行政院及所屬各機關出國報告

（出國類別：研究）

端末作業未來之發展及應用

服務機關：臺灣土地銀行資訊室
出國人　職　　稱：領　　組
　　姓　　名：張 賜 安
出國地區：美　國
出國期間：91年12月7日至91年12月21日
報告日期：92年3月15日

系統識別號:C09105886

公 務 出 國 報 告 提 要

頁數: 49　含附件: 否

報告名稱:

端末作業未來之發展及應用

主辦機關:

臺灣土地銀行

聯絡人／電話:

陳元雙／02-23483170

出國人員:

張賜安　臺灣土地銀行　資訊室　領組

出國類別: 研究

出國地區: 美國

出國期間: 民國 91 年 12 月 07 日 -民國 91 年 12 月 21 日

報告日期: 民國 92 年 03 月 15 日

分類號/目: D3／銀行　D3／銀行

關鍵詞:　端末作業未來之發展及應用

內容摘要: 本次奉派赴美國研習「端末作業未來之發展及應用」，出國參加研究及拜訪期間，主要目的在於訪察瞭解國外電腦公司、外商銀行對分行端末系統作業未來之發展及應用方向，了解分行端末系統趨勢及重要技術並蒐集相關資料，藉由參訪美國UNISYS電腦公司、美國ALLTEL電腦公司、美國商業銀行（Bank Of America），再參酌現行美國業界在分行端末系統之發展與應用經驗，配合新資訊科技的發展，並考量本行業務之特性，以作爲本行未來對分行端末系統發展與應用之參考，祈能提供本行未來端末作業規劃之參酌，有助於本行端末作業系統之整合與擴充。

本文電子檔已上傳至出國報告資訊網

摘　要

本次奉派赴美國研習「端末作業未來之發展及應用」，出國參加研究及拜訪期間，主要目的在於訪察瞭解國外電腦公司、外商銀行對分行端末系統作業未來之發展及應用方向，了解分行端末系統趨勢及重要技術並蒐集相關資料，藉由參訪美國UNISYS電腦公司、美國ALLTEL電腦公司、美國商業銀行（Bank Of America），再參酌現行美國業界在分行端末系統之發展與應用經驗，配合新資訊科技的發展，並考量本行業務之特性，以作為本行未來對分行端末系統發展與應用之參考，祈能提供本行未來端末作業規劃之參酌，有助於本行端末作業系統之整合與擴充。

目　　次

圖 目 錄

壹、任務範圍

本次奉派赴美國參加研究及拜訪期間，主要目的在於訪察瞭解國外電腦公司、外商銀行對分行端末系統作業未來之發展及應用方向，了解分行端末系統趨勢及重要技術並蒐集相關資料，藉由參訪美國UNISYS電腦公司、美國ALLTEL電腦公司、美國銀行（Bank Of America），再參酌現行美國業界在分行端末系統之發展與應用經驗，配合新資訊科技的發展，並考量本行業務之特性，以作為本行未來對分行端末系統發展與應用之參考，祈能提供本行未來端末作業規劃之參酌，有助於本行端末作業系統之整合與擴充。本次考察期間自民國九十一年十二月七日至九十一年十二月二十七日（內含九十一年十二月二十一日至九十一年十二月二十七日自費休假七天）。

貳、研習內容

為了解美國一般銀行最新端末系統發展與應用現況，以作為本行未來銀行端末系統發展技術之參考，本次奉派赴美國優利電腦公司、美國ALLTEL電腦公司總部、及美國商業銀行（Bank Of America）小岩城分行訪察，此行目的在瞭解目前歐美金融服務電腦化情形，茲針對其銀行端末系統發展與應用現況訪察內容說明如下：

一、參訪美國優利（UNISYS）電腦公司

本次參訪首站來到位於賓州費城美國優利（UNISYS）電腦公司的Tredyffrin Executive Briefing Center，本行目前使用的中心正式作業主機NX4802與裝置於彰化的備援主機NX5822係為優利股份有限公司所生產，並負責系統維護事宜。

UNISYS公司是全球著名的資訊管理公司，創始於一九八六年，當時美國寶來總公司（Burroughs）宣佈與史伯利（Sperry）公司合併成立一新公司，並將之命名為UNISYS Corporation，隨之台灣寶來股份有限公司亦更名為台灣優利系統股份有限公司。UNISYS自從合併之後，在一九八七年曾躍升為全球僅次於IBM的第二大資訊服務公司，以提供高品質的資訊管理服務為其宗旨。UNISYS公司的服務範圍非常廣泛，計有5萬家客戶遍及世界100多個國家，其中包括政府機構、國防管理、金融機構、製造及服務業、交通運輸、教育機構等等。

UNISYS公司共分成四個主要部門：Global Industry、Global Network Services、System & Technology及Global Otusourcing。UNISYS公司除了生產主機、個人電腦、票據分類機等硬體產品之外，也提供業務分析規劃、應用軟體開發、委外服務、系統整合、網路整合、多品牌產品維修服務及教育訓練等服務。UNISYS至今已發展成為一家橫跨美、歐、亞、非四大洲的跨國企業，員工總數約36,000人。台灣優利系統股份有限公司（UNISYS Taiwan Ltd.）為UNISYS台灣分公司，且在台灣提供資訊服務已有近三十年的歷史。台灣優利系統股份有限公司的資本額於一九九〇年時增為新台幣1億2,400萬元，員工亦擴編為300人。其主要客戶包括：台灣銀行、土地銀行、交通銀行、華僑銀行、萬通銀行、華信銀行、泛亞銀行、遠東銀行、福特六和汽車……等。

UNISYS公司累積過去二十多年在台灣建置銀行自動化系統的經驗，運用最新的資訊科技，歷經多年十餘位工程人員所研究發展完成最新一代開放式的分行系統，以目前的資訊科技現況，在開放的架構之下使用單一作業平台，處理全功能的各類業務自動化（不同中心主機之櫃台連線交易、會計、印鑑、及分行MIS、遠程報表列印……等環境），並且兼顧各層級人員不同的使用者介面及應用需求、電子郵件、公文管理收發、理財服務、證券與期貨即時資訊、匯率看板、Intranet、Internet、Call Center、電子商務、電子銀行、網路銀行等。

二、參訪美國ALLTEL電腦公司總部

美國ALLTEL電腦公司其總部位於美國前任總統柯林頓的故鄉阿肯色州（Arkansas）的一個小城市－小岩城（Little Rock），主要核心事業為通訊服務（超過1千萬客戶）及金融服務（超過876名客戶），年收入約76億美元，員工人數超過24,000人，市值超過190億美元，美國財富雜誌（Fortune）前500大的公司，ALLTEL過去五年花費超過565萬美元經費在方案研究上。有關ALLTEL電腦公司的詳細歷史背景可參考圖2-1。

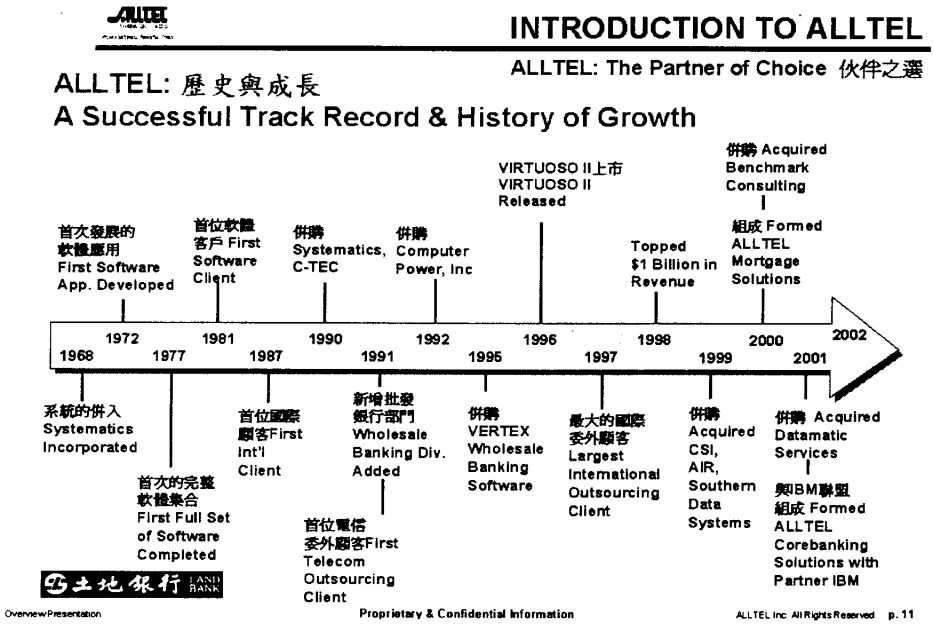

圖2-1、ALLTEL電腦公司的歷史與成長

ALLTEL電腦公司所提供的金融服務如下：

- 資料庫中心合併（Data Center Consolidation）
- 商務延續服務（Business Continuity Services）
- 應用軟體委外服務（Application Outsourcing）
- 商務流程整合（Business Process Integration）
- 應用軟體管理（Applications Management）
- 電話服務中心管理（Call Center Management）
- 帳單管理/付款解決方案（Bill Production, Presentment and Payment Solutions）
- 系統整合（Systems Integration）
- 委外服務（Outsourcing）
- 合併/收購之系統整合（Merger/Acquisition Human Resources Assimilation）
- 科技委外服務（Total Information Technology Outsourcing）
- 電腦系統操作（Computing Systems and Operations Client/Server, Web Hosting, Net-Centric, Mainframe）

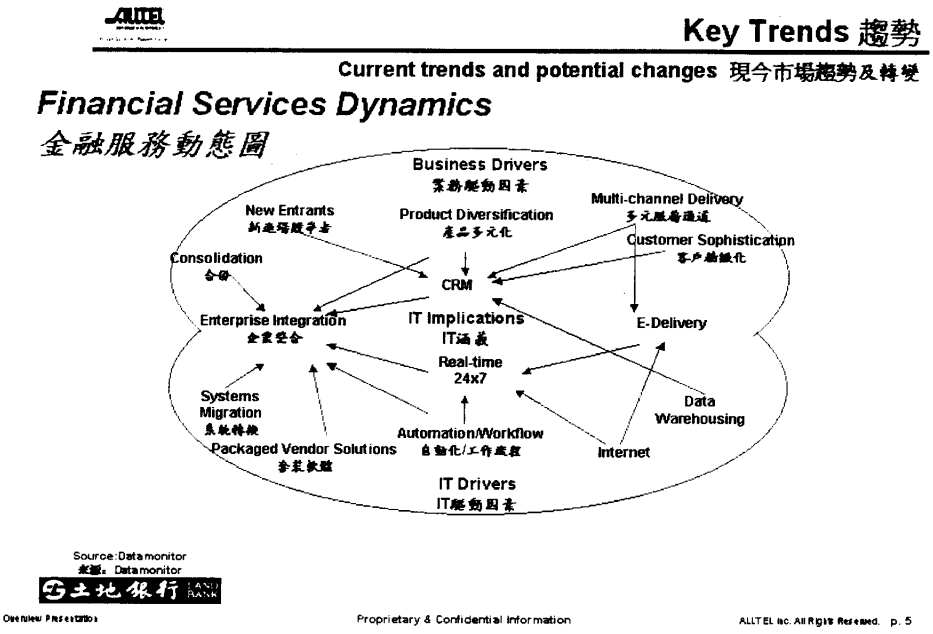

圖2-2、ALLTEL電腦公司的金融服務整合方案

三、參觀美國商業銀行（Bank of America）小岩城分行

此次參訪非常感謝美國ALLTEL電腦公司特別安排參觀美國商業銀行位於阿肯色州的小岩城分行，雖然地屬美國中部的小城市，但是它的營業廳卻呈現出與紐約這種大城市擁擠的狹小空間，有著截然不同的感覺（圖2-3）。

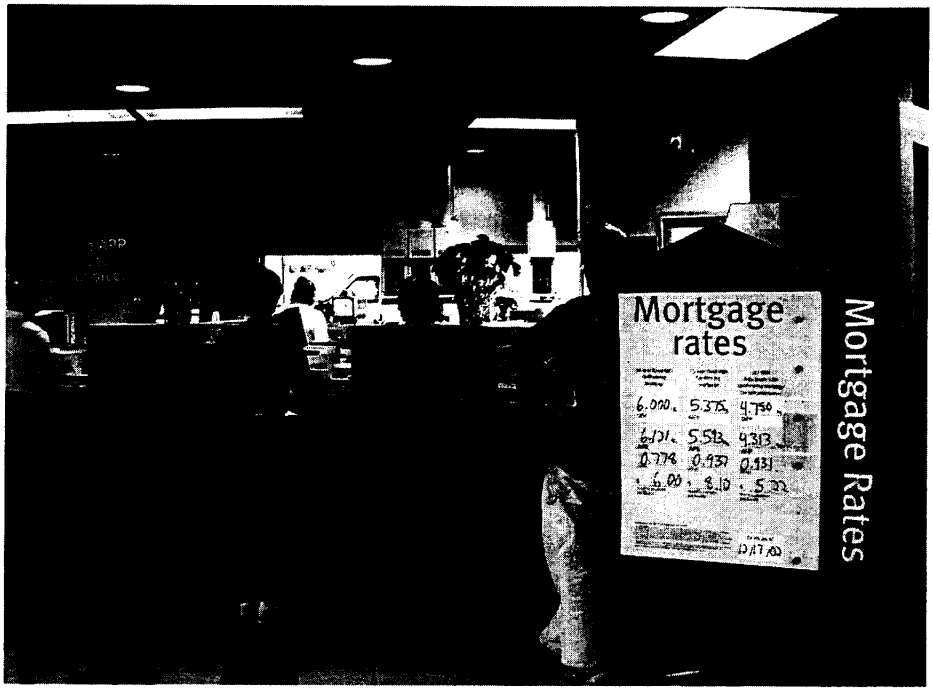

圖2-3、美國銀行小岩城分行營業廳

此行我們特地觀察它的臨櫃作業流程，在服務台有專人針對放款客戶提供商品的選擇、利率期別組合及風險分析，憑藉著是完整的客戶關係管理系統（CRM），透過分行的主機連接至該總部中心主機，以視窗人性化介面供服務人員使用。更令人驚奇的是營業廳外有著跟台灣麥當勞「得來速（Drive Through）」類似的快速車道（圖2-4），直接開車進來，不必下車，就可辦理一般臨櫃存提款業務。

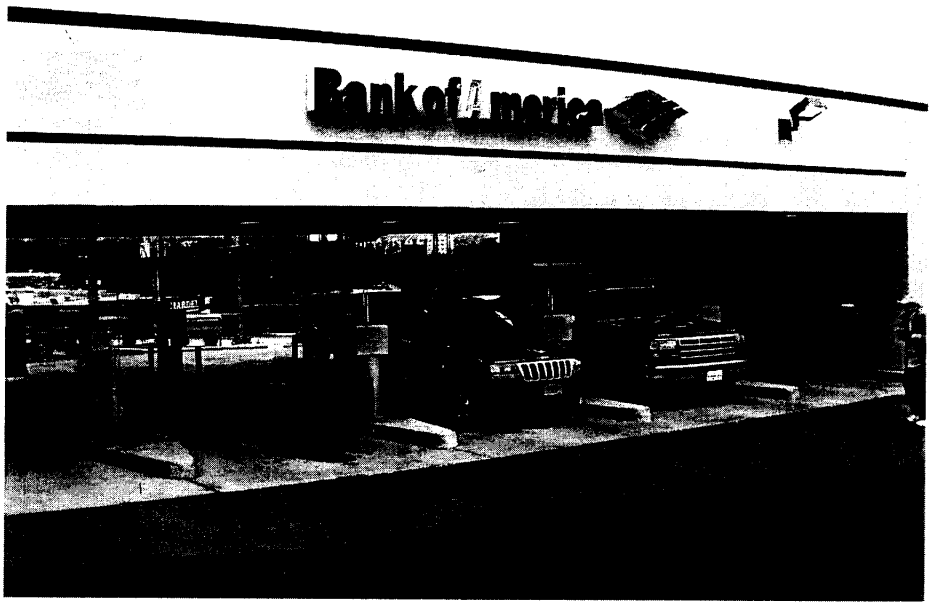

圖2-4、營業廳外類似麥當勞「得來速」Drive Through的快速車道

關於美國商業銀行最著名的一段歷史，是位於洛杉磯的美國商業銀行（Bank of America）於一九五九年開始將信用卡推廣到全加州現在大家耳熟能詳的Visa卡，原名並不叫Visa。Visa的前身叫做Bank Americard，是美國商業銀行在一九六五年所發行的信用卡，一九六六年授權其商標給其他家銀行發行這種藍、白、金三色帶圖案的Bank Americard。為了進一步擴充業務，美國商業銀行成立National Bank Americard Incorporated（NBI），而原先使用Bank Americard授權商標的銀行，就成了NBI的非持股會員。藉此，美國商業銀行也將業務一舉擴充到美國國境之外。不

過由於Bank Americard的名稱，具有太過強烈的美國本位色彩，較難受到國外銀行的青睞。於是在一九七七年，Bank-Americard正式改名為Visa，改名後的Visa果然是勢不可擋，發卡量逐年遽增，穩坐住信用卡機構的第一把交椅。

總部位於北卡羅來納州（North Carolina）的美國商業銀行（Bank of America）是世界上最大的銀行控股公司之一。它提供了各種銀行業務和其他金融服務給全美的消費者和企業經營者，並透過1,800個完全服務分行、約7,700台ATM、電話、和其他傳遞通路提供服務給某些國際市場。這些服務包括零售存款信用卡房屋抵押、汽車貸款融資、投資銀行業務和資金市場和信用產品。透過其銀行子公司和各式各樣非銀行子公司，美國商業銀行提供各種銀行業務和非銀行金融服務和產品，主要遍佈美國之中大西洋、東南、西南、西北、西部區域和某些特定國際市場。該公司共服務這些地區約3,000萬個家庭和200萬家企業。

參、研習心得

分行端末系統除了提昇系統穩定度與提昇系統執行效能外，如何整合未來金融服務作業平台，如何有效降低安裝建置時間及開發、維護的成本，及如何提昇系統的擴充性與延展性等，才是新一代分行端末系統必須考量的重要課題。如何利用新的資訊技術有效解決上述問題，提供高效能及多元化的分行臨櫃服務，是分行端末系統未來發展的重要目標。

一、本行分行端末系統現況

本行分行端末系統自開始電腦化以後，近年來歷經了二個主要階段的整合及調整，其中第一階段調整為民國八十五年間決定將本行分行存放款及託收票據系統由原本屬於傳統專屬架構進而轉換為主從式（Client / Server）開放系統架構，將原本侷限於專屬而封閉的系統架構轉換為開放系統架構，而開放的主從式架構有助於未來分行端末系統整合。

繼民國八十五年採用了開放式主從架構為本行未來端末系統之系統架構之後，為了讓分行端末系統能提供更穩定且更有效率的服務，本行在民國八十九年間開始第二階段的調整。經過長達約半年之久的轉換期，將原本屬分行端末系統之放款系統轉換為與後端中心主機直接連線的中心放款連線系統，將所有端末系統中的放款客戶資料轉換儲存於中心主機資料庫中，所有放款戶之帳務處理皆為透過中心連線交易，即時更新中心放款資料庫中的

檔案，同時也加入額度控管機制，易以作客戶放款額度之追蹤控管並降低授信風險。自此之後本行分行端末系統才正式走入了集中式管理系統架構，即所有的檔案資料皆儲存於中心主機資料庫之中，端末系統透過連線交易即時同步更新中心資料庫相關檔案，這種集中式管理架構的方式，也讓本行端末系統面對未來整合之路，有一個很好的基礎。

而為人詬病的是由不同廠商開發維護，各廠商所使用之分行伺服器不一致，分行端末軟體版本也隨之略有差異，因此軟硬體建置成本與維護費用皆倍增，分行端末軟體版本更是不易控管。另外因應分行業務需求增加，應用程式修改、換版頻繁，維護人員須與不同的端末廠商溝通本行需求與交易功能，因此如何減少重複開發、測試、維護、管理及人力成本之浪費，如何降低軟體部署所需成本，及如何減少因端末軟體版本控管不易而造成系統不穩定，是本行目前亟需解決之課題。

二、分行端末系統架構的演進

(一)二層式主從架構

相對於電腦語言的發展，資訊系統架構自從區域網路的技術有了突破性的發展後，電腦的計算（Computing）也由原來以大型主機為基礎的方式，進入了分散式計算（Distributed Computing）的新紀元，各種分散式計算的模型也相繼被提出，這之中最簡單也最主要的模型，莫過於主

從式架構（Client-Server Architecture）。主從式架構的出現，為分散式系統立下了一個良好的參考模型，Socket、RPC、乃至於分散式物件，都可以用主從式架構來描述。最簡單的主從式架構模型即所謂的傳統二層式主從式計算模型（2-Tier Client/Server Computing Model），目前本行分行端末系統採用的即是屬於傳統二層式主從式計算模型。

在二層式模型中，原本單一程式所要處理的使用者介面（User Interface）或資料展現層（Presentation Layer）、應用程式邏輯（Application Logic）或企業邏輯（Business Logic）與資料層（Data Layer）或資料庫層（Database Service）可以依不同需求拆成用戶端（Client）與伺服端（Server）兩部份。雙方的互動企業處理流程即應用程式邏輯（Application Logic）通常包含於用戶端內。

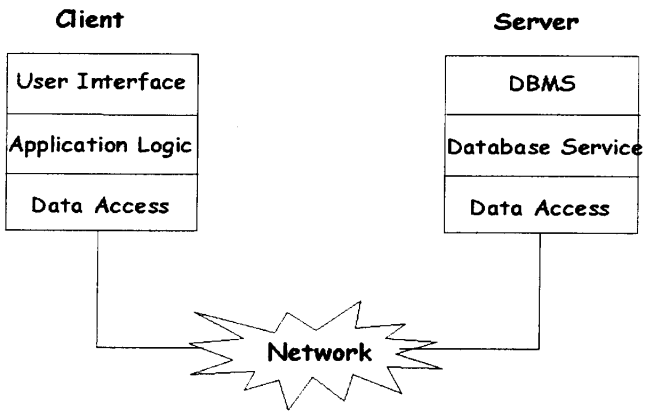

圖 3-1、二層式主從式計算模型

(二)二層式主從架構的極限

在二層式的模型內，用戶端程式直接與伺服端程式溝通，這意味著用戶端程式對不同的伺服端程式必須有不同的溝通方式。但這種二層式主從架構有兩個主要缺點，其一是因為分行交易的負載（Loading）全都集中在分行伺服器上，因此伺服器同時可接收與處理的交易量就有其限度；另一項缺點是一旦分行伺服器發生故障，該分行即無法提供交易服務。

若從Web應用程式的角度來看二層式主從架構，現今的Web服務大都使用CGI（Common Gateway Interface）由Web Server根據需要呼叫各種服務程式，當然我們可使用Perl、VBScript、JavaScript等Script語言來撰寫CGI程式，但其執行效能不佳。雖然後來有ASP（Active Server Page）、JSP（Java Server Page）等效能比CGI好的機制問世，但只要二層式主從架構的結構沒有改變，根本的問題仍無法解決。從資料層（二層式主從式計算模型）的角度來看，Web Server為應用程式邏輯的一部分，所以是屬於用戶端（Client）的一部分。在此狀況下，因應用程式邏輯功能所提昇的部分，對於Web Server的整體結構問題，並沒有多大助益。

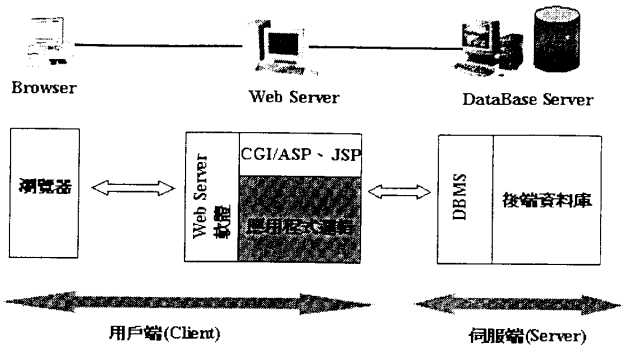

圖3-2、二層式主從架構的極限

(三)擴充成三層式主從架構

早在大型電腦主機時代的整批式的交易服務系統（Batch Transaction Processing System）中，就引入了「第三者」的觀念，在前端的交易系統並不直接將要處理的交易（Transaction）交給後端真正處理交易的程式，而是丟給一個交易處理監控伺服器（Transcation Processing Monitor, TP Monitor），例如IBM的CICS以及BEA的TUXEDO等，待累積足夠的交易後，這個第三者才將這些交易根據一定的規則交由後端程式處理。

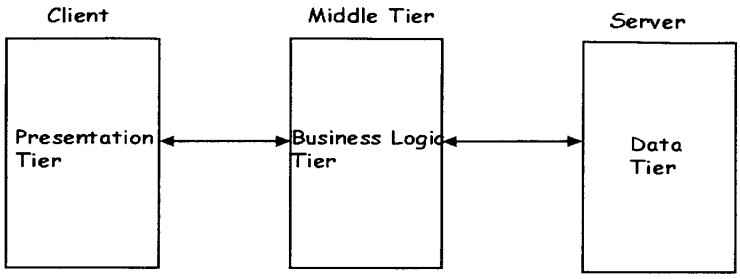

圖 3-3、三層式主從式計算模型

同樣的觀念應用在二層式的模型上，就逐漸發展出三層式的主從式計算模型（3-Tier Client/Server Computing Model）。簡單的說，三層式的模型就是在原來的用戶端（Client）與伺服端（Server）中間，插入一個中介層（Middle Tier）。在資料處理的領域，更明白地將三層式架構中，各層的任務作清楚的區分：資料展現層（Presentation Tier）－主要為使用者介面、處理邏輯層（Business Logic Tier）－包含操作資料的邏輯、資料層（Data Tier）－資料庫的存取。

對用戶端而言，三層式架構可以省去同時面對許多種不同伺服端的窘境。用戶端可以用一致的程式界面與協定與中介層溝通，伺服端的異質性，則被中介層隱藏住。對伺服端而言，三層式架構免除了直接面對所有用戶端的情形，由另一種角度來看，中介層的存在，可以讓伺服端容易地以「一群」伺服器來提供服

務，而分散單一伺服器的負擔。

我們再以Web應用程式的角度來看，在大多數情況下，只要將層數增加，就能解決二層式主從架構Web Server負擔過重的問題。假設將應用程式邏輯交由另一台電腦負責作為另外一層（Business Logic Layer），這樣一來Web Server的負擔當然會減輕，更因為Web Server不用直接置資料庫伺服器存取資料，因此安全方面的控管也較容易。

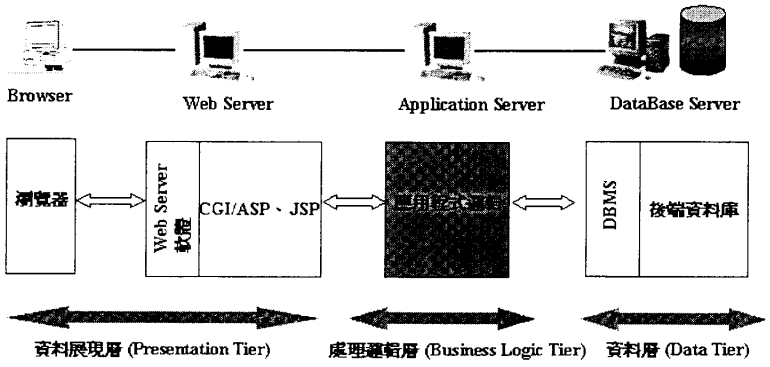

圖 3-4、擴充成三層式主從架構

三、分散式物件技術的興起

電腦網路的急速發展與普及，輕易地實現分散處理，如此一來，降低軟體開發及維護成本，開發具通透性（Transparency）的軟體需求日益增加，而分散式物件（Distributed Object）技術

便是因應這種需求而被發展出來。分散式物件可說是各電腦的物件之間透過網路來形成可相互交換資料的環境，並使軟體資源容易共用的技術。分散式物件是一種可以從遠端系統中呼叫的元件，茲以圖3-5說明客戶端呼叫分散式物件的方式：

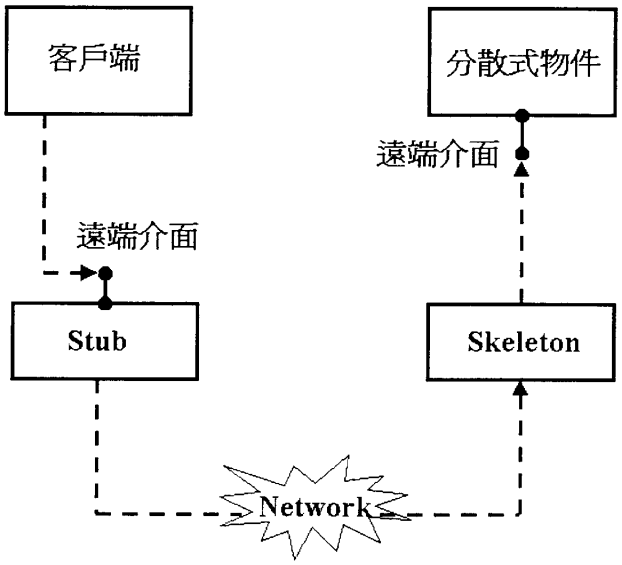

圖 3-5、分散式物件

步驟1 — 客戶端呼叫Stub，此為客戶端的Proxy物件。Stub負責隱藏客戶端的網路通訊，同時Stub知道該如何透過網路以socket及訊息參數等必要網路表現層方式來呼叫遠端物件。

步驟2 — Stub透過網路呼叫Skeleton，此為伺服端的Proxy物件。Skeleton可以協助分散式物件隱藏網路通訊，Skeleton也知

道該如何從socket接收呼叫，同時知道如何從網路表現層中將參數傳遞給Java表現層。

步驟3 — Skeleton將呼叫委派給分散式物件加以處理。分散式物件完成必要的工作之後會將控制權交回Skeleton，再由Skeleton將控制權交回Stub，而Stub則會將控制權交給客戶端。

也就是說，Stub與分散式物件都實作相同介面（稱為遠端介面），呼叫Stub方法的客戶端會以為他直接呼叫了遠端物件的方法，事實上客戶端呼叫的只是知道如何透過網路呼叫遠端物件的空Stub而已。這種做法即是本地/遠端通透性（Local/Remote Transparency）。

而現今在分散式物件模型這方面發展，資訊科技業界就有三大陣營－CORBA（Common Object Request Broker Architecture）、Sun MicroSystem的Java 2 Platform Enterprise Edition（J2EE）和微軟的Windows DNA與.NET。

(一)工業界標準OMG（Object Management Group）的CORBA（Common Object Request Broker Architecture）

CORBA是由八百多家業界的公司參與之OMG（Object Management Group）所制定的標準，其目的是要整合大部份的物件系統。這種物件管理架構（Object Management Architecture, OMA）的主要溝通機制，其中物件與物件間之溝通是透過物件訊息仲裁者（Object Request Broker, ORB）。而ORB是在主從架構間提供一仲裁服務，決定訊

息的流向。CORBA規範介於物件介面的層次（Interface Level），而物件間就以介面描述語言（Interface Definition Language, IDL）來描述。將介面（Interface）與實作（Implementation）分開，提供多重的繼承架構及動、靜態介面的呼叫方式。

CORBA是所有物件模型中發展歷史最為悠久的開放式物件架構標準，所以後來的其他物件模型使用的架構也都有它的影子存在。基本上CORBA擁有比其他物件模型較高的效率及延展性，CORBA物件可以放在網路上任何平台執行、可以利用任何語言撰寫（如Java、C++、SmallTalk等），這和CORBA在設計上就是以大型企業應用架構為中心觀念有很大的關係，再加上能夠整合SSL安全機制，因此獲得許多大型的電信系統及金融系統的青睞。

(二)Sun MicroSystem的Java 2 Platform Enterprise Edition（J2EE）

Java是昇陽微系統（Sun MicroSystems）所提出結合了程式語言和虛擬機器（Virtual Machine）的分散式物件環境。而J2EE架構則包含整個完整的Java架構，其中包含的技術包括Servlet、JSP（Java Server Page）、Java Applet、EJB（Enterprise Java Bean）等相關的技術，其中Presentation Tier利用Java Application、Java Applet、Java Servlet、JSP處理使用者介面，Business Tier則是利用EJB中Session Bean、Entity Bean等元件技術去處理商業流程邏輯。不論Java

Application、Java Applet、Java Servlet或JSP皆可利用Java命名技術JNDI（Java Naming and Director Interface）與EJB溝通，而其溝通協定則使用RMI（Remote Method Invocation）或IIOP（Internet Inter-ORB Protocol）的方式去呼叫遠端EJB元件中的方法。如果是CORBA的前端，也可以利用CORBA的命名服務（COSNaming）來後端的EJB溝通。另外，Java也提供了JMS（Java Message Service）服務來傳遞非同步性的資料。在Data Tier包括後端的資料庫系統以及企業以存在的資訊系統，EJB則是利用JDBC（Java Database Connectivity）的資料庫介面技術與資料庫進行溝通。如果要連結舊有已存在的資訊系統，則可透過Java Connector的技術來作連結。

(三)微軟的 Windows DNA與.NET

微軟（Microsoft）的DNA架構與J2EE一樣，都適合用來建構企業分散式架構的解決方法。最大的不同是，Windows DNA架構只支援微軟平台的非開放性架構，而J2EE則支援開放的平台。Windows DNA的核心是COM+元件技術，利用特定的程式語言開發軟體（Visual Basic、Visual C++等），用來設計可重複使用的元件。微軟的COM+就是根據其原本非分散式物件模組COM（Component Object Model）所進化而來的，它已被定為新一代電腦語言的發展基礎。COM+其實是二次元層次（Binary Level）的

分散式物件介面標準，它是由圖形使用者介面應用程式之間溝通的機制逐漸轉變為Windows平台下的物件模型，並進而演進成為Windows平台下分散式架構的核心技術。也顯示了Microsoft欲從以桌上型應用為主企圖轉變為以企業應用為主的決心。

因此，微軟在二〇〇一年發展新的分散式物件技術架構，稱之為「.NET」。.NET架構中以COM+為核心的技術上，仿造Java的JVM（Java Virtual Machine）架構，在作業系統上架了一個CLR（Common Language Runtime）架構。除了可與COM/DCOM相容外並增加多項新功能，支援多層式系統開發的技術、產品，COM+是屬於Windows 2000的技術體系，而原本裝載Window NT 4.0的MTS（Microsoft Transaction Server）與MSMQ（Microsoft Message Queuing）等技術，在Windows 2000已經以COM+這種新的執行時期環境的形式納入作業系統中。

四、分散式物件與應用伺服器

分散式物件的用處相當大，因為透過這種做法可以讓應用程式散佈到網路上。不過一旦分散式物件應用系統愈來愈龐大時，就需要「中介軟體」（MiddleWare）服務的協助了，例如交易服務與安全性服務等。目前典型的企業應用系統，當然包括銀行分行端末系統均為分散式系統，如要這類典型的大型系統切開

來，劃分成各個層次，而且每一層之間都是完全獨立且各自不同，當將大型系統切成多個客戶端跨越網路連接到多台伺服器與資料庫的分散式系統後，最需要關心的恐怕就是「中介軟體」了！

過去很多公司都自行設計中介軟體，如很多金控公司都積極建立中介軟體服務以便與股票交易系統串在一起，但是不少公司都在設計中介軟體的過程中遭遇風險且面臨失敗的命運。因為建立與維護高階中介軟體是相當複雜的，需要有專家級的知識而且這些知識與大多數公司本身的核心事業毫無關係，既然如此，於是很多提供中介軟體服務的「應用伺服器」（Application Server）的廠商就應運而生，甚至部分廠商也提供可以解決部分企業問題的方案。因此，企業本身只需專注於如何選擇應用伺服器及應用系統的開發上，而不需著墨於強固伺服端部署環境的中介軟體開發上。

(一)應用伺服器的基本運作方式

應用伺服器的基本思考方向，在於如何將企業舊有應用系統、資料整合到多層式Web應用程式上，以目前的應用伺服器架構而言，大都追隨符合J2EE的架構或微軟的.NET架構，一般應用伺服器架構應該是圖3-6所示，應用伺服器將負責處理邏輯，希望能減少WebServer的負擔，讓Web Server回歸處理HTML展現的基本功能，專門負責將Web Page展現到Web Browser上即可。所以應用伺服器就是負責

處理從後端來的資料並轉換成HTML格式，透過Web Server以HTTP展現給前端的瀏覽器。這樣一來，前端使用者介面設計者就可專心撰寫HTML檔案，應用程式開發人員則進行JSP、Servlet或Java程式的開發。此外，應用伺服器透過其所提供的負載平衡、錯誤回復、效能提昇等功能，以配合複雜的商業處理邏輯需求。

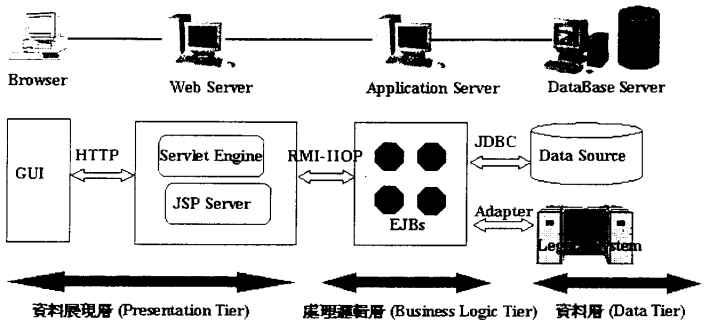

圖 3-6、典型的應用伺服器架構

(二)應用伺服器的技術架構

一般而言，應用伺服器的技術架構，不同廠商會有不同的架構方式，茲以J2EE為架構的應用伺服器為例，其使用的核心技術有：

Java Servlet Engine — Java Servlet技術，接收前端HTTP的請求（Request）後，將執行結果（Response）轉換為HTML

送回前端。

JSP Server － 使用JSP（Java Server Pages）技術，因其為一種描述語言（Script），使得開發人員可以比較容易開發資料展現層（Presentation Tier）的程式，然後轉換為Java Servlet Code讓Java Servlet Engine執行。

EJB Container － 利用EJB（Enterprise Java Bean）技術來執行處理邏輯（Business Logic），開發人員可以將常用的程式邏輯寫成EJB，然後提供給Java Servlet或JSP程式去呼叫，應用伺服器本身通常會提供EJB程式發展與對應的開發工具，來開發與部署（Deploy）應用程式。

JDBC － 利用JDBC（Java DataBase Connectivity）去存取後端資料庫。

Adapter － 為了能讓應用伺服器能夠連結到各種不同的後端系統，一般應用伺服器會提供的Adapter會依循JCA（Java Connection Architecture）規格連結後端系統，以整合舊有系統。

多層式伺服器部署環境的觀念已經發展多年，發展至今市場上已經出現50種以上的應用伺服器了。剛開始市場上所出現的是沒有標準且專屬的元件服務伺服器，這種應用伺服器的出現主要是因為當時並沒有共通的元件定義，一旦選擇了某種應用伺服器之後，所設計的程式碼就必須綁死在某家廠商的解決方案上。如此便會降低系統的可攜性，但也由於這種緣故讓強調開放性與可

攜性的Java崛起。也說明了為何應用伺服器廠商大都採符合J2EE標準的EJB元件架構，EJB架構裡面主要有三大價值，茲說明如下：

產業標準：使用EJB的廠商會因為EJB架構廣受歡迎而獲得好處，由於大家使用相同標準，因此未來很容易僱用熟悉系統的員工（至少具備EJB的相關知識及經驗），也可將最好的實作經驗應用到企業的系統中，同時也可以與企業夥伴一起攜手合作，此即所謂的「訓練一次，隨處可用」的觀念。

更具可攜性：EJB規格是公開且可以免費取得的，由於EJB是一套標準，因此不需要將賭注下在某家專屬廠商的架構上。因為專屬技術絕對不可能免費，而且只要是不具標準的技術其價格就不會便宜。

快速應用程式開發：因為EJB元件（稱為Enterprise Bean）是可以部署的元件，而放置元件的應用伺服器可以匯入或載入元件，可以加速應用程式的開發，因為從應用伺服器就可取得中介軟體服務，同時也可降低維護成本。

(三)應用伺服器的種類

而目前為止，因為分散式物件技術的成熟與Java的諸多特點，使得Java很適合用在企業運算（Enterprise Computing）領域，而EJB（Enterprise Java Beans）可以說是其中最重要的技術。目前除了微軟之外，所有的應用伺服器（Application Server）幾乎都是支援EJB的。支援開發符合

J2EE標準的EJB架構平台的應用伺服器軟體已不在少數，其中較具代表性的有IBM WebSphere、BEA WebLogic、iPlanet Application Server，以及Borland公司所推出的AppServer等等。但基於不同公司的產品戰略以及研發方向，各個產品在服務性能、對程式語言的支援和作業系統平台方面有稍有差別。而且很多EJB伺服器都以CORBA為基礎，且可適用CORBA架構。相對地，微軟公司也積極發展.NET的新一代分散處理技術，想要為目前蓬勃發展的電子商務中的B to B（Business to Business）、B to C（Business to Consumer）與企業業務系統中的B to E（Business to Employee）等各種系統，提供系統建置與運用的環境。此外，.NET則進一步讓DCOM物件能與CORBA物件，或在Java環境開發，於EJB環境下運作的物件互動。

IBM宣佈新版的WebSphere應用伺服器（Websphere Application Server），通過J2EE（Java2 Enterprise Edition）1.3相容認證，為業界首先通過者。IBM表示，WebSphere通過 J2EE 1.3認證，是為了能更確實支援XML與網路服務（Web services）標準，協助企業整合既有的非Java資訊系統和新的Java資訊架構架構。IBM指出，最新版的J2EE 1.3標準具備簡化企業整合與開發工作流程、及自由選擇平台等效益，其中JMS（Java Message Service）功能讓應用程式之間能以非同步的方式建立、傳送、接收與閱讀訊息。對程式

開發人員而言，不但可以運用最新的網路服務應用程式介面如XML、SOAP與UDDI，建置最新的應用程式，並可沿用以JSP（JavaServerPages）、Servlet或EJB（Enterprise Java Beans）2.0等標準開發的既有應用程式與相關軟體。

比爾亞系統（BEA Systems, Inc.）是全球電子商務應用基礎架構中介軟體（Middleware）領導廠商，為全球超過13,000個客戶提供企業軟體基礎架構，其中包含財星全球500大企業中的多數。BEA及其商標WebLogic是電子商務領域中，在全球有超過12,500多家用戶，其中包括《Fortune》雜誌全球500大中大部分公司。

構建在BEA軟體上的企業能夠採用IT來實現組織內的快速變革，並在企業運營效率和回應速度方面實現突破。BEA WebLogic提供的應用基礎架構，能夠簡化資訊流，降低應用管理成本，使企業更靈活、更富效率，並達成全面的整合。BEA的平台也是2,100多家系統整合廠商（SIs）、獨立軟體供應商（ISVs）和ASP的既定標準，他們與BEA合作，以確保企業解決方案的成功部署。BEA WebLogic Enterprise Platform是一套「Unified」的軟體基礎架構，其設計宗旨是順應用戶的需求－採用更簡單的方法開發、部署、整合和管理企業應用和Web服務。

BEA WebLogic Platform的核心部分是BEA WebLogic Server，並包括BEA WebLogic Portal、BEA WebLogic

Integration和一個新的應用開發和部署框架BEA WebLogic Workshop。BEA WebLogic Enterprise Platform整合功能：BEA Weblogic Server7.0、BEA WebLogic Portal及BEA WebLogic Integration現已是高度整合的解決方案。WebLogic Portal和WebLogic Integration都以WebLogic Server為基礎，且都將WebLogic Server納為該產品的元件之一。WebLogic Enterprise Platform所提供的強化功能，是BEA對原有整合能力提升的結果，它為開發人員、管理人員及合作夥伴，創造了一個更為統一的解決方案。

BEA也承諾將在未來推出的版本中擴充新的整合功能，同時，軟體元件的功能也將再度加強，以確保WebLogic企業平台在市場上的優勢地位。BEA WebLogic Server 7.0－新版的BEA WebLogic Server為大幅度改版產品，功能也大幅強化，包括Web Services、效能、使用性、J2EE支援、企業訊息處理能力、安全性及管理能力。WebLogic Enterprise Platform是WebLogic Server 7.0的基礎。BEA WebLogic Portal－WebLogic Portal是以WebLogic Server為基礎的產品，可用以建立各種入口網站（Portal），如員工、消費者及合作夥伴等入口網站。它提供完整的基礎服務，可讓開發人員輕鬆建立複雜的入口網站，以改善使用者的使用經驗，並評量目標使用者的互動情形，協助企業營運成功。BEA WebLogic Integration－BEA WebLogic

Integration以WebLogic Server為基礎，為企業提供應用程式整合、營運流程管理及企業對企業整合功能。它提供標準化的「依整合需求而建置」（build to integrate）的方法，容許企業建置新的應用程式，並整合既有的ERP、CRM及MRP軟體，使複雜的營運流程更順暢，並與事業夥伴連結在一起。

iPlanet應用伺服器（iPlanet Application Server）同時支援以C/C++來開發應用系統，以及以J2EE元件為基礎的Java來開發應用程式；J2EE元件包含中間層可重用之企業運算規則的EJB架構、Java Servlets，以及JavaServer Pages等技術。J2EE標準提供了一個統整、以元件為基礎的應用系統模型，可以用來處理多層式應用系統中用戶端、後端資料庫以及舊有資源間的異質性問題。支援J2EE可以減少開發的時間。應用系統開發環境－iPlanet Application Builder可以讓開發人員快速開發多層式（multi-tier）的應用系統。透過精靈技術的使用，可以自動產生應用系統中的元件，包含：Entity與Session兩種EJB、Servlets、SQL查詢以及其他的元件。延伸模組開發環境－iPlanet Extension Builder提供一組開發工具，可以讓企業中的開發人員自行開發伺服器的延伸模組，以便整合企業內部各項舊有系統、應用程式以及Internet各項服務。iPlanet也提供了幾項預先建立的伺服器延伸模組，能夠與BEA Tuxedo；SAP R/3；PeopleSoft；IBM

CICS、IMS及MQSeries系統相連結。

Borland AppServer Edition是提供的一個可靠、具延展性的J2EE平台。支援最新的業界標準包括：J2EE 1.3、EJB 2.0、JMS 1.02、Servlet 2.3、JSP 1.2、XML及SOAP。Borland Enterprise Server架構在Borland VisiBroker之上，得以支援CORBA 2.4規格。此外，與Java開發工具－JBuilder緊密整合後，無論開發無線（wireless）應用程式或是Web應用程式，皆可提昇企業系統開發效率，縮短開發與部署之周期。Borland Enterprise Server支援分散式交易、安全性、訊息交換，並具備叢集（clustering）、負載平衡（load-balancing）和錯誤回復（fail-over）等功能，使得Borland Enterprise Server, AppServer Edition成為部署J2EE應用程式的最佳平台。AppServer Edition特有之分區（partition）技術、叢集技術和資料複製架構，能更有效率和彈性地分配系統資源，多個關鍵任務的應用程式或服務，可同時在一台應用程式伺服器上確實、獨立地執行。伺服器分區（Server Partitions）技術能夠在單一主機上提高效能，亦可跨越網路善用公司所有的硬體設備。延展性和效能最佳化的設計，讓AppServer Edition得以降低建置J2EE應用系統時的硬體設備成本。

五、百家爭鳴之多層式分行系統架構

根據前節三層式系統架構、分散式物件技術及中介軟體介紹後，可發現不論國內外電腦公司，近來皆紛紛提出三層式（多層式）的Web Based應用系統架構，因為隨著網路的蓬勃發展，許多因應網路環境而生的金融服務也相繼推出，網路銀行（Internet Bank）就是其中之一。網路銀行是使用現今標準的網際網路技術與交易環境，執行銀行服務與帳務性交易，即以瀏覽器（Browser）整合Java Applet、Active X、JSP、ASP、Servlet及EJB等技術完成交易的過程。

未來整合性金融服務平台方面，將含括電話、大哥大、有線電視衛星通訊……等。銀行利用整合性金融服務平台提供之服務將除了原有的分行據點（Branch）外尚有自動櫃員機（CD/ATM）、電話語音（IVR）、自動無人銀行（Kiosk）、電話服務中心（Call Center）、網路銀行系統（Internet Banking）、行動銀行（WAP、Mobile Banking）、家庭銀行（Home Banking）等。站在銀行整體金融服務營運規劃的角度分析，分行端末系統要考慮使用多層式分散式物件環境，以符合銀行未來多元化的金融服務需求。

茲列舉此次訪察的國外電腦廠商與國內幾家知名電腦廠商，針對其提出之多層式Web Based分行端末系統軟、硬體架構比較分析如下：

(一)OrbiTech新一代分行系統架構

OrbiTech與ALLTEL都是與BEA合作的系統整合廠商，OrbiTech所提出的分行端末系統（Orbi-Teller System）主要為一四層式（多層式）的分行系統架構（圖3-6）。所有分行工作站前端使用者界面的網頁頁面，完全由位於Web Server的JSP（Java Server Page）設計產生，而其採用的EJB（Enterprise Java Bean）架構則為BEA公司生產的應用伺服器（Application Server），其核心為Web Logic Server。另外以JDBC（Java DataBase Connectivity）存取後端Oracle資料庫中的Stored Procedures以加速對資料處理的速度。Orbi-Teller System所設計的銀行企業邏輯元件擁有一致的介面，可提供各種前端應用程式使用（如櫃檯機與ATM等），換言之，其具有可重用性（Reusaility）。而由於它採用的是多層式的架構，各層皆獨立完整可保有彈性及延展性。以下所列為其軟、硬體架構：

- Hardware & O/s：UNIX （SUN, HP, IBM, Compaq-Digital）
- Database System：Oracle
- Languages：Java, EJB, C, PL/SQL
- Application Server：WebLogic
- Web Server：iPlanet
- Information Security through：Orbi-Armor

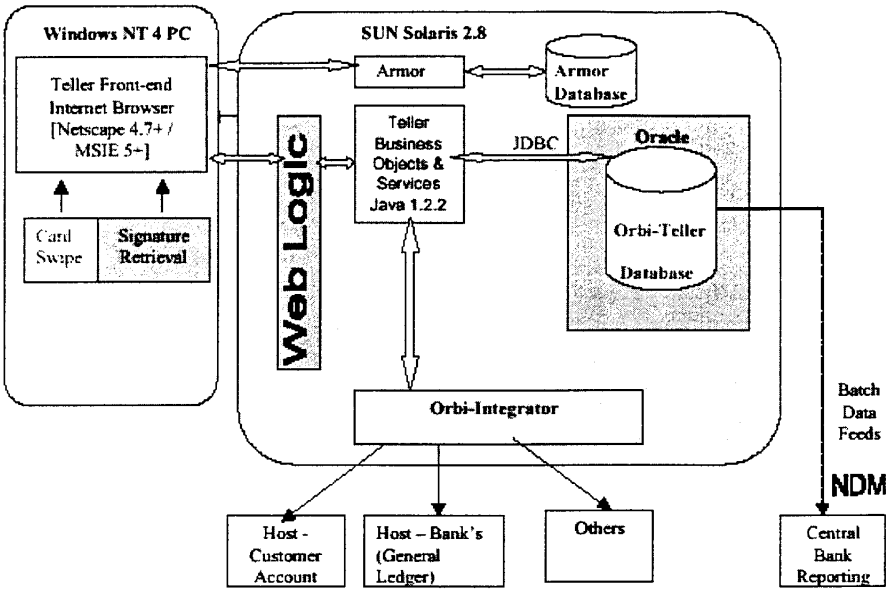

圖3-7、OrbiTech新一代分行系統架構

(二)和平整合資訊新一代分行系統架構

和平整合資訊公司（Peace）是ALLTEL公司在台灣的代理合作的一家整合資訊公司，其提出的新一代分行系統架構採用的是微軟公司的分散式物件技術.NET解決方案，其為一三層式的分行系統架構（圖3-7），前端設備均具備瀏覽器功能（Browser），瀏覽器則利用HTTP與Web Server中的伺服器軟體IIS，以ASP（ActiveServer Page）來做溝通，而在Web Server內以COM/DCOM技術開發之各種COM物件，則透過位於應用伺服器（Application Server）中COM+的MTS（Microsoft Transaction Server）來發揮交易控制等各

種功能。而其開發的各種銀行企業邏輯元件（存、放款、外匯等），則位於.NET Componet Server中，可透過一致的介面供前端使用。另外，.NET Componet Server中的COM物件則透過ODBC（Open DataBase Connectivity）或ADO/OLEDB與後端資料層（各種資料庫伺服器）溝通。

- Hardware & O/s：Windows （Windows 2000 Server）
- Database System：SQL Server
- Languages：VB,C#（Visual Studio .NET）
- Application Server：Microsoft .NET Enterprise Server
- Web Server：IIS

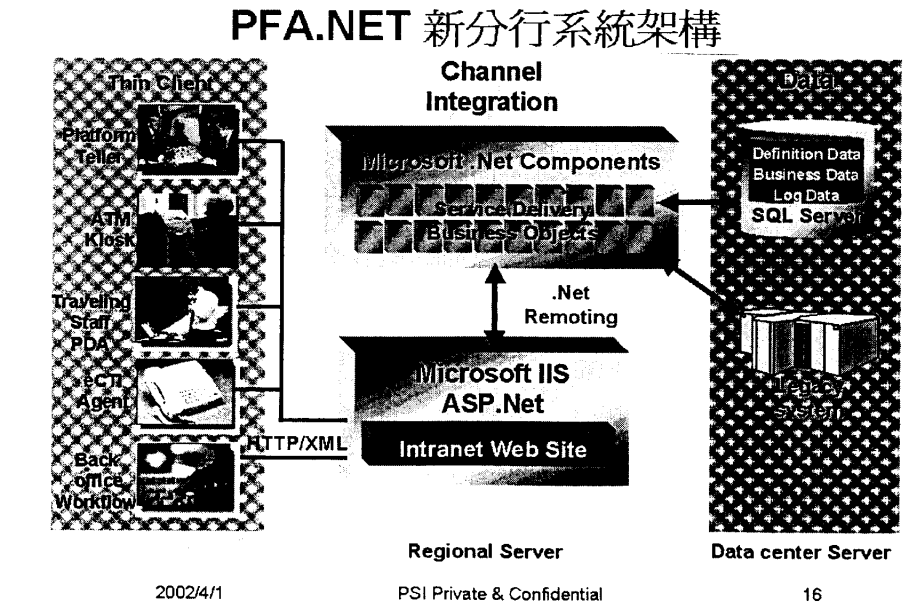

圖3-8、和平整合資訊新一代分行系統架構

(三)鼎盛公司新一代分行系統架構

鼎盛資料股份公司是目前本行端末系統維護廠商之一，其提出的新一代分行系統架構採用的是以J2EE元件為基礎的Java來開發應用程式；J2EE元件包含中間層可重用之企業運算規則的EJB架構、Java Servlets，以及JavaServer Pages等技術。同樣地，它也是三層式的分行系統架構（圖3-8），使用者端採用標準的Browser介面，接收並顯示Web Server端產生的標準HTML格式資料。由應用伺服器（Application Server）單一和中心主機互動，交換電文。其他不同系統亦可整合由應用伺服器處理。透過JDBC的共通介面，與具開放性的資料庫（Oracle、Informix等）做溝通。另外由LDAP Server負責統一安控，使用者只要login一次即可執行授權的作業。Hardware & O/s：NT/UNIX

- Database System：Oracle、Informix
- Languages：Java, EJB, C, PL/SQL
- Application Server：iPlanet Application Server
- Web Server：iPlanet Web Server
- Information Security through：LDAP Server

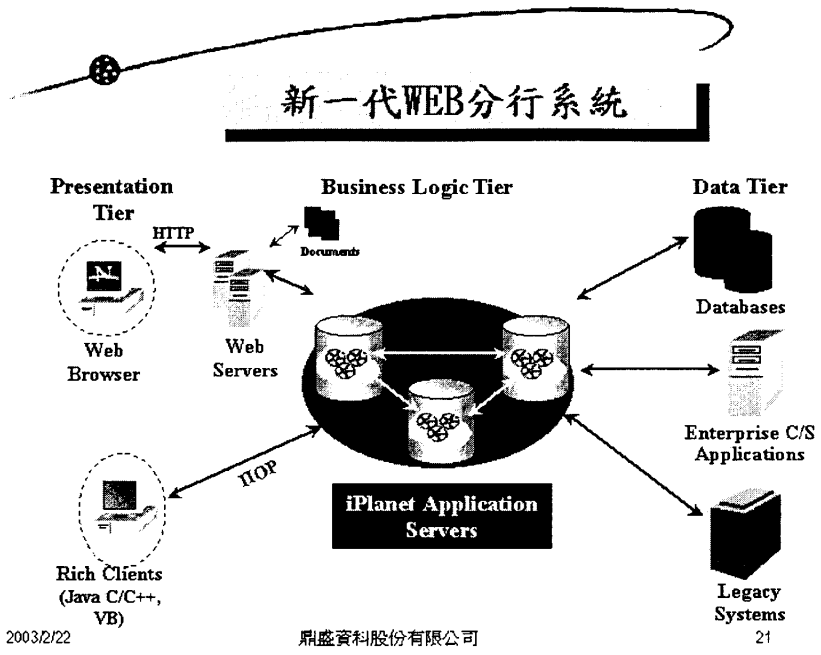

圖3-9、鼎盛公司新一代分行系統架構

六、分行端末系統之整合與運用

金融業已邁向全面自動化服務時代，提供完整的金融產品及全方位的業務服務整合，在日益競爭的金融市場中，已成為競爭力提升的不二法門，依據金融服務未來的實際需求，建置e-Service之服務系統以建立專業客服中心Call Center，處理來自不同通道（Multi-Channel）的客服需求，結合客戶關係管理（CRM）、線上行銷、企業間的電子交易（B2B）等，建立一個整合性、國際性的客戶互動平台（圖3-10）。進而達到分析客戶群特性、增加銷售量、提高客戶忠誠度、降低銀行客服中心的

營運成本及提高利潤等目的，而未來在建置新的分行端末系統的同時，也須考慮如何將分行端末系統整合到金融服務作業平台中。分行端末系統漸朝多重服務通道發展，如何將本行端末系統建置在未來的金融服務平台環境，以降低分行營運成本、提供多元通路資訊、連結全行的資訊資源，創造更多樣化的資訊服務與商機，值得我們對這些議題做深入探討。

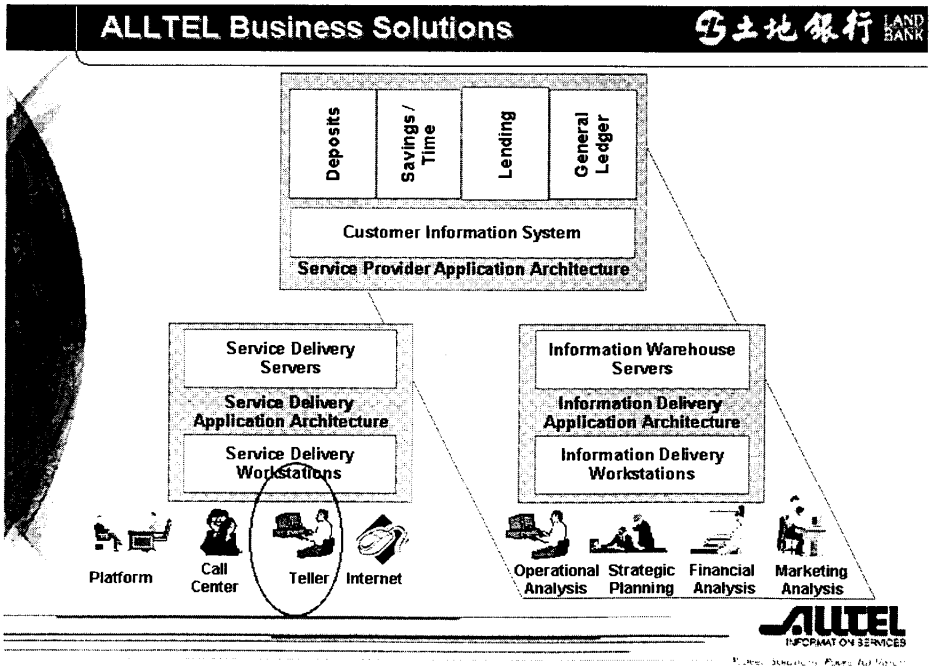

圖3-10、ALLTEL電腦公司未來金融服務發展藍圖

前面幾節談到的，不論是EJB、COM+、CORBA等技術，在企業上的應用上大都是僅限於企業內部系統的開發或整合，原因

是這些技術不但複雜且標準不一，當要整合所有系統時，必須了解各系統所支援的分散式物件遠端呼叫或其應用程式API的規格。然而，應用到企業間的電子交易（B2B）可就不是那麼簡單了，因為軟體或應用伺服器供應商在開發其他系統時（如監理資費前置系統或現正開發中的外匯系統），並不會幫你想到未來系統的整合性，更何況企業的合作伙伴又常常改變，根本無法了解對方的系統。

不過資訊產業界積極思索如何利用標準的訊息交換的溝通方式，因為不論是以往傳統的RPC（Remote Procedure Call）、Socket或RMI的溝通，也都是一種訊息交換的溝通模式，為何不採取一種標準化的溝通方式呢？於是，便有了XML（eXtenson Markup Language）的誕生，XML是一種描述訊息的語言，與HTML文件一樣，是階層式結構的語言。差別在於HTML僅限於描述瀏覽器展示的功能，而XML是可以定義企業自己所要描述的元件。

對於應用伺服器而言，為了整合不同通道的服務（如語音、網路銀行、PDA、WAP等）或不同系統的整合，自然要利用XML的技術了。而其應用方式一般可分為以下幾種層次：

(一)展現層次

除了藉由XML的文件語言來描述資料展現的內容外，還可利用如XLST技術轉換成不同的資料展現方式。利用因應各種端末服務需求，由於端末設備不同而有不同的限制，

前端界面展現的方式也不盡相同。以往傳統主從架構，必須設計不同版本的前端應用程式，不但程式開發困難、維護也不易。如前端交易畫面多一個欄位，就必須修改不同版本的前端應用程式。而利用XML的方式定義了資料內容標準，而轉換方式也是以標準的格式（轉成HTTP或WAP格式），因此，只要依照XML標準格式，就可輕易轉換至不同端末設備的展現方式，無形中為企業省下不少成本。

(二)應用程式層次

不同的技術（如RMI、COM+、CORBA），有著不同的定義、不同的溝通協定，為了這樣的整合，企業必須克服這些不相容的複雜技術問題，為了處理這些惱人的問題，資訊產業界提出了Web Service的概念，Web Service是電腦軟體界目前最熱門的話題，這種在網路上用標準的協定及介面的開放式軟體元件被認為是下一波軟體的革命。Web Service可用無線的手機、桌上電腦、或是從一個應用程式，甚至從另一個Web服務上面，透過網路去呼叫調用的功能，以提供某種服務，諸如是B2C、B2B的網上服務，而這網路可以是Internet、Intranet或是Extranet。亦即將應用程式視為服務的方式來提供使用者使用。比如，銀行提款交易，使用者不須了解它背後所撰寫的程式邏輯，只須告知需要「提款交易」服務即可，這特別適合於企業間的電子交易。Web Service包含服務提供者（Service Provider）、服務使用者

（Serice User）、服務註冊處（Service Registry）。服務提供者將服務註冊於服務註冊處供服務使用者取用。而Web Service有以下幾種XML標準規範：

SOAP（Simple Object Access Protocol）－簡單物件存取協定，用來定義Request/Response方式的訊息，管理佇列的儲存、發送與接收。可以將它視為是XML標準的RPC定義。

WSDL（Web Services Description Language）－用來描述服務的介面協定。可以將它視為是類似RMI中的IDL（Interface Definition Language）定義。

UDDI（Universal Description Discovery and Integration）－搜尋服務之電話簿，用來規定如何去註冊服務。可以將它視為是類似JNDI中的命名服務。

(三)資料層次

XML技術將改變我們交換和處理資訊的方式，我們能在接收到資訊的同時，根據我們自己的需要來處理這些資訊。例如：我們可以使用XML語言描述的公司的訂單，這些訂單可以和資料庫結合在一起，在不同的程式中自動更新庫存和出貨記錄而不需要重新輸入資料，同一份訂單在不同的應用中可以有不同的涵義，在採購部，可以賦予訂單編號、指定客戶編碼與修改金額，而供貨廠商只能確認訂單和修改金額，收貨人則只能查看、存儲或列印這份訂單，但是，不論在哪一種情況下，實際上所使用的是同一份文件，

而根據不同的使用者與使用時機，會有不同的行為。

利用XML的好處是可以定義自己需要XML結構，這樣的XML結構可能來自資料庫實際架構的轉換或是資料庫架構的一小部分。如此一來，程式設計師不用操心資料在資料庫中實際對應的欄位。

肆、結論與建議

一、結　論

最後歸納本論文所要強調的大都是系統架構或技術面的探討，其實從現今資訊技術的發展趨勢就可看目前企業需求的一些端倪，因為電腦科技的發展始終是跟著使用者（企業）需求而走，不管是多層式系統架構或分散式物件技術都有其演進的過程，所謂「網路服務只是分散式物件模型演進過程下一步」的觀念，或許有些相關的研究報告討論過分行端末系統相關議題，但大多是業務面的探討，在本文中不須再贅述。

此次赴國外參訪主要目的在於瞭解國外電腦公司、外商銀行對分行端末系統作業未來之發展及應用方向，了解分行端末系統趨勢及重要技術。因此我們瞭解最新的資訊技術趨勢之後，配合本行未來端末系統需求才能決定採用何種系統架構？有了正確的大方向之後，對於使用者介面需求、整合現行系統的需求等等的問題，只需要建立完整的需求功能清單，衡量每一個功能的重要性，根據功能的重要性排定其優先順序。

二、建　議

根據Gartner Group的報告指出，二〇〇〇年企業在多層式的應用伺服器系統架構上的花費超過10億美元，因此如何決定未來正確的系統架構？便成為我們需要思索的課題。以下將列舉幾項建議，除了支援基本J2EE的功能外，需要考慮到應用伺服器系

統其他管理及支援功能，以作為評估的參考：

(一)支援跨平台（Cross Platform Supports）

一個應用伺服器系統應該能夠支援跨平台的作業功能，亦即能夠很輕易地將某一平台開發好的軟體移植到別的平台，目前應用伺服器系統大都追隨Java架構，應該都能夠符合這樣的需求。不過儘管如此，不同的應用平台為了開發程式的便利及差異性，某些應用伺服器系統會使用獨特的應用程式介面（API）。使得以Java開發應用軟體的可移植性功能大打折扣。

(二)可再使用的軟體元件（Reusable Software Components）

開發可再使用的軟體元件時，必須做好現實考量，不只盡信Java規格定義中的Enterprise Java Bean是可以重複使用，要讓Bean可以重複使用（例如要將存款系統中的存、提款交易包裝成一個Bean，供其他系統作連動交易時重複使用），需要考慮不同的應用系統，不同的使用者，同時須儘可能以最彈性的方式設計，絕大多數的軟體專案都因為時間限制與缺乏彈性等問題，無法將軟體元件好好利用。

(三)負載平衡（Load Balance）

所謂的「負載平衡」是指能讓工作負擔平均分擔置不同的機器上或行程以減輕負荷的瓶頸，一個大型的企業應用伺服器的行程配置方式，一般利用網路的負載平衡器（Network Load Balance Machine）來分配進入的HTTP

Request而給予多台的Web Server及多台的Application Server來做負載平衡。J2EE架構應用伺服器都是利用Multi-Threads的架構；亦即，開啟一個行程即利用一個Java Virtual Machine，而在上面執行的應用程式都是這個JVM的一個執行緒（Thread），每個Application Server上可能跑一至多個JVM，每個JVM有數個執行緒（Threads）以執行各個應用程式。一般負載平衡的設計，有以下幾種方式：

1.集中式管理

集中式管理意味著會有一台管理伺服器來記錄、控制目前所有的應用伺服器，所有需求都會先到這個管理伺服器後才會分配到各個應用伺服器。優點是集中管理、校調容易；缺點是管理伺服器會成為效率瓶頸，若當機則造成其應用伺服器無法運作。

2.分散式管理

每個伺服器都會記載目前各個應用伺服器的狀況，根據某種特殊演算法（如：Random或Round-Robin）的方式，將Request導向某個應用伺服器再交由較為空閒的伺服器處理。優點是延展性高，可隨時新增一台應用伺服器；缺點是無法做最正確的資源分配，同時輪詢（Pooling）也會浪費頻寬與效率。

3.叢集式管理

叢集管理的意義在於將一群應用伺服器視為一個整體，這

些應用伺服器可能透過共享記憶體（Shared Memory）、共享硬碟（Shared Disk）或不共享（Shared Nothing）的方式來共享設定與資料。它啟動一個行程來接受前端的Request，而讓前端視這一群應用伺服器為一個整體，而不需要知道要將Request轉到哪個應用伺服器。目前很多應用伺服器都是使用這種叢集技術。

(四)選擇隱性中介軟體

在傳統的分散式物件程式設計中（如CORBA、DCOM、RMI等傳統分散式物件技術），可自行撰寫呼叫中介軟體API的程式，例如以下為一個「轉帳交易」可以將金額從一個帳戶轉到另一個帳戶的銀行帳號分散式物件。

```
Transfer（Account account1, Account account2, long amount）
{
//1：呼叫中介軟體API執行安全檢查
//2：呼叫中介軟體API啟動交易
//3：呼叫中介軟體API從資料庫中載入記錄
//4：從一個帳戶一扣除金額，然後將金額加到帳戶二
//5：呼叫中介軟體API將記錄存入資料庫中
//6：呼叫中介軟體API結束交易
}
```

這種需要撰寫程式碼，透過API來存取中介軟體軟體元件來完成企業邏輯的方式，稱為顯性中介軟體。不過這種方

式仍有以下缺點：（1）程式難以撰寫－程式碼過多，只執行轉帳交易卻撰寫一堆程式。（2）程式難以維護－如果要修改與中介軟體的互動行為，必須重新撰寫程式。

而新一代的以元件為主的分散式物件技術（如EJB、CORBA元件模型、Microsoft .NET等），將複雜的中介軟體放置到企業應用伺服器中不需要撰寫程式呼叫中介軟體API。其所需要呼叫僅是含有企業邏輯的分散式物件方法，所以程式碼只須一行。

```
Transfer（Account account1, Account account2, long amount）
{
//1：從一個帳戶一扣除金額，然後將金額加到帳戶二
}
```

主要不同的做法，在於它是利用一個描述檔來宣告分散式物件所需的中介軟體服務，這種描述檔可以是一般的文字檔。例如，可以宣告需要交易服務、安全檢查服務等。透過中介軟體廠商所提供的的命令列工具，將描述檔當作輸入檔而產生呼叫要求攔截元件（Request Interception）的物件。要求攔截元件可以攔截客戶端的要求，而執行分散式物件所需的中介軟體服務（交易服務、安全檢查服務等），並將呼叫委派給分散式物件來處理，這種做法稱為隱性中介軟體。這種方式有以下優點：（1）程式容易撰寫－不需要撰寫呼叫中介軟體API的程式碼，只要在描述檔宣告即可，要求攔

截元件可以通透性的方式提供所需的中介軟體邏輯。因此，重心不需要放在中介軟體上，只須專注在應用程式的企業邏輯上。（2）程式容易維護－利用這種方式可以將企業邏輯與中介軟體劃分的相當清楚，且容易維護，撰寫的程式碼少又簡單。

伍、參考文獻

[1]簡進興，“銀行連線系統端末應用軟體與硬體之配合”，臺灣土地銀行八十一年度出國研究報告，民國八十一年十二月。

[2]張賜安，“運用分散式物件環境於銀行資訊系統之研究”，臺灣土地銀行九十一年度研究報告，民國九十一年四月。

[3]王慧珍/柯月惠，“本行端末系統整合之研究”，臺灣土地銀行九十一年度研究報告，民國九十一年四月。

[4]小泉　修 著/周明憲 譯，“分散式物件新技術圖解入門”，博碩文化，二〇〇三年二月。

[5]蔣智康，“金融服務業新趨勢—網路銀行”，資訊與電腦，民國八十五年，第230期。

[6]許昭仁，“銀行業自動化未來新契機”，資訊與電腦，民國八十五年，194期。

[7]資訊工業策進會，“金融資訊系統整合方案介紹”，民國八十三年。

[8]UNISYS，“美國UNISYS電腦公司簡報”，民國九十一年十二月。

[9]ALLTEL，“美國ALLTEL電腦公司簡報”，民國九十一年十二月。

[10]和平整合資訊，“和平整合資訊公司簡報”，民國九十一年四月。

[11]鼎盛電腦公司，“鼎盛電腦公司簡報”，民國九十一年四月。

[12]JP Morgentha, "Enterprise Applications Integration with XML and Java", 2000/07/31。

[13]David S. Linthicum, "Enterprise Application Integration", Addison-Wesley Longman, 1999/11/12。

[14]David S. Linthicum, "B2B Application Integration: e-Businesse-Enable Your Enterprise" , Addison-Wesley Longman, 2000/12/15 。

[15]Ed Roman, Scott W.Ambler, Tyler Jewll , "Matsering Enterprise JavaBeans , 2nd Edition", The Middleware Company, 2002 。

[16]IBM, EAI Journal, http://www.eaijournal.com/ 。