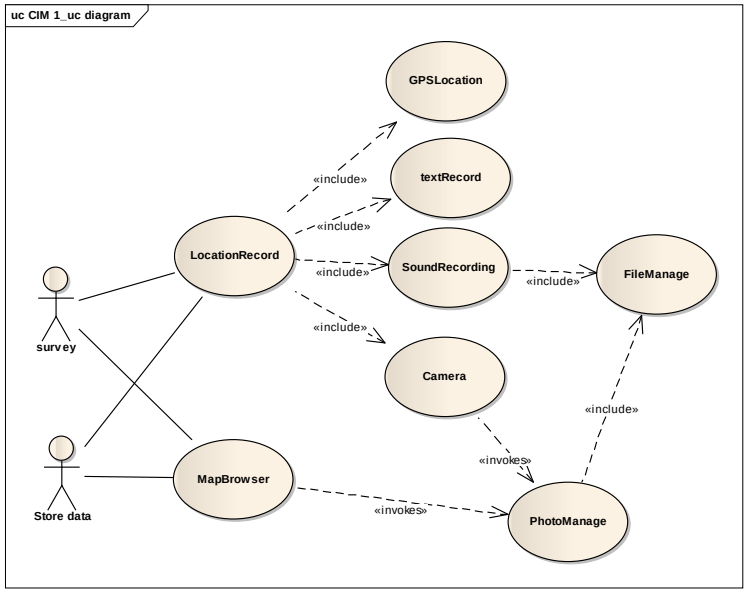

圖 4.1 LocationRecord 系統的 use case diagram / 使用者案例

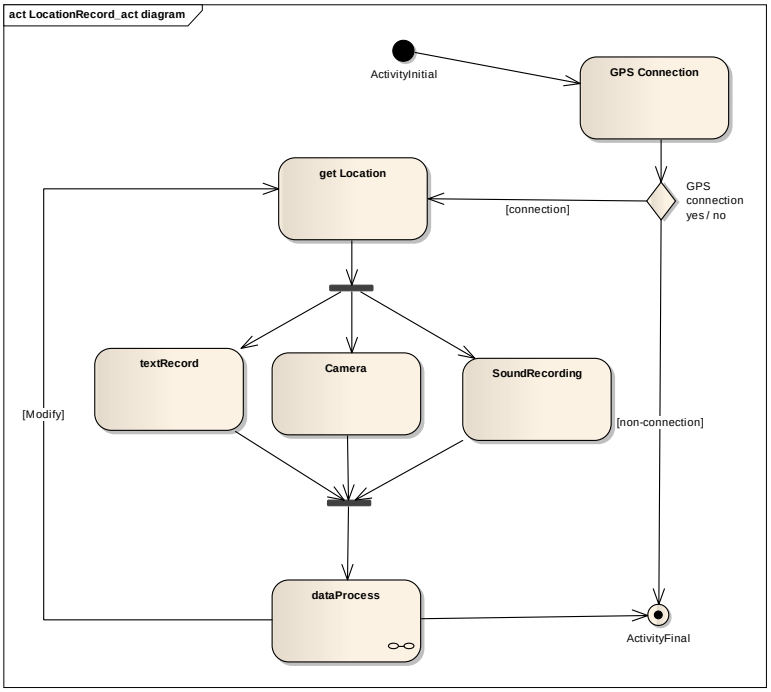

圖 4.2 LocationRecord 系統的 active diagram / 活動圖

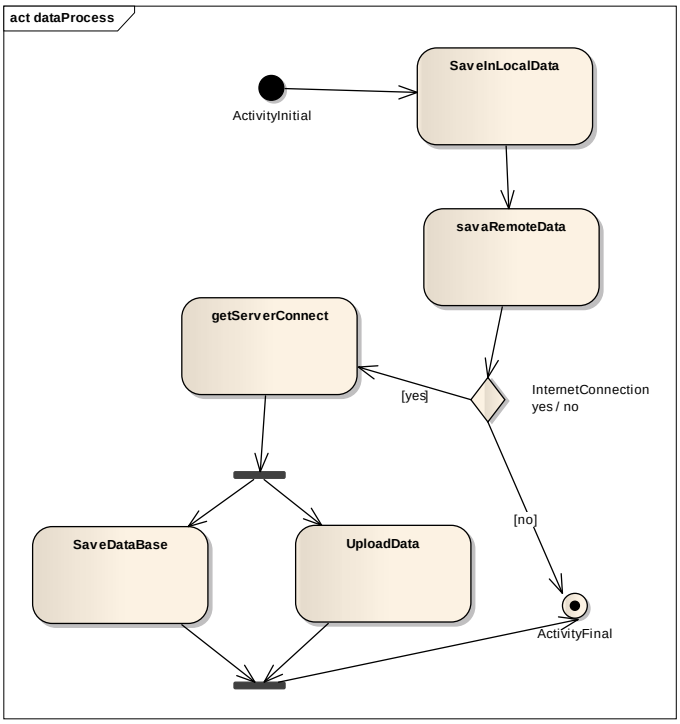

圖 4.3 dataProcess 的 subActive diagram / 子活動

4.2PIM階段

“platform independent model, PIM，描述系統的各項功能，但不探討最後所部署的平台。”

在 PIM 階段，我們所針對系統功能再深入的探索細節。我們將圖 4.1 中的 LocationRecord 和 TextRecord 共同表示為使用者介面(UI)設計上的表單，如圖 4.4 所示，LocationRecord class 為一個使用者介面(UI)設計上的表單，並以 “”stereotype 的圖示表示，含有的設計如：地點名稱、備註、或其他欄位。與 LocationRecord 有關的為四個主要類別 Camera 、 PhotoManage 、 SoundRecording 、 GPSLocation 和 FileManage interface 由 PhotoManage、SoundRecording 來實作。

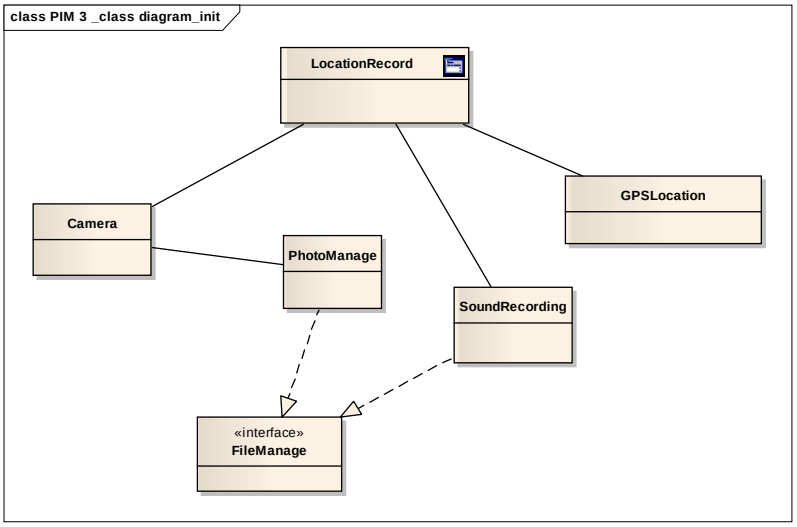

圖 4.4 class diagram

4.3進階PIM分析

進階 PIM 分析階段，我們考慮企業內部 Component 的重覆使用

性，因此採用 SOA 架構的，應用在開發上。另外在進階分析上我們採用 SoaML Profile 中的 stereotype，來標示 SOA 的執行服務。構想上是將圖 4.4 class diagram 進一步分析服務，如圖 4.5 表示，看似將 GPSLocation、SoundRecording、Camera、PhotoManage、FileManage 當作 SOA 上的服務，LocationRecord 為一個 UI 設計的表單(Form)，由上述各項服務組合而成。我們希望一個系統的表現，是 UI 的設計加上各項服務所組合而成。而當 LocationRecord class 不存在時，服務仍可以獨自存在。換句話說，當 LocationRecord 表單不存在時，其他各項服務 GPSLocation、SoundRecording、Camera、PhotoManage、FileManage 仍存在於 ESB 中，可以讓其他專案使用服務。

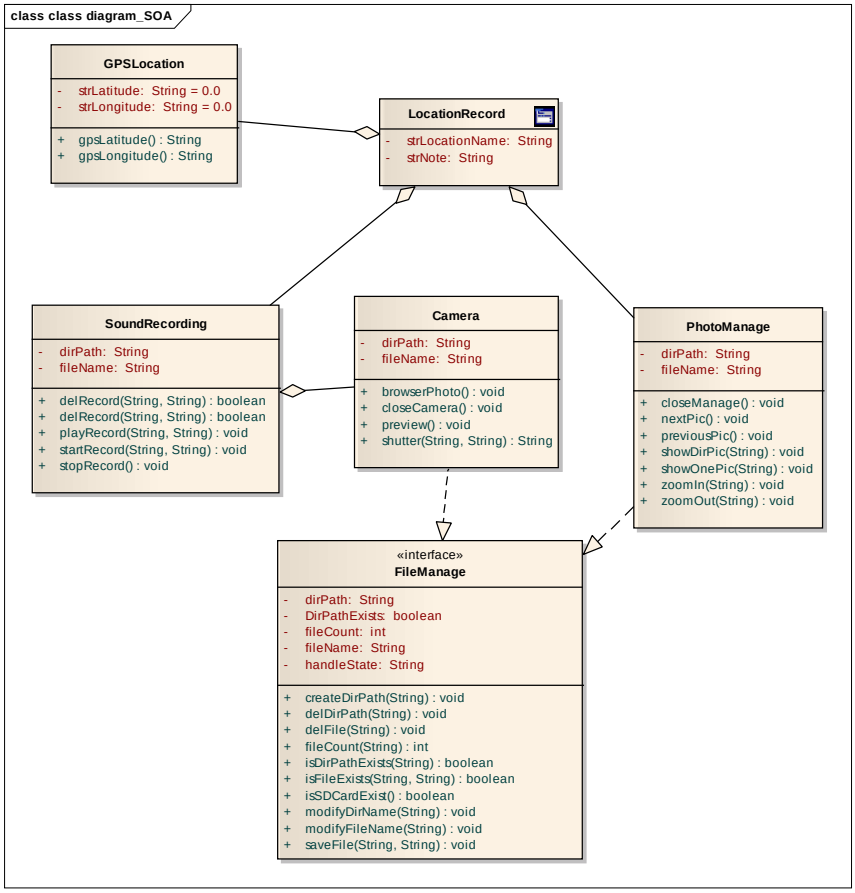

圖 4.5 組合 SOA 的 class diagram

4.3.1服務需求描述

當我們導入 SOA 架構的思考時，可以從 class diagram 來拆解各項服務，每一個 class 都可以當作一個服務，但仍需考慮重覆使用性的問題。以重覆使用性觀點來思考，可將 GPSLocation、SoundRecording、Camera、PhotoManage、FileManage 各自獨立為單一的服務。而服務與服務的結合也可以成為一個新的服務，例如，Camera 和 PhotoManage 兩個服務的組合可以成為一個具有照像功能且含有檔案管理的功能。或者 GPSLocation 也可以單獨成為一個服務

而應用在需要 GPS 定位系統之中。

這一小節我們使用了 SoaML Profile 來表示 SOA 架構中的服務。由圖 4.5 的 class diagram 中我們可以先擷取出服務，並且描述服務所需的 operation，以及服務之間的關係。圖 4.6 是我們設定五個服務 GPSLocation interface、SoundRecording interface、Camera interface、PhotoManage interface、FileManage interface，以及它們之間的關係，使用<<use>>的 stereotype 來表示。

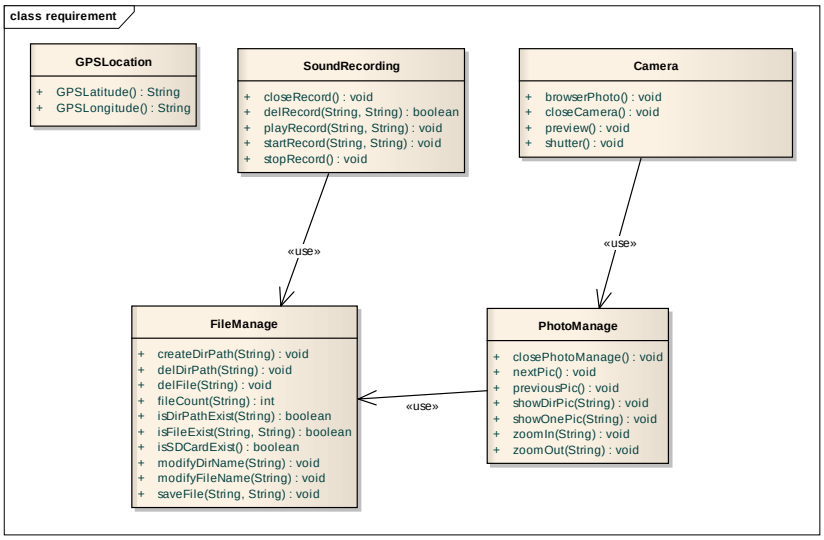

圖 4.6 GPSLocation、SoundRecording、Camera、PhotoManage、FileManage 服務與關係

在圖中，我們也可以看成是四個服務，來提供系統的服務。1).GPSLocation 本身是一個服務，2).SoundRecording 和 FileManage 組合的服務，3).PhotoManage、FileManage 組合的服務，4).Camera、PhotoManage、FileManage 三者組合的服務。從物件的角度來看

FileManage 可以是抽象類別(abstract class)或介面(interfact)，再由 PhotoManage 來繼承(inheritance)或實作(implementation)。但是，從服務的角度來看 PhotoManage 可能還包含了 zoom In、zoom Out 等功能，也因此在上圖中，我們將這兩服務分開來看待。另一個重點，由於 SoaML Profile 所定義的基本服務(base service)是由 UML 中介面(interface)來定義，而物件導向 Java 是單一繼承(Single Inheritance)的觀念，但可實做多個介面(interface)，也因此在圖中顯示的 FileManage 並不是抽象類別(abstract class)而是介面(interface)。

4.3.2服務的規格

在服務的規格中，首先我們先做服務的管理。每一個服務可以用 Package 來表示，並且也可以依照特定專案上的使用，來標示 Package 與 Package 之間的關係，如圖 4.7 所示，每一個 Package 都代表一個服務的管理，並且使用<<use>>來表示之間的關係。

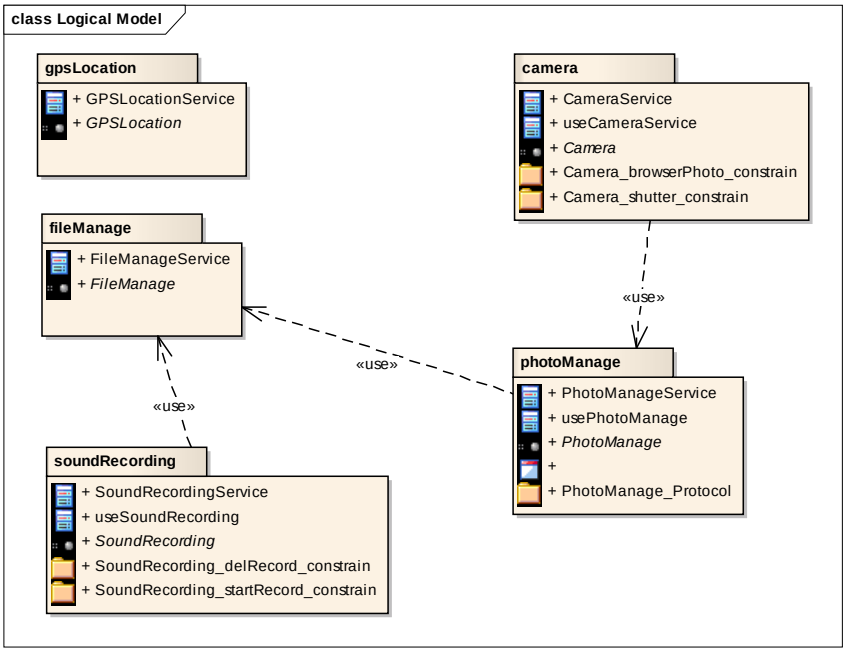

圖 4.7 Package 表示服務的關係

使用 SoaML Profile 中<<ServiceInterface>> 的 stereotype 是用來定義說明服務的細節，讓消費者知道如何使用服務，且也讓提供者知道如何執行(implement)服務。並透過服務的協議(protocol)來約束每一個服務的運作流程。在圖 4.8 中 stereotype 為<<ServiceInterface>>的 SoundRecordingService 服務，提供 SoundRecording interface 定義了 closeRecord()、delRecord()、playRecord()、startRecord()、stopRecord() 五個 operations 以提供服務。SoundRecordingService 服務需要另一個 FileManage 服務，來提供的服務的需求。

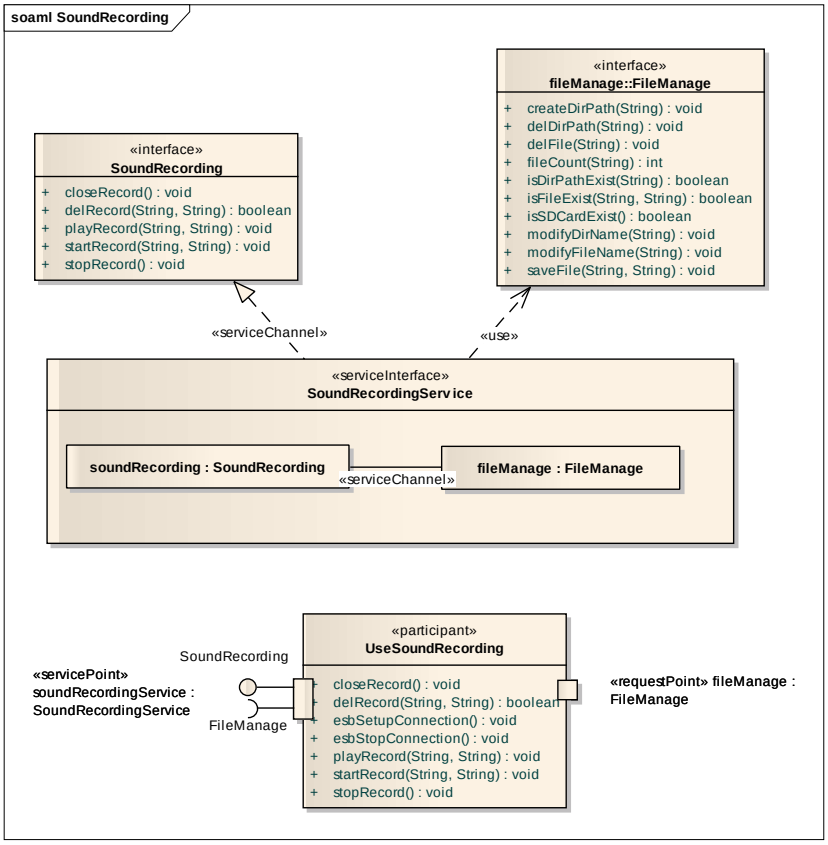

圖 4.8 SoundRecordingService serviceInterface

另外我們也定義了 SoundRecordingService 服務中 startRecord() operation 和 FileManage 之間的協議(protocol)以循序圖(sequence diagram)表示。圖 4.9 循序圖說明了，當服務消費者在使用 SoundRecordingService 的服務，使用 startRecord() operation，SoundRecording 會先要求 FileManage 檢查是否有 SD 卡的存在(isSDCardExist() operation)，如果回傳為 true，則 SoundRecording 再檢查要存檔的目錄是否存在(isDirPathExist() operation)，如果回傳為 false 則建立目錄(createDirPath() operation)，否則儲存初始檔案(saveFile() operation)並開始執行錄音(startRecord() operation)，直到停

止錄音(stopRecord() operation)，最後結束錄音(closeRecord() operation)。

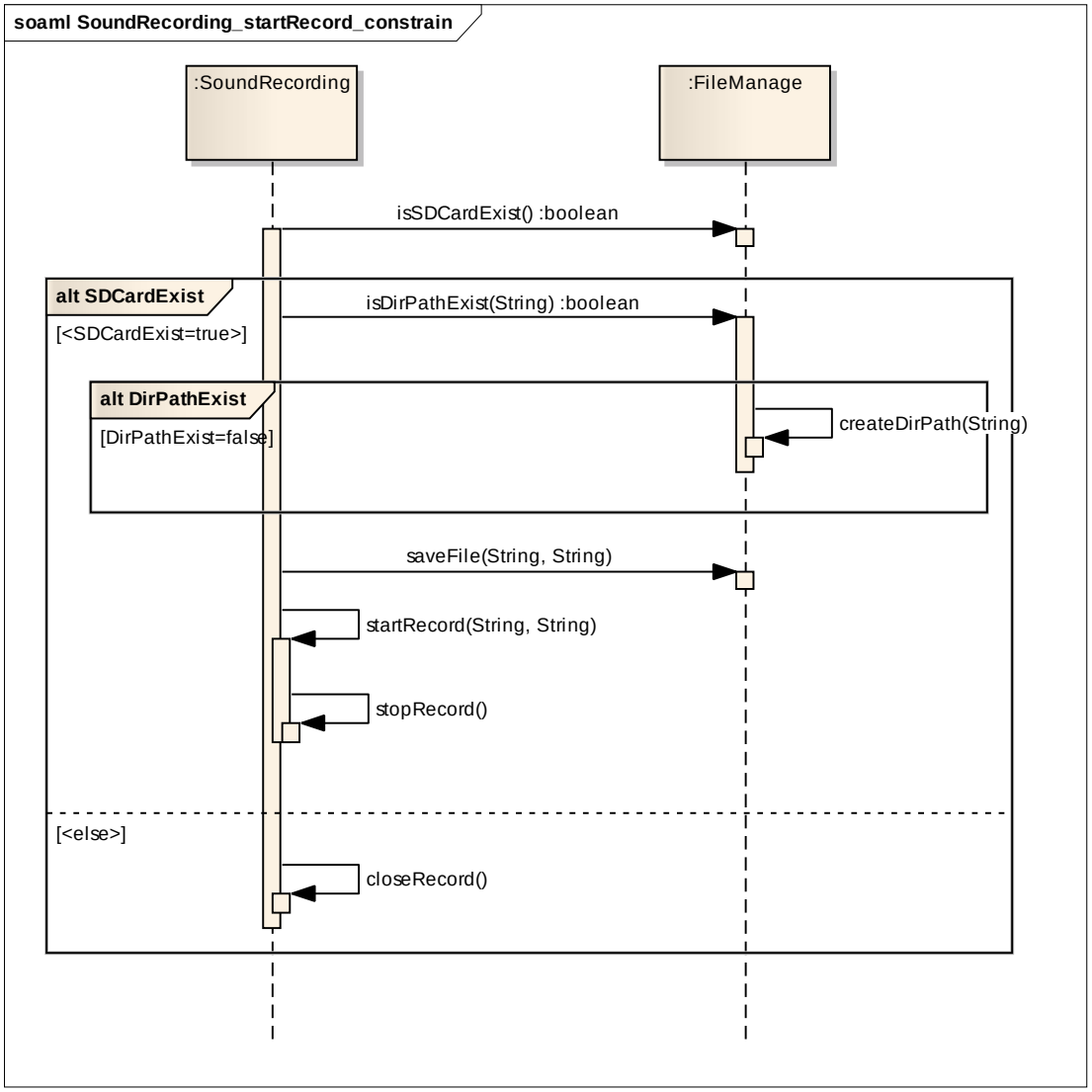

圖 4.9 SoundRecordingService 服務 startRecord () operation 與 FileManage 的協議(protocol)

在服務的規格中，每一個服務的 operation 都必須詳盡的表達。因為每一個服務裡的 operation 都表示了一個操作流程，當然有些 operation 是在協助其他 operation 的執行。例如，SoundRecording 中

我們可以多一個檔案名稱產生的 method (createFileName() operation)，來協助當引用 saveFile() operation 時，產生一個檔案再進行存檔。當這些 operation 集合成一系列的操作流程，就可以表示完成錄音功能(SoundRecording)中的錄音(startRecord() operation)服務流程。

4.3.3服務的實現

服務的實現是表示參與者(participant)將提供或消費什麼服務。由 SoaML Profile 的<<participant>> stereotype 來表示。圖 4.8 中，stereotype 為<<participant >>的 UseSoundRecording 為一個參與者，<<servicePoint>> 表示了 UseSoundRecording 參與者提供了 soundRecordingService 的服務，以及 UseSoundRecording 參與者需要 fileManage 這一個服務需求，以<<requestPoint>>表示。來協助提供完整的 soundRecordingService 服務。

服務的實現除了採用 SoaML Profile 的塑模來表達服務外,服務最終仍要以程式碼來表現。MDA 劃分了企業架構和系統架構的觀點，在系統架構中，我們採用了 SoaML Profile 來表示系統架構中的各項服務。我們知道手持式應用程式的開發必須引用 API，如：Android、J2ME、iPhone、Windows Mobile。不像網頁式平台便利，有標準的語言 HTML、XML。也因此如果我們的目的是要將此系統讓多數不同

平台的設備使用，勢必必須針對每一個設備而引用不同的 API。透過 MDA 方式的轉換可以使用 pattern 的撰寫或新增對映的關係，而達到對映原始碼的轉換。此舉減少了因設備不同而重新撰寫程式的時間，另外利用 SoaML Profile 統一不同設備的服務設計，也減少程式設計師在程式上撰寫錯誤的機會。圖 4.10 為 Java 格式的程式碼，圖 4.11 為 C# 格式的程式碼，這兩種格式的程式碼都是由 SoundRecordingService 模型所轉出來的。由程式碼與圖 4.8 中 <<serviceInterface>>為 soundRecordingService 來相互對映，程式碼中不難看出 soundRecordingService 實作 soundRecording 介面(interface)，而且 import 了 FileManage 介面(interface)，以及列出方法(Methods，operations)。更進一步的，我們也可以在分析設計階段透過協議(protocol)的訂定，利用 OCL 來加強模型上的邏輯性關係。

```
package soundRecording;
import fileManage.FileManage;

/**
 * @author Administrator
 * @version 1.0
 * @created 22-十二月-2009 下午 06:47:53
 */
public class SoundRecordingService implements SoundRecording {

	public SoundRecordingService(){

	}

	public void finalize() throws Throwable {

	}

	public void closeRecord(){

	}

	/**
	 *
	 * @param strDirPath
	 * @param strFileName
	 */
	public boolean delRecord(String strDirPath, String strFileName){
		return false;
	}

	/**
	 *
	 * @param dirPath
	 * @param fileName
	 */
	public void playRecord(String dirPath, String fileName){

	}

	/**
	 *
	 * @param strDirPath
	 * @param strFileName
	 */
	public void startRecord(String strDirPath, String strFileName){

	}

	public void stopRecord(){

	}

}
```

圖 4.10 SoundRecordingService Java 程式碼

```
///////////////////////////////////////////////////////////
//  SoundRecordingService.cs
//  Implementation of the Class SoundRecordingService
//  Generated by Enterprise Architect
//  Created on:      22-十二月-2009 下午 06:56:19
//  Original author: Administrator
///////////////////////////////////////////////////////////

using soundRecording;
namespace soundRecording {
        public class SoundRecordingService : SoundRecording {

                public SoundRecordingService(){

                }

                ~SoundRecordingService(){

                }

                public virtual void Dispose(){

                }

                public void closeRecord(){

                }

                ///
                /// <param name="strDirPath"></param>
                /// <param name="strFileName"></param>
                public bool delRecord(string strDirPath, string strFileName){

                        return false;
                }

                ///
                /// <param name="dirPath"></param>
                /// <param name="fileName"></param>
                public void playRecord(string dirPath, string fileName){

                }

                ///
                /// <param name="strDirPath"></param>
                /// <param name="strFileName"></param>
                public void startRecord(string strDirPath, string strFileName){

                }

                public void stopRecord(){

                }

        }//end SoundRecordingService

}//end namespace soundRecording
```

圖 4.11 SoundRecordingService C#程式碼

4.3.4部署服務

經由 4.3.1 服務的分析及在此範例中定義服務之間的關連性;4.3.2 定義了服務內部的規格和服務與服務之間的協議(protocol)；4.3.3 定義了服務的整體性，實現服務以及產生程式碼。在部署階段我們採用 Jboss ESB 做為 SOA 的 Middleware，並將服務部署在 ESB 之中。另外 Jboss ESB 也可以 plug-in 在 Eclipse 之中並結合 Jboss web service 使

用，其安裝方式可參考 Jboss Community [50][51][52]。

在 Eclipse 中使用 Jboss ESB 的部署是透過新增 ESB Project 中 jboss-esb.xml 來配置服務的。jboss-esb.xml 中主要可以分為 <providers> 和 <services> 兩個主要元素，<services> 元素又分為 <listeners>和<actions>兩個元素，其用法和說明可參考。圖 4.12 我們列出部份的<providers>和<services>來說明。圖中顯示了兩個服務，StartRecord 服務(33 行)和 StopRecord 服務(44 行)，各執行了<actions> 元素中 class="soundRecording.SoundRecordingService" 路徑下的 MessageStartRecord(41 行)和 MessageStopRecord(55 行)的 Process。其中 class 路徑中的 “SoundRecordingService”是我們在上一階段所產生的原始碼，而 MessageStartRecord 和 MessageStopRecord 兩個 operation，是因為我們採用 Jboss Message 來執行 ESB 的引用而手動方式加入。

另外 MessageStartRecord 和 MessageStopRecord Process 也表示了，當服務消費者使用 StartRecord 服務，則在 ESB 中會去執行 MessageStartRecord 程序，來執行錄音功能。當服務消費者使用 StopRecord 服務，則在 ESB 中會去執行 MessageStopRecord 程序來停止錄音的執行。

```xml
<?xml version="1.0"?>
<jbossesb parameterReloadSecs="5"
    xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb-1.0.1.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb-1.0.1.xsd
    <providers>
        <jms-provider connection-factory="ConnectionFactory"
            name="JBossMQ">
            <!-- 開始錄音 -->
            <jms-bus busid="soundRecording_StartRecord_GwChannel">
                <jms-message-filter dest-name="queue/soundRecording_StartRecord_Request_gw"
                    dest-type="QUEUE" />
            </jms-bus>
            <jms-bus busid="soundRecording_StartRecord_EsbChannel">
                <jms-message-filter dest-name="queue/soundRecording_StartRecord_Request_esb"
                    dest-type="QUEUE" />
            </jms-bus>
            <!--停止錄音-->
            <jms-bus busid="soundRecording_StopRecord_GwChannel">
                <jms-message-filter dest-name="queue/soundRecording_StopRecord_Request_gw"
                    dest-type="QUEUE" />
            </jms-bus>
            <jms-bus busid="soundRecording_StopRecord_EsbChannel">
                <jms-message-filter dest-name="queue/soundRecording_StopRecord_Request_esb"
                    dest-type="QUEUE" />
            </jms-bus>
        </jms-provider>
    </providers>
    <services>
        <!-- 開始錄音服務 -->
        <service category="SoundRecordingService" description="soundRecording StartRecord Service"
            name="StartRecord">
            <listeners>
                <jms-listener busidref="soundRecording_StartRecord_GwChannel"
                    is-gateway="true" name="JMS-Gateway" />
                <jms-listener busidref="soundRecording_StartRecord_EsbChannel"
                    name="SoundRecording_StartRecord" />
            </listeners>
            <actions mep="OneWay">
                <action class="soundRecording.SoundRecordingService" name="action1"
                    process="MessageStartRecord" />
            </actions>
        </service>
        <!-- 停止錄音服務 -->
        <service category="SoundRecordingService" description="soundRecording StopRecord Service"
            name="StopRecord">
            <listeners>
                <jms-listener busidref="soundRecording_StopRecord_GwChannel"
                    is-gateway="true" name="JMS-Gateway" />
                <jms-listener busidref="soundRecording_StopRecord_EsbChannel"
                    name="SoundRecording_StopRecord" />
            </listeners>
            <actions mep="OneWay">
                <action class="soundRecording.SoundRecordingService" name="action1"
                    process="MessageStopRecord" />
            </actions>
        </service>
    </services>
</jbossesb>
```

圖 4.12 部署 ESB 的 jboss-esb.xml 檔案

在 4.3 進階 PIM 分析當中，我們使用了 SoaML profile 來分析服務、執行流程的訂定、利用 MDA 的方式，依特定的平台轉換及產生原始碼，最後部署服務到 Jboss ESB 之中。每一個服務的執行，可能會依賴某個服務來完成它所提供的服務給消費者，因此一個服務的修正，可能會影響到其他服務的正常執行，而造成系統的失敗，也因此我們建議，當分析設計完服務之後，務必執行服務的測試，及相關文

件的產生，以隨時的追蹤。

4.4PIM 轉換PSMs

“platform specific model, PSM，探討系統最後所要部署的平台。”

經由進階 PIM 分析階段，我們使用 SoaML Profile 來分析設計服務及部署之後，我們原本預期產生圖 4.5 class diagram 的架構，改成了圖 4.13 的 class diagram。stereotype 為 <<participant>> 的 UsePhotoManage、UseCameraService、UseGPSLocationService、UseSoundRecording 四個 Class，是由 SOA 中的 ESB 所提供使用。原本圖 4.5 中的 FileManage Class 以經變成一個服務部署在 ESB 中，並且與 SoundRecording、PhotoManage 組合成 SoundRecordingService、PhotoManageService 兩個新的服務。也就是說當服務消費者使用 SoundRecordingService 這一個服務的 startRecord() 的功能時，依照 SoundRecording 與 FileManage 之間的協議(protocol)，startRecord()會去要求使用 FileManage 相關的操作，而我們跟本不會直接使用到 FileManage 這一個服務。

另外圖中四個 <<participant>> 都有 esbSetupConnection()、esbStopConnection()兩個 operation，做為連結到 ESB 取得服務及結束 ESB 連結之用，其餘的 operation 為服務所提供的操作。

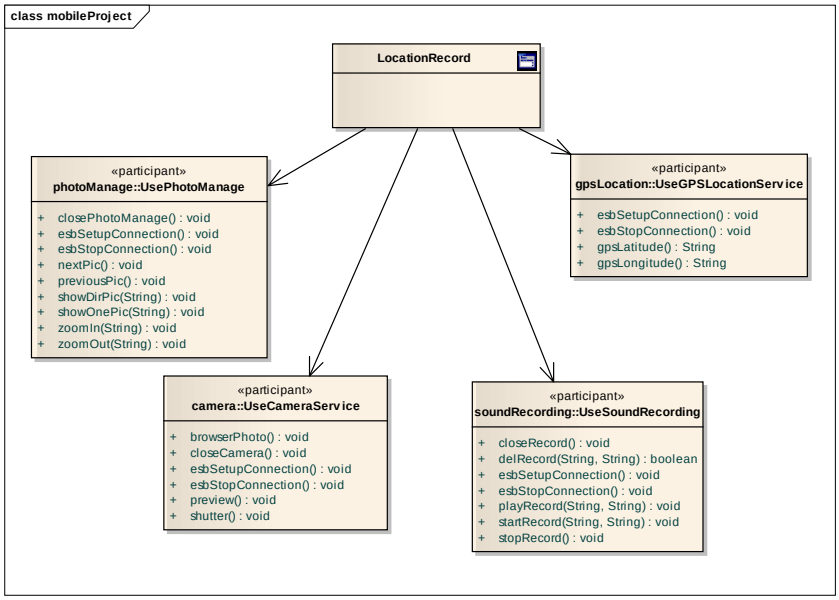

圖 4.13 系統與服務之間的關係

在 PSM 階段中，如同在 4.3.3 服務的實現，選擇了最後所要部署的 Java 或 C# 的平台。另外我們也可以透過 EA 功能中的 Transformation Templates 來設計新的對映平台。以目前 EA 所提供的轉換平台共有九種，除了 C#、Java EJB、WSDL 外也包含資料庫格式的 DDL 格式。我們這裡採用了 Java 的平台，當然也可以選擇 EJB 的架構當作部署的平台。

4.5PSM轉換程式碼

手持式設備的應用程式開發，除了產生可以執的程式碼外，另一個使用者介面設計(UI design)，也會隨著不同的設備而有不同的 UI 設計介面，不像網頁有共同的 HTML 標準語言能執行在瀏覽器當中。也因此在各種不同手持設備中，所要開發的應用程式 UI，仍必須各

自的獨立設計操作畫面。但一般而言，應用程式和網頁的 UI 設計，都是由某個元素(Element)來驅使事件(Event)的發生，雖然在不同設備上，UI 的設計無法相互的轉換，然而 UI 背後所對應的各種事件卻大至相同，像 Button 的 onClick 事件、Text 的 onForce 事件…等。然而，各種事件的背後所要進行的動作是可以對應到，我們所轉換出來的 Methods 或 Function 上。所以在這一階段，我們分成兩個面向來執行，一為手持設備的 UI 設計，另一個為 PSM 轉換程式碼。圖 4.14 表示了我們將 UI 的設計，以及我們在上一階段選擇轉換平台之後，再轉換成原始碼的產生。轉出來的 Java 程式碼，除了和 UI 相對映的事件驅動的方法(methods)外，也包含了連接 ESB 以取得服務的 methods，來提供事件的使用。

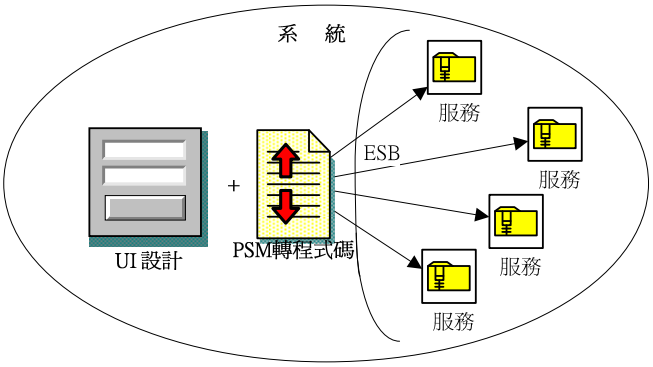

圖 4.14 UI 設計、PSM 轉換程式碼相結合示意圖

圖 4.15 是擷取部份 LocationRecord 轉出的原始碼，圖中的 import 了 UseCameraService、UseGPSLocationService、UsePhotoManage、UseSoundRecording 四隻原始碼。圖 4.16 是我們利用 Android 模擬器

執行出來的 UI 設計。

```
package mobileProject;
import camera.UseCameraService;
import gpsLocation.UseGPSLocationService;
import photoManage.UsePhotoManage;
import soundRecording.UseSoundRecording;

/**
 * @author Administrator
 * @version 1.0
 * @created 23-十二月-2009 下午 06:05:00
 */
public class LocationRecord {

    private UseSoundRecording m_UseSoundRecording;
    private UsePhotoManage m_UsePhotoManage;
    private UseCameraService m_UseCameraService;
    private UseGPSLocationService m_UseGPSLocationService;

    public LocationRecord(){

    }

    public void finalize() throws Throwable {

    }

    public UseCameraService getUseCameraService(){
        return m_UseCameraService;
    }
```

圖 4.15 LocationRecord 部份程式碼

圖 4.16 Android 模擬器執行 UI 設計

在產出 UI 設計以及原始碼的產生後，接下來的動作就是做 UI 與程式碼的接合，針對 UI 中的 Element 來產生 event 的驅動，而接合到 LocationRecord 程式碼中的 methods，這些的動作都需要手動的方式來做。然而，事件驅動後引發的一連串服務的執行，是我們在“進階

PIM 分析”階段中所設計服務的執行流程，例如，當我們按下圖中錄音的按鈕，會引發錄音的事件發生，而呼叫服務的執行來完成錄音的服務。當按下停止按鈕，引發停止的服務來執行停止的動作。

4.6部署系統

最後階段將完成的專案，再部署到外部的 ESB 上供 end-user 下載使用。到此整個開發的流程也告一段落。

這裡我們沒有強調測試的方式，但以 Eclipse 做為開發的工具，可利用 JUnit 來產生使用者案例，針對服務的測試和者系統的測試都能提昇系統的穩定度。文件的產生，EA 和 Eclipse 也都有提供這方面的文件，都可以好好的善加利用。

第5章 MDA開發流程比較

第三章中，我們提出了 MDA 與 SOA 的開發流程方法論，在第四章中，我們實作了開發流程。在這一章節，我們進一步的嘗試將我們提出的方法論與 RUP 開發流程來做比較。

MDA 規格書中所提出的三個觀點，是引導開發人員在開發軟體中，應將系統分成三個觀點來進行分析、設計以及透過轉換來達到軟體的開發。當我們將 MDA 的觀念，納入整個系統的開發流程當中，我們也試著去定義一個開發模式。傳統上開發模式，如：瀑布模式(waterfall model)、RUP。每一個都有開發的程序，來引領軟體開發人員完成系統的開發。也因此我們參考 RUP[36]中，九個工作流程中的六個核心工作流，來做為我們所提出的開發模型的對映和比較。

圖 5.1 是我們提出的 MDA 開發模型，類似於 RUP 的四個階段、九個工作流。與 RUP 的程序定義一樣，我們也認同軟體的開發是一種反覆式的開發，且都著重在模型上的設計。但是 MDA 更進一步，提供了系統開發另一種思維方式，也就是“轉換和重覆使用架構”。不管在 PIM 轉換 PSM 或 PSM 轉到程式碼，都必須明定轉換的規則。也就是說必須針對來源的模型語言，是如何轉至目標模型、平台、程式語言來訂定轉換的規則，另外模型也可以透過 XMI 的格式做為轉

換的依據。所以我們在階段中增加了“模型轉換階段”，且大幅改變流程。但不可否認，開發一個系統使之完善，是需要精心的分析和設計，也因此我們在流程中所增加的 CIM、PIM、PSM，仍對映於 RUP 的需求分析和設計的流程當中。底下是我們提出的五個階段、六個流程。

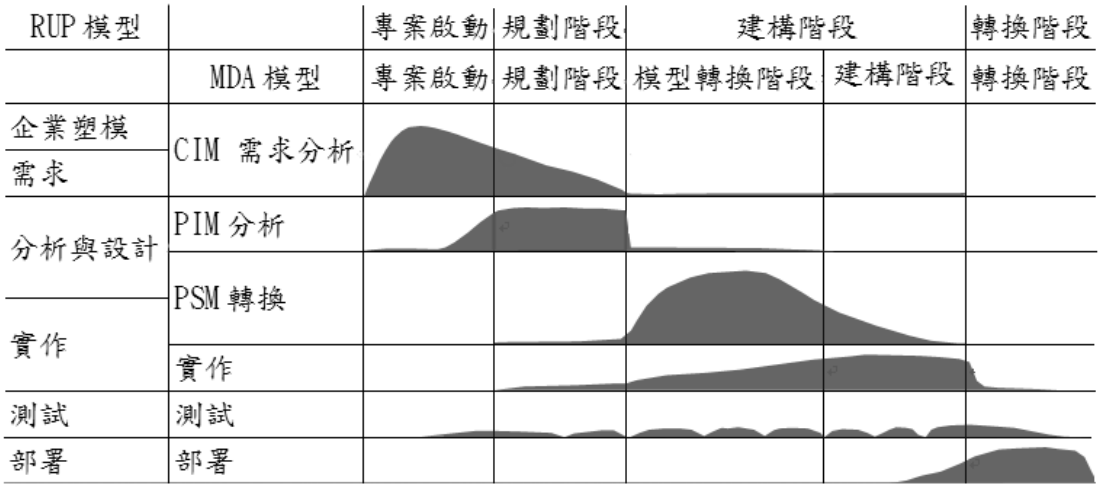

圖 5.1 MDA 開發模型

✓ 五個階段：

1. 專案啟動：同 RUP 的專案啟動。找出企業領域的需求、工作目標、時程評估、風險評估。利用使用者案例來描述系統架構的輪廓、驗證系統。
2. 規劃階段：同 RUP 的規劃階段。分析企業領域的需求定義，針對問題進行深度分析，並設計系統架構與物件的模型。
3. 模型轉換階段：對映 RUP 建構階段。透過系統的雛型來進一步了解使用者需求是否符合。並且決定企業使用平台，描述、

撰寫轉換平台的規則。最後測試轉換的規則是否符合目標平台。

4. 建構階段：對映 RUP 建構階段。正式的將模型轉換，以及將轉換的程式碼做調整，並進行後續的功能測試和驗證。
5. 轉換階段：同 RUP 的轉換階段。做系統轉移的動作，進行系統的調整、整合測試、文件交付。

✓ 六個核心流程：

1. CIM 需求分析：與 RUP 中企業塑模、需求兩流程相同。同樣著重於企業領域的分析，以釐清企業流程是如何由待開發的系統來支援取代。CIM 後期可借由 use case diagram 及 active diagram 來呈現系統的流程及牽涉的系統、人員等。
2. PIM 分析：對映 RUP 的分析與設計流程。利用各種不同的塑模的方式或利用特定的 UML Profile，來建構系統的不同觀點。並且經過進一步的細部分析及利用 OCL、Tagged values 來標示模型，讓模型更貼進最終的系統。後期可採用 class diagram、state diagram 等，以及利用 profile、stereotype、OCL 及 Tagged Values 的設定，來表示系統。
3. PSM 轉換：介於 RUP 的分析與設計流程和實作流程。依照模型編寫轉換的語言，以支援特定的 pattern、架構以及最後部署

平台的轉換。所產生的 templates、patterns 都可以重覆的使用，並且可以提供架構師選擇。

4. 實作：對映 RUP 的實作流程。依照所轉換出來的程式碼接續系統的開發，以補全系統的完整性。另外實作流程，也著重於工程師的單元的測試和測試案例的產出。
5. 測試：依照產出的測試案例加以進行專業的測試、整合測試、系統測試，並且衡量系統測試的標準。
6. 部署：可交付的系統、相關文件，以及系統配置的相關設定。在企業內部可以列出相關的 templates、patterns 以供下一專案參考使用。

第6章 研究結論

在本篇論文中，我們提出了“SOA 應用 MDA 的開發流程”方法論和“MDA 開發流程比較”，並且我們也在一個系統專案裡，應用了此開發流程方法論。在整個方法論與實際運作在手持設備上開發，我們仍遭遇到些許的問題和不同的想法。我們將在這一章節提出討論並提出看法，並提出未來工作與方向。

6.1研究討論

SoaML Profile 對於 SOA 架構而言，的確是一個非常好的塑模工具，它能很清楚的描述服務是如何組成及服務間的資訊流。並且在系統分析採用 MDA 方式將系統分離成 CIM、PIM、PSM 三個階段，在每一個階段中清楚的描述系統，再利用 SoaML Profile 串接整個企業架構所需要的服務，如此，每一個服務都可以有脈絡可尋，並且增加了服務的重覆使用性。但當我們將此開發流程運用在手持設備專案上執行，在系統和 SOA 提供服務的分析和設計，我們利用了 UML 2.0 和 SoaML Profile 工具來進行。理想上，一個服務的設計，應提供不同的設備引用。但是，我們真正將模型轉換到目標程式之後，仍必須針對不同的設備引用不同的 API，才能符合特定設備的使用，這樣的服務方式，感覺上已經失去了 SOA 的意義，而且這樣的做法，必須

對服務要有妥善的分類管理。可以想像的，如果在企業上開發手持設備的應用，我們並不能限制員工的設備，也因此，網頁的方式似乎是可以解決這樣的困難。就像我們另一個功能 MapBrowser，採用網頁來開發，以解決不同設備的問題。

另外，以 MDA 來說，轉換語言的撰寫是非常重要的。以我們案例為例，我們在服務的部署上，單純的只是轉換成 Java 的平台，但在有組織規劃的企業上，可能會再將服務轉成 Enterprise JavaBeans, EJB 的架構，從各項的服務再擷取出相同的邏輯層，換句話說，在不同手持設備的環境下，程式的邏輯可以再細分服務出來。例如，錄音功能中的檔名產生。檔名產生在不同設備都有這樣的需求，也因此，可以將檔名產生當作另一項服務，且用 EJB 的方式來呈現。

整個“SOA 應用 MDA 的開發流程”，與傳統的軟體開發流程相比，實值上並沒有太大的不同，唯一不同的是 MDA 的觀念。整個流程來說，都必須經過，系統分析、設計、系統實作等階段。但是 MDA 的三個觀點，CIM、PIM、PSM，提供給我們的觀念，是對系統的另一種思考方式，也就是“轉換和重覆使用架構”。轉換是針對模型轉換到相對應的平台，來減少程式師重新開發程式的時間；重覆使用架構是針對一個完整的架構重覆使用，而不用重新的打造整個系統架構。

6.2未來工作

SOA 的架構是一個很好的方式，不僅適用在 Distributed 的架構中也強調 Loosely coupled 的開發方式，以此增加了 Components 的重覆使用性。另一份 OMG 所提供 Super Distributed Object, SDO[46]文件，是運用 MDA 的開發方式應用在數位家庭中的硬體上，並且利用 Super Distributed Objects, SDO 來表示硬體或軟體設備的元件，並讓其他 SDO 可以來存取使用，而每一個 SDO 都有一組 Resource Data Model 來描述硬體的資訊如，Device profile、Service profile、Configuration profile、Organization、Status。當有其他 SDO 想要取用時，可依它所需來呼叫並使用。例如，計時器就是一個很好的例子，一個計時器可被結合在洗衣機或烤箱內來呼叫使用。也因此我們認為這樣的模式類似 SOA 的架構，可運用在數位家庭之中。家庭成員可以結合他所需的服務，例如，即時的醫療服務、GPS 定位預防人員走失、家庭電器設備管理等，都可以整合在手持式設備上使用。

在我們提出的開發流程方式中，SoaML Profile 應用在 SOA 上，分析服務的功能、整體架構，是一個很好的 Profile 分析、設計工具。它透過了 UML 來表示服務，但服務最終是部署在 ESB 中，ESB 變成服務部署重要的集散地，也因此 ESB 對服務的維護管理、異動追蹤、服務路徑組合也相對的重要，這也都我們未來所必須努力的方向。

第7章 參考文獻

[1] 楊展岳(民98年9月21日)。“2009年第二季我國行動上網觀測。”資策會FIND。民98年12月20日，取自：http://www.find.org.tw/find/home.aspx?page=many&id=240

[2] Elmer-DeWitt, philip (MAR 4 2009), “iSuppli:Smartphone sales could grow 11% in 2009.CNN,” Retrieved Dec 20 2009, from http://tech.fortune.cnn.com/2009/03/04/isuppli-smartphone-sales-could-grow-11-in-2009/

[3] Samsung Application Store, from http://www.samsungapps.com/

[4] Android Market, from http://www.android.com/market/

[5] Open Handset Alliance (2007), *“Industry Leaders Announce Open Platform for Mobile Devices,”* Open Handset Alliance. Retrieved Dec 20 2009, from thttp://www.openhandsetalliance.com/press_110507.html

[6] BlackBerry, *“Java Application Development Overview,”* BlackBerry. Retrieved FEB 3 2010, from http://na.blackberry.com/eng/developers/javaappdev/

[7] iPhone, *“iPhone DevelopmentGuide,”* iPhone .Last updated MAY 28 2010, Retrieved JUN 29 2010, from http://developer.apple.com/iphone/library/documentation/Xcode/Conceptual/iphone_development/000-Introduction/introduction.html

[8] Nicolaou, A. (APR 7 2009). *“HTML5 and WebKit pave the way for mobile web applications,”* Google Code Blog. Retrieved DEC 20 2009, from http://googlecode.blogspot.com/2009/04/html5-and-webkit-pave-way-for-mobile.html

[9] Stevens, H. and Pettey, C. (MAR 11 2009). *“Gartner Says Worldwide Smartphone Sales Reached Its Lowest Growth Rate With 3.7 Per Cent Increase in Fourth Quarter of 2008,”* Gartner

[10] Hickson, I. (OCT 12, 2009), *"HTML5 Draft Standard - 12 Oct 2009,"* WHATWG community. Retrieved DEC 22 2009, from http://www.whatwg.org/specs/web-apps/current-work/

[11] Qusay H. Mahmoud (APR 2005), *"Programming the BlackBerry With J2ME,"* ORACLE Sun Developer Net work. Retrieved NOV 22 2009, from http://developers.sun.com/mobility/midp/articles/blackberrydev/

[12] Anonymous. *"J2ME Android Bridge,"* Assembla. Last updated MAR 8 2009, Retrieved NOV 22 2009, from http://www.assembla.com/wiki/show/j2ab/Converting_From_J2ME

[13] Krill, P. (MAR 7 2008), *"Sun: We'll put Java on the iPhone,"* infoworld. Retrieved NOV 22 2009, from http://www.infoworld.com/d/developer-world/sun-well-put-java-iphone-042

[14] Anonymous, *"Web specifications supported in Opera 9.5[HTML and XHTML support],"* Opera. Retrieved JAN 10 2010, from http://www.opera.com/docs/specs/opera95/

[15] Anonymous, *"Packager for iPhone,"* Adobe Labs. Retrieved MAY 8 2010, from http://labs.adobe.com/technologies/packagerforiphone/

[16] Xiao, Y., Tao, Y., and Li, Q. (2009), *"A New Mobile Web Presentation with Better User Experience,"* 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing, vol. 1, pp.137-141.

[17] Reynolds, F. (2009), *"Web 2.0 In Your Hand,"* IEEE Pervasive Computing, 2009, vol. 8, no. 1, pp. 86-88.

[18] Vaughan-Nichols, Steven J. (2008). *"The Mobile Web Comes of Age,"* IEEE Computer, 2008, vol. 41, no 11, pp. 15-17.

[19] Miller, J. and Mukerji, J.(2003), *"MDA Guide Version 1.0.1,"* Object Management Group[OMG], Retrieved JUL 13 2009, from http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

[20] McLean, P. (AUG 21 2009), *"Canalys: iPhone outsold all Windows*

Mobile phones in Q2 2009," AppleInsider. Retrieved JUL 10 2009, from http://www.appleinsider.com/articles/09/08/21/canalys_iphone_outsold_all_windows_mobile_phones_in_q2_2009.html

[21] Anonymous, *"Mobile application development,"* Wikipedia. Last updated JAN 3 2009, Retrieved JUL 29 2009, from http://en.wikipedia.org/wiki/Mobile_development

[22] Anonymous, *"Mobile browser,"* Wikipedia. Last updated MAR 10 2009, Retrieved JUL 29 2009, from http://en.wikipedia.org/wiki/Microbrowser

[23] Hathaway, J. (OCT 20 2009), *"Mozilla makes its mobile move, brings Firefox 3.6 to Android,"* downloadsquad. Retrieved JUL 29 2009, from http://www.downloadsquad.com/2009/10/20/mozilla-makes-its-mobile-move-brings-firefox-3-6-to-android/

[24] Endrei, M., Ang J., Arsanjani J., Chua, S., Comte, P., Krogdahl, P., et al. (2004), *"Patterns: Service-Oriented Architecture and Web Services,"* IBM Redbooks. Last updated JUL 29 2004, Retrieved MAY 4 2009, from http://www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf

[25] Deng, W., Yang, X., Zhao ,H., Lei D., and Li, H. (2008), *"Study on EAI Based on Web Services and SOA,"* 2008 International Symposium on Electronic Commerce and Security, 2008, pp. 95-98.

[26] Zhang, X. and Yin, X. (2009), *"Researh of B2B E-business application & development technology based on Web services,"* 2009 2nd IEEE International Conference on Computer Science and Information Technology, Beijing, China, 2009, pp. 586-589.

[27] Bajwa, I.S., Samad A., Mumtaz S., Kazmi R., and Choudhary, A. (2009), *"BPM Meeting with SOA: A Customized Solution for Small Business Enterpris*es," 2009 International Conference on Information Management and Engineering, Kuala Lumpur, Malaysia, 2009, pp. 677-682.

[28] Keen, M., Bishop S., Hopkins, A., Milinski, S., Nott, C., Robinson, R., et al. (2004), *"Patterns: Implementing an SOA using an*

Enterprise Service Bus," IBM Redbooks. Last updated JUL 25 2004, Retrieved JUN 4 2009, from http://www.redbooks.ibm.com/redbooks/pdfs/sg246346.pdf

[29] Lin, K.J., Panahi, M., Zhang, Y., Zhang, J., Chang, S. (2009), *"Building Accountability Middleware to Support Dependable SOA,"* Internet Computing, IEEE, 2009, vol. 13, no. 2, pp. 16-25.

[30] Anonymous, *"HTML 5 Tag Reference,"* W3school. Retrieved DEC 22 2009, from http://www.w3schools.com/html5/html5_reference.asp

[31] Anonymous, *"Android 2.0 Platform Highlights,"* Android developers. Retrieved DEC 22 2009, from http://developer.android.com/sdk/android-2.0-highlights.html.

[32] Gilbertson, S. (FEB 19 2009), *"Google Turns to HTML 5 to Solve Offline Mobile Woes. Feb.2009,"* Retrieved DEC 22 2009, from http://www.wired.com/epicenter/2009/02/google-turns-to

[33] Berre, A.J. (2009), *"Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS),"* SoaML WIKI. Last updated APR 9 2009. Retrieved AUG 20 2009, from http://www.omgwiki.org/SoaML/doku.php?id=specification

[34] Object Management Group (2009), *"OMG Model Driven Architecture,"* OMG. Last updated JUN 2 2010, Retrieved SEP 10 2009, from http://www.omg.org/mda/index.htm

[35] Mellor, S.J., Watson, A., *"Roles in the MDA Process,"* OMG. Retrieved DEC 10 2009 from http://www.omg.org/registration/registration-roles_mda.htm

[36] IBM, *"IBM Rational Unified Process,"* IBM. Retrieved JAN 10 2010 from ftp://ftp.software.ibm.com/software/rational/web/datasheets/RUP_DS.pdf

[37] Riba, N., Cervantes, H. (2007), *"A MDA tool for the development of service-oriented component-based applications,"* Current Trends in Computer Science. ENC 2007. Eighth Mexican International Conference on, 2007, pp. 149-156.

[38] Choi, Y., Yang J.S., and Jeong, J. (2009), *"Application framework for multi platform mobile application software development,"* Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on, 2009, vol.1, pp. 208-213.

[39] Natchetoi, Y., Kaufman, V., and Shapiro, A. (2008), *"Service-oriented architecture for mobile applications,"* Proceedings of the 1st international workshop on Software architectures and mobility, Leipzig, Germany: ACM, 2008, pp. 27-32.

[40] Cao, Y., Jarke, M., Klamma, R., Mendoza, O., and Srirama, S., *"Mobile Access to MPEG-7 Based Multimedia Services,"* In Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, 2009, pp. 102-111.

[41] Kesteren, A.V. (2009), *"HTML5 differences from HTML4,"* W3CWorking. Last updated JUN 24 2010, Retrieved AUG 20 2009, from http://dev.w3.org/html5/html4-differences/

[42] Anonymous, *"Comparison of layout engines (HTML5)"* Wikipedia. Last updated JUL 8 2010, Retrieved AUG 21 2009, from http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(HTML_5)

[43] Papakipos, M. (MAY 28-28 2009), *"Google's HTML 5 Work: What's Next?,"* Google. Retrieved NOV 1 2009, from http://code.google.com/intl/zh-TW/events/io/2009/sessions/GoogleHTML5Work.html

[44] Marzullo, F.P. de Souza, J.M. Blaschek, J.R. (2008), *"A Domain-Driven Development Approach for Enterprise Applications, Using MDA, SOA and Web Services,"* 2008 10th IEEE Conference on E-Commerce Technology and the Fifth Enterprise Computing, E-Commerce and E-Services, pp. 432-437.

[45] Enterprise Architect, Retrieved NOV 1 2009, from http://www.sparxsystems.com.au/

[46] Object Management Group (OCT 2008), *"Platform Independent Model and Platform Specific Model for Super Distributed Object,"*

OMG. Retrieved JUL 1 2009, from http://www.omg.org/spec/SDO/1.1/PDF

[47] Meservy, T.O. and Fenstermacher, K.D. (2005), *"Transforming software development: an MDA road map,"* IEEE Computer, 2005, vol. 38, no 9, pp. 52-58.

[48] JBoss ESB, Retrieved DEC 2 2009, from http://www.jboss.org/jbossesb/

[49] Anonymous, *"Top-down and bottom-up design,"* Wikipedia. Last updated JUN 26 2010, Retrieved SEP 10 2009, from http://en.wikipedia.org/wiki/Top-down_and_bottom-up_design

[50] JBoss Community (2009), *"Getting Started Guide,"* JBoss. Retrieved DEC 10 2009, from http://www.jboss.org/jbossesb/docs/4.7/manuals/pdf/GettingStarted.pdf

[51] Andersen, M. (2008), *"Installing JBoss Tools,"* Jboss Community. Last updated JAN 20 2010, Retrieved DEC 10 2009, from http://community.jboss.org/wiki/InstallingJBossTools

[52] JBoss Community (2009), *"Programmers Guide,"* JBoss. Retrieved DEC 10 2009, from http://www.jboss.org/jbossesb/docs/4.7/manuals/pdf/ProgrammersGuide.pdf