行政院及所屬各機關出國報告

（出國類別：實習）

參加 WEB 資料庫應用技術研習

（2003 Borland 研討會及 STARWEST 2003 研討會）

服　務　機　關　：中國石油公司資訊處

出 國 人：　職　　稱　：軟體工程師

姓　　名　：陳偉政

出 國 地 區：美　國

出 國 期 間：92 年 10 月 28 日 - 92 年 11 月 7 日

報 告 日 期：93 年 2 月 5 日

系統識別號:C09204870

公 務 出 國 報 告 提 要

頁數: 30　含附件: 否

報告名稱:

參加WEB資料庫應用技術研習

主辦機關:

中國石油股份有限公司

聯絡人／電話:

葉宇容／87258422

出國人員:

陳偉政　中國石油股份有限公司　資訊處　軟體工程師

出國類別: 實習

出國地區: 美國

出國期間: 民國 92 年 10 月 28 日 -民國 92 年 11 月 07 日

報告日期: 民國 93 年 02 月 05 日

分類號/目: G0／綜合（各類工程）　G0／綜合（各類工程）

關鍵詞: Web Services,XML,軟體測試

內容摘要: 在使用網際網路已成爲風潮下，各企業無不思考如下何將企業所提供之服務網際網路化，企業間異質平台或資料庫造成服務網路化之最大阻礙，Web Services正巧興起，成爲不同資料庫或應用程式整合之解決方案，而且已爲各大軟體廠商所接受推崇爲解決企業內EAI（Enterprise Application Intergration）或企業間B2B之最佳選擇，Web Services被預測將是未來電子商務交易所採用的一種新技術與應用標準。本次2003 Borland研討會，Borland公司邀集相關專家針對Web Services相關技術、應用及安全考量，進行了精闢之介紹。STARWEST 2003(Software Testing Analysis & Revierw)是由Software Quality Engineering公司主辦之軟體系統測試研討年會，研討會中邀請了軟體廠商專家（From Micrisoft、IBM、BMC...）、學者（Rex Black、Hans Buwalda...）、實務使用者（From Capital One Financial...）針對軟體測試最新技術及策略進行介紹。

本文電子檔已上傳至出國報告資訊網

摘　要

在使用網際網路已成為風潮下，各企業無不思考如下何將企業所提供之服務網際網路化，企業間異質平台或資料庫造成服務網路化之最大阻礙，Web Services 正巧興起，成為不同資料庫或應用程式整合之解決方案，而且已為各大軟體廠商所接受推崇為解決企業內 EAI（Enterprise Application Intergration）或企業間 B2B 之最佳選擇，Web Services 被預測將是未來電子商務交易所採用的一種新技術與應用標準。本次 2003 Borland 研討會，即由 Borland 公司邀集相關專家針對 Web Services 相關技術、應用及安全考量，進行精闢之介紹。

STARWEST 2003(Software Testing Analysis & Revierw)是由 Software Quality Engineering 公司主辦之軟體系統測試研討年會，研討會中邀請了軟體廠商專家（From Micrisoft、IBM、BMC...）、學者（Rex Black、Hans Buwalda...）、實務使用者（From Capital One Financial...）針對軟體測試最新技術及策略進行介紹。研討會並另闢專區進行測試工具相關軟體廠商展示會，讓參與學員不僅於會中習得軟體測試技術理論，並可於展示會中實際由參展軟體工具驗證測試對系統穩定度及效能之影響。

目　錄

摘　　要	2
壹、出國目的	4
貳、出國行程	4
參、心　　得	5
一、參加 2003 Borland Conference 研討會	5
（一）服務導向之 Web Services	6
（二）Web Services 帶來的便利	6
（三）Web Services 的執行模式	6
（四）Web Services 相關的角色及動作	8
（五）Web Services 之應用方式	9
（六）分散式系統架構與 Web Services	11
（七）Web Services 應用程式管理原則	12
（八）Web Services 基礎交換格式標準-XML	16
二、參加 STARWEST 2003 研討會	20
（一）軟體測試觀念	20
（二）反覆式測試流程	21
（三）擬定測試計畫	23
（四）軟體測試技術	24
（五）軟體測試方法	25
肆、建　　議	33

壹、出國目的

本公司企業資源規劃系統、全球資訊網、內部企業網站、經營資訊系統、電子行政系統等，已相當程度地應用了 Web 資料庫技術，唯隨著電腦軟硬體技術之進步，欲規劃開發各種具備更高效能、更富彈性及更可靠之資料庫應用系統，則必須對現今系統效能與安全之測試技術及 Web 資料庫應用技術有更深入之瞭解，本次奉派參加 Borland 公司主辦之 2003 Borland Conference 研討會及 Software Quality Engineering 公司主辦之 STARWEST 2003 研討會，期能透過軟體測試技術，提昇公司各應用系統之效能與安全程度，並對最新 Web Services 應用程式整合平台架構技術有所瞭解，藉以提昇現有資料庫應用程式整合運作效能，建構企業新世代 Web 應用系統。

貳、出國行程

起迄日期	天數	到達地點	詳細工作內容
92.10.28	1	台北→聖荷西	起　程
10.29～10.31	3	聖荷西	參加 Software Quality Engineering 公司主辦之 STARWEST 2003 (Software Testing Analysis & Revierw)研討會
11.01～11.05	5	聖荷西	參加 Borland 公司主辦之 2003 Borland Conference 研討會
11.06～11.07	2	聖荷西→台北	返　程

參、心得

一、參加 2003 Borland Conference 研討會--研習最新 Web Services 應用程式服務架構

網際網路發展迄今，可謂是一波未平一波又起，網際網路的第一波是用 EMAIL 將人和人連接起來，讓人與人之間的溝通和互動有了一個新的管道，網際網路的第二波是用 BROWSER 將人和資訊連接起來，讓信息與資訊的流通有了一個新的管道，而網際網路的第三波則預期是將機器和機器連接起來，讓系統與系統、程式與程式之間有一個在網路上互通和互動的新管道，此第三波即是本次研討會討論之重點—Web Services。

在使用網際網路已成為風潮，各企業無不思考如下何將企業所提供之服務網際網路化，Web Services 之概念於是成型，Web Services 可提高企業在現今變動性極高的市場當中回應的速度，亦可加速市場之拓展，知名國際調查機構 Gartner 已提出預測，Web Services 將於 2004 年主導企業新應用的建置，各軟體廠商亦十分支持此 Web Services 架構概念，唯所提供的相關支援亦有不同的名稱（微軟稱之為.NET，在 Sun 稱之為 ONE）。

Web Services 被預測將是未來電子商務交易所採用的一種新技術與應用，有關 Web Services 的概念，可視為當前網際網路 BROWSER-HTTP 之發揚光大，我們知道現在的網路上早已有許許多多的 Services，例如我們可以透過 BROWSER 進行內容搜尋，查看最新新聞、進行股市下單等，這些 Services 多數都受到一個限制－那就是使用者必須使用 BROWSER 才能享用這些網路服務，Web Services 時代的到來，使用者將無須受限使用 BROWSER 去享用這些服務。

（一）服務導向之 Web Services

服務導向的應用程式架構(Services Oriented Application Architecture，SOA)，就是將個別程式或系統視為一個一個的服務，再利用共同約定之的服務介面進行溝通的一種應用程式架構。

- 在 SOA 架構下，每一個服務皆具有完整的資料結構與程式邏輯，並透過定義好的訊息格式進行溝通，其核心特性之一即是將服務的內容與存取介面分開，所有程式邏輯及服務內容全部包裹在服務內部，僅透過標準的介面與外部作溝通。
- 在 SOA 架構下，用戶取得之軟體服務將可能來自於不同的應用程式系統、不同的平台、甚至可能來自於 Internet。
- Web Services 即是 SOA 一實例，其運用 HTTP、XML、UDDI、SOAP 和 WSDL 等 Internet 上的標準技術規範，實現跨平台、跨語言、跨 Internet 的整合目的。
- 在 Web Services 的觀念中，Internet 本身就被視為是一個作業平台，而應用程式系統則是架構在整個 Internet 之上。

（二）Web Services 帶來的便利

- Web Services 擺脫了以往硬體的限制，將資料存放在 Internet 中，再透過各種平台設備，包 Desktop、Notebook、Mobile 或 PDA 來達到即時傳遞資訊的。
- Web Services 能讓程式設計人員輕鬆地在資料或應用程式間建立連結並完成交易。
- Web Services 架構跳脫出目前網頁應用的限制，讓使用者可以隨心隨欲地透過任何裝置，在任何時間、任何地點自由地存取資料或將這些資訊整合於應用程式中。

（三）Web Services 的執行模式

在 Web Services 之新概念中，「Services」，不再單指一般網路上所提供的下單、訂票、查詢、購物...等服務，而是一種元件的服務，此元件服務將可分享給他人用來建構更強大服務的元件。若是要建構一個完整的 Web Services 平台，它使用的標準及網路協定除了 XML 外，必需再加上 SOAP、WSDL 與 UDDI，亦即我們可將 Web Services 視為等於 XML + SOAP + WSDL + UDDI。

- SOAP（Simple Object Access Protocol）

SOAP 是一個標準協定，SOAP 運用 XML，提供應用程式與網路交換資料的機制。也就是說，SOAP 可藉由 HTTP 與 XML，呼叫遠端程式(Remote Procedure Call)予以執行。SOAP 以 XML 格式發送訊息有很多好處，尤其是其跨平台的特性，因為這可將使其軟體使用範疇更加寬廣。

- UDDI (Universal Description, Discovery and Integration Service)：

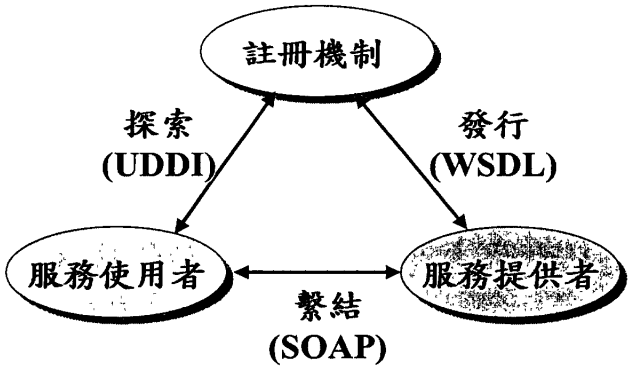
UDDI 是在 WS-I 組織所定義的登錄服務技術，在 UDDI 目錄提供一個服務登錄的集中位置，提供用戶端在網路上動態尋找其他 Web Services 的機制，透過 UDDI 的服務，使用者可以尋找公司內部或其他公司所提供的 Web Services，或是當用戶端程式找不到原本繫結的 Web Services 時，可到此取得最新的服務狀態。我們可以把它想像成商業應用程式上的 DNS 服務，而 UDDI 的註冊分為兩種：一種是要發佈服務的客戶，另一種則是想要取得特定服務的客戶。

- WSDL（Web Services Description Language）

WSDL 是一種 Web 服務定義語言，WSDL 為服務提供者提供描述在不同協定或編碼方式上呼叫 Web Service 的方法，簡單的說，WSDL 就是用來描述一個 Web Services 能做什麼？位置在哪？如何呼叫？

（四）Web Services 相關的角色及動作

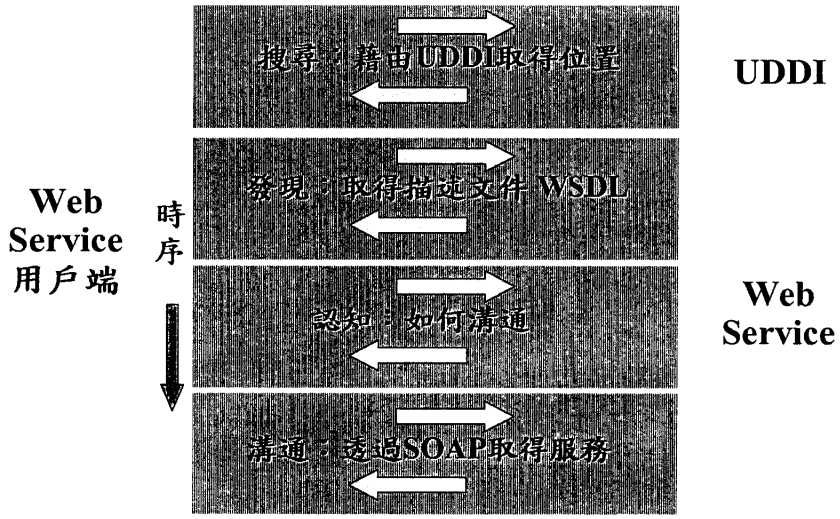
根據 Web Services 的架構，Web Services 是由三種不同的角色共同完成，分別為服務的提供者、服務的使用者以及註冊機制。首先，服務的提供者必須將本身的 Services 以 WSDL 的格式描述清楚，建立 Services 的 WSDL 說明文件。其次，服務的提供者必須將該文件以及該 Service 的存取端點登錄於註冊機制。爾後，服務的使用者便可從註冊機制中找到該 Service 的 WSDL 文件及 Service 的存取端點。接下來服務使用者端的程式只需依照 WSDL 的描述建立存取介面，並將訊息透過 SOAP 通訊協定傳送到服務提供者端的存取介面即可。服務提供者端的程式將自動依據使用者的請求，回覆適當的訊息。



[Web Service 組成之三種角色]

Web Services 應用程式的執行模式分為四個動作，包括搜尋、發現、認知、溝通。

- 搜尋：用戶端程式找不到原本繫結的 Web Services 時，透過 UDDI 取得公司內部或其他公司所提供的 Web Services，若用戶端已知道服務描述的位置，便可略過這一個步驟。
- 發現：下一個步驟則依據所取得的資料，找到服務的窗口，取得服務的描述文件 (WSDL)。
- 認知：透過 WSDL 文件了解其服務描述，用戶端知道如何與 Web Services 溝通才能使用其服務。
- 溝通：最後則是透過標準的通訊協定 SOAP 作為溝通管道，與 Web Services 進行溝通，並使用所提供之服務。



[Web Services 執行之四個動作]

（五）Web Services 之應用方式

■ 利用 Web Services 進行企業內應用程式整合（Enterprise Application Intergration，EAI）

多數企業內部各部門因不同資訊需求而各自採用不同之軟體系統，且各系統間各有不同之溝通模式，在 ERP 風行且強調資源共享之今日，企業內應用程式整合一直是企業資訊人員欲解決之難題。Web Services 替企業找尋到內部應用程式整合之出路，透過 SOA 的設計方法，將企業應用程式的功能與資料轉換或包裝為 Web Services，即可透過標準的 Web Services 技術平台進行溝通，達到資源共享、擴充原有應用程式功能或延伸舊有應用程式生命週期之目的。

■ 利用 Web Services 進行 B2B 交易

在 B2B 市場中 Web Services 較傳統 EDI 具有多項優勢，詳如下表。利用 Web Services 再加上流程管理軟體，可以讓企業與合作夥伴間迅速建立起具彈性之作業流程，進而提升效率並降低企業經營的成本。

EDI	Web Services
舊有共通 B2B 電子交換格式	全新 B2B 電子交換格式
需特殊交換平台與資料格式	不需要特定平台，Internet 即為平台
程式的複雜性高	程式複雜性低
建置成本高	建置成本低

■ 利用 Web Services 提供或取得軟體服務

透過 Web Services，企業應用程式可不需全部由企業內部自行開發，企業可接受由外部所提供之軟體服務，進而簡化系統開發流程。當然企業也可改變角色，對外提供軟體服務，例如隨選服務(Services-on-Demand)或軟體積木等。

（六）分散式系統架構與 Web Services

- 企業系統架構演變三部曲：
  - 第一部曲：大型主機與終端機的組合
  - 第二部曲：主從架構或三層式架構為主
  - 第三部曲：分散式運算與網際網路存取架構

- 分散式應用程式架構之優點
  - 企業可依需求逐步擴充系統
  - 系統功能模組可分階段建置或升級
  - 企業可將資料庫或資訊系統分布在不同的地方
  - 有助於將商業邏輯集中管理，並取得較佳的安全機制
  - 伺服器與資料庫資源的充分利用
  - 多樣化的用戶端的設計模式

分散式應用程式架構雖然有其優點，然而目前各企業在處理分散式應用程式架構時，皆運用自己獨特的一套技術，如下表不同分散式應用程式有不同的訊息格式或探索機制，使得系統整合極為不易，Web Services 就是在這個大家都說 Yes，且願意遵守其標準的環境下誕生。

	CORBA	DCOM	Java RMI
通訊協定	IIOP	RPC	IIOP or JRMP
訊息格式	CDR	NDR	Java Ser. Format
描述語言	OMG IDL	IDL	Java
探索機制	Naming Service	Windows Registry	RMI Registry or JNDI

[目前各軟體廠商的分散式應用程式架構皆不相同，
造成企業應用程式整合時之困難]

企業在考慮建置 Web Services 時應考慮如下之技術與標準：

- 選定能管理 XML 之資料庫
- 選定能支援 Web Services 之應用伺服器
- 選定 Web Services 應用程式開發工具
- 選定支援 XML 及 SOAP 之 Enterprise Portal 架構
- 選定 EAI Middleware
- 選定可監控 Web Services 運作之管理工具

（七）Web Services 應用程式管理原則

程式開發人員在設計 Web Services 相關應用程式時應考慮如下三項管理原則。

- 安全性管理原則
- 通訊管理原則
- 設計通訊原則

一般安全性指導方針

當程式開發人員在考慮設計安全性原則時，首先須注意下列指導方針：

- 儘可能使用經過測試且已業界或廠商證實的安全性系統，不要建置自訂的解決方案。
- 使用最小權限原則，永遠分派最小但足夠任務進行之權限給使用者或相關人員。
- 減少與外界接觸表面區域，透過應用程式所顯露的元件和資料越多，則系統風險就越大。
- 永遠假設外部系統是不安全的，並對外部輸入資料存以戒心。
- 具備預設安全組態，以利系統遭受異常干擾時，可迅速重新啟動。
- 不可忽略企業內部網路使用者對系統可能造成之風險，在嚴謹地預防其他企業或來自網際網路的惡意攻擊時，不可忽略內部用者往往是病毒等風險之最常來源。
- 避免模糊之安全措施，模糊之安全措施往往比不施行還危險，例如將資料檔案存於意想不到之檔案路徑等。
- 請遵循 STRIDE 原則，STRIDE 代表 Spoofing (欺騙)、Tampering (竄改)、Repudiability (否認)、Information disclosure (洩漏資訊)、Denial of Service (拒絕服務) 以及 Elevation of privileges (權限提高)。這些是安全性漏洞的種類，必須保護系統免於發生這些情形。

安全性管理原則

- 驗證與授權機制是否完備：確定使用者確有權限使用資源 (例如檔案或資料庫表格) ，且針對不同使用者應有不同授權原則決定其可存取之資源。
- 通訊通道是否安全：確保應用程式執行時，各不同層面之間的通訊是安全的，以避免資料在傳輸中或儲存於佇列時，被他人惡意

入侵修改或竊取，在通道安全機制上，可使用 SSL (Secure Sockets Layer)、IPSec、VPN(Virtial Private Network)或通道加密保障整個通道的安全。

- 是否妥善管理設定檔：使用者設定檔內包含應用程式用於自訂其行為的使用者資訊，其中重要之資訊如個人基本資料或信用卡詳細資料皆置於設定檔內，由於設定檔的資訊可以由 Principal 物件公開為集合元件，故設定檔應妥為保存，甚至可考慮進行加密處理。
- 是否定期進行稽核：就如同會計稽核，在安全考量下，需定時進行使用者行為及系統運作行為之稽核，確保使用者依照規劃之行為模式進行作業，而系統也先前之資源規劃進行運作。

操作管理原則

- 是否進行例外狀況管理：所有應用程式都應該針對不同之例外狀況建置相關處理機制，以排除例外狀況並讓系統持續運作，或是但向使用者顯示有意義的訊息，並提出例外狀況相關資訊，以方便進行偵錯。
- 是否進行監控：操作人員應使用工具隨時監視應用程式的運作情況及其與服務層次合約 (SLA) 的配合情況，適時進行規模與容量調整。
- 是否使用中繼資料：若要讓應用程式在變更 Run-Time 條件方面更具彈性，就需要在特定地方設計應用程式以使用中繼資料，讓應用程式更容易維護，並且使應用程式容易變更。在應用程式中使用中繼資料的主要時機有兩個：

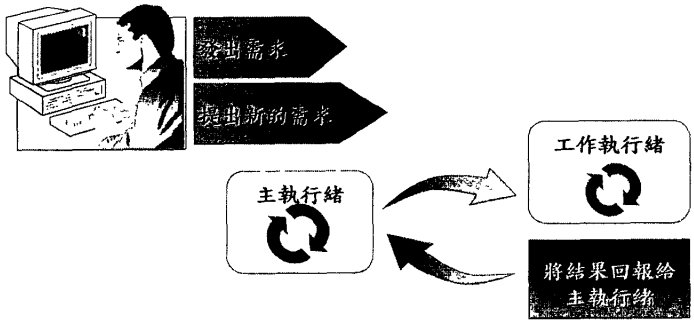
  設計階段：對於經常重複的模式，如使用特定資料庫的資訊以產生程式碼、預存程序或甚至是使用者介面元件。此時使用中繼資料將可節省開發時間，並減少開發人員間之溝通。

執行階段：在建立元件之間的關係或是處理可預測模式時，可在應用程式中運用中繼資料。不過，需注意在 Run Time 使用中繼資料通常會耗費效能並易造成安全性漏洞。

- 是否正確設定安全原則所需參數：設定資料安全原則所需參數是指可以修改原則 (安全性、操作管理和通訊) 行為的設定值，應用程式需要妥善設定參數才能讓系統安全機制正常運作，此設定值可以很容易地使用應用程式中的 ConfigurationSettings 類別來存取。
- 確認服務位置：當使用者呼叫遠端服務時，必須判斷可以處理要求的 .NET 物件和外部服務位於何處。

設計通訊原則

- 採用同步或非同步通訊性：一般以訊息為基礎的通訊是屬非同步通訊。然而，以訊息為基礎的通訊也可以封裝在同步通訊的應用程式模型中，在同步通訊模式下，用戶端會等候回應的訊息，應用程式開發人員反倒可以避免處理非同步通訊模式下程式設計的複雜性，不過，Web Services 環境仍建議以非同步通訊方式進行程式撰寫，如此將可減少使用者等待回應時間。



[在非同步通訊下，應用程式將部份工作交由其他的執行緒(thread)執行，而主執行緒繼續執行原工作，應用程式的使用者無須等待需求回應，可以立即提出新需求]

- 通訊格式和通訊協定：需考慮是否將企業應用程式元件或資料庫提供外部人員使用？是否有 B2B 企業間商務需求？若是，建議依照 Web Services 標準使用 XML、 SOAP 等格式，並使用標準 Internet TCP/IP 網路通訊協定。

（八）Web Services 基礎交換格式標準-XML

XML 是 Extensible Markup Language 的簡稱，中文名為可延伸性標籤語言，它在外表上很像 HTML，許多人以為 XML 是 HTML 的延伸或是新的版本。然而 XML 和 HTML 這兩者之間最大的不同在於 HTML 的目的是給人看的，而 XML 卻是給機器讀的，亦即，XML 並不是用來編排內容，而是用來描述資料，XML 並沒有如 HTML 一般預設之標籤，使用者必須自行定義描述資料所需之標籤，廣義來看，HTML 可視為 XML 的一種應用。

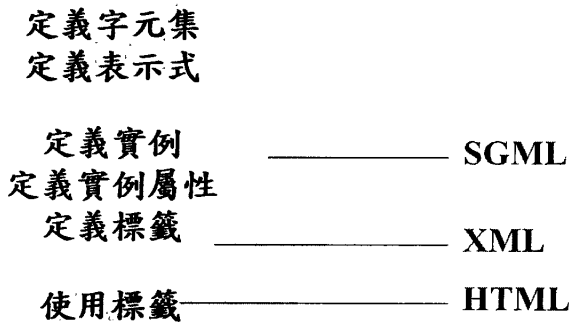
標籤語言

- 由一些特殊字碼或控制標籤所組成
- 它們單獨存在時並無任何的意義

- 需要特殊的軟體經由一定的規則解讀
- 標籤語言可以使文件更具結構化，這樣的結構化使得應用程式能夠便於管理、解讀與運用文件中的資料
- 標籤語言可分為「特定標籤語言」與「一般化標籤語言」

XML 和 HTML 之比較

- HTML 的主要目的是形容一個網路上文件的格式，讓瀏覽器按照文件之中的標誌去展示文件，而 XML 的主要目的用來形容結構化的資料，讓程式系統能夠閱讀。
- HTML 是一種表達的技術，並不能夠描述 HTML 標籤中所包含資料的意義 HTML 標記的設計未將內容結構與呈現劃分，導致網站資料的傳輸與交換難以再利用，XML 則將內容結構與分開呈現。
- HTML 中對標籤在使用上的規則比較鬆散，然而 XML 對標籤的使用方式規定得比較嚴謹



[HTML、XML 及 SGML 關係圖]

XML 是從 SGML（Standard Generalized Markup Language）推演出來的，SGML 的缺點在於複雜性高，導致在制訂 DTD（Data Type Definiation）文件時非常耗時，XML 則擷取了 SGML 中文件結構的核心部份，繼承 SGML 自訂標籤之優點，並刪除 SGML 複雜之部分。

SGML 之特色

- 解決資料交換問題而發展的
- 為封閉式環境所設計
- 過於複雜，並不適合在網際網路上使用

XML 之特色

- XML 可於 Internet 上運作
- XML 提供一種中立、標準的交換格式
- XML 具有自我描述資料能力
- XML 之文件淺顯易懂
- XML 用來描述文件的內容與結構，而不會去定義如何顯示或運用這些文件的內容
- XML 之標籤可依重要性最小化
- 撰寫處理 XML 文件的應用程式十分簡單
- XML 可支援多樣性之應用程式，可讓許多不同的軟體解讀文件
- 解讀 XML 後的資料要如何運用，端視各種應用程式個別需求

XML 之優點

- 資訊完整性高：讓搜尋更快速、更有效率
- 可降低重複資料的傳輸，強化網路資源的使用效率
- 可擴充性高：XML 允許使用者自行定義標籤。
- 共通性高：XML 可跨異質作業平台，讓應用系統可以一致方式存取所有資料。

- 安全性高：具有資料提供者的識別與驗證功能

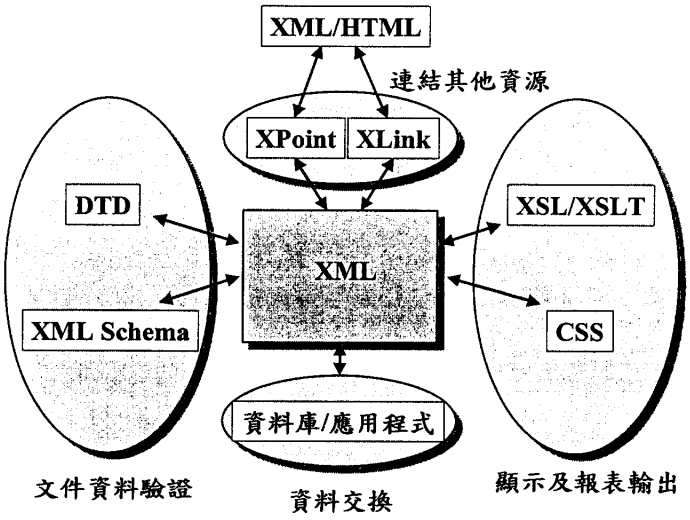
XML 之應用領域

- 跨平台資訊交換：XML 本身為文字，無論任何平台都可應用，此外 XML 將文件內容與呈現方式分離，有助於資訊於不同平台的再利用。
- 知識管理與資訊搜尋：發展知識管理系統之關鍵在於如何有效地建立知識之模組，並迅速加以擷取，XML 具有儲存並管理文件架構之能力，且由於其描述資料能力，非常適合一般搜尋引擎進行資訊搜尋。
- 企業間電子商務：隱藏在當前網際網路電子商務熱潮下的是資料交換問題，由於各個企業在不同的作業平台下使用不同之程式語言開發資訊系統，並存取不同之資料庫資料，造成企業間資料交換實在相當的困難。傳統 EDI 雖然實現了資料的一致性，卻嚴重缺乏可延伸性。

XML 的衍生相關的技術及規格

- Xlink：定義如何在 XML 中加入超鏈結
- XPoint 及 XFragments：指向 XML 文件中某份資料的語法
- CSS：可以同時應用在 XML 及 HTML 的樣式表語言，
- XSL：以 XSLT 為基礎的進階樣式語言 針對 XML 文件所建立的格式化語言，負責 XML 呈現部分
- DOM：存取 XML 資料的函式，提供對 XML 文件處理標準界面
- XML Schema ：定義 XML 的資料格式，作為文件定義與資料範圍限定，XML 配合 XML Schema 提供對內容的精確宣告，可以讓使用者在對各種不同平台進行搜尋時，有辦法針對其中的語義加以查詢

□ XML Namespace：定義命名空間



[XML 衍生相關技術關連圖]

二、參加 STARWEST 2003 研討會--研習軟體測試技術

為何系統開發完成後，使用者總是抱怨系統經常當機，或抱怨系統執行太慢，為何會如此，這都是測試工作沒有被正確之規劃並加以確實執行。

（一）軟體測試觀念

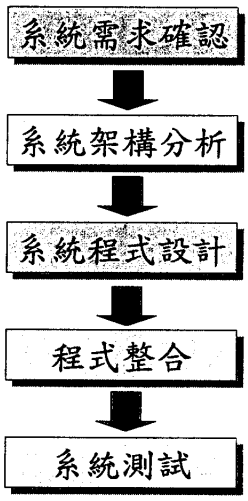
在實際進入軟體測試之重要性及方法前，先釐清一些有關測試之觀念：

□ 我們藉由軟體測試來確保軟體品質。

- 測試是一連串 Verification and Validation 的活動，前者重視過程，後者重視結果
- 測試的目的是為了找出程式的錯誤，而不是證明程式是對的
- 好的測試案例（Test Case），是指有較高可能發現錯誤的測試案例。
- 軟體開發不同階段時應採用不同的測試技術
- 成功的測試是發現程式的錯誤，若無發現錯誤，則是失敗的測試
- 測試工作應為獨立團隊完成，絕不能由開發人員自行進行

（二）反覆式測試流程（Rational Unified Process，RUP）

資訊部門進行系統開發時，總是採用如下「瀑布式（Waterfall）」流程進行：



也就是把開發過程分為「系統需求確認」、「系統架構分析」、「系統程式設計」、「程式整合」、「系統測試」等階段，一個接一個依序進行，此種流程之盲點在於總是「把測試排在最後進行」，「把測試排在最後進行」會為整體系統開發帶來麼不利之影響，概述如下：

■ 測試人員無法明確瞭解系統架構，僅能進行最終之功能測試

在一個具規模之系統開發，除了系統開發人員外，尚有所謂之測試人員，依上面之流程規劃，測試人員總是在程式整合後才進行系統測試，往往無法對系統架構有明確之瞭解，只能針對主要功能進行概略之測試工作，測試工作可謂是粗糙地被進行，往後系統運作執行時必定是問題百出，而往後針對層出不窮之問題進行之補破洞式 debugging，其人員付出將是一經過完整正確進行之測試所需付出之數倍。

■ 測試人員經常無足夠時間進行測試

「把測試排在最後進行」所導致的另一個缺點是「時間不夠」。由於多數系統開發流程分配時間時，總是把大部分時間分配給「系統程式設計」、「程式整合」兩階段，「系統測試」總是被分配到最後剩餘且寥寥無幾之時間，若前面階段稍有閃失造成延遲，則會壓縮「系統測試」時間甚至犧牲掉「系統測試」這個項目。

■ 系統成功完成之風險係數提高

系統開發時往往隱藏有許多變數及風險，這些變數及風險仰賴確實之測試即時加以發覺並修正改進，「把測試排在最後進行」將造成變數及風險在開發設計過程中未被發掘並不斷累加，終至無法收拾或失敗之結果，就算可以補救，也必須付出相當大的代價。

由上可知傳統系統開發之流程「單向瀑布式」在測試項目之排序上需做一修正，應摒除「把測試排在最後進行」而改採「反覆式」的 RUP（Rational Unified Process）測試流程。

RUP 重要概念

■ 測試人員從「一開始」就進入整個開發工作

測試人員「系統需求確認」階段起，即參與擬定相對的測試計畫，進而對系統架構有明確之瞭解，測試工作隨著各開發階段持續地被追蹤與進行細部修改，並逐步接近系統功能整合測試。於是，測試人員的工作是平均分佈到整個開發流程中，而不是集中在最後階段。

■ 一有產品，即行測試

當任一子系統（或模組）在被開發出來時，測試工作就會即刻配合進行，檢驗其是否吻合使用者需求與各項設定條件測試，如此像前述之變數及風險，將在最初時即加以排除。

（三）擬定測試計畫

系統開發在不同階段進行測試前，皆需訂定測試計畫，測試計畫之目的，無非是讓使用者、開發人員與測試人員之間的能充分溝通，讓軟體測試能依系統化的方式來進行，軟體測試更加順利的進行，最後由測試計畫之逐項執行結果確認開發出之系統（或模組）是否符合使用者及開發人員之需求並符合各項預定條件。

計畫書可區分成如下三類：單一文件測試計畫書（STP）、主要方針測試計畫書（MTP）、詳細運作測試計畫書（DTP）。

■ STP 單一文件測試計畫書（Single Test Plan）

STP 就是將測試計劃所有的議題，包括軟體測試所需之常數與變數，集中撰寫在同一份的文件上。

■ MTP 主要方針測試計畫書（Master Test Plan）

MTP 是將測試分成為不同的進行階段，對於每個階段規劃出概略的測試方針。

■ DTP 詳細運作測試計畫書（Detail Test Plan）

DTP 則是撰寫各階段的詳細測試計劃，通常一個 MTP 會伴隨好幾個 DTP 一起出現。

另外，IEEE 829標準中建議測試計畫應包括如下16個大綱要項：

項目綱要	解釋
Test plan identifier	測試計劃書的目標
Introduction	基本介紹
Test items	測試的項目
Features to be tested	需要測試的產品功能
Features not to be tested	不需要測試的產品功能
Approach	採用的測試模式
Item pass/fail criteria	項目通過測試的標準
Suspension criteria and resumption requirements	測試中斷與開始的規定
Test deliverables	測試完成所要提交的項目
Testing tasks	測試工作項目
Environmental needs	測試環境的設定
Responsibilities	人員的工作分配
Staffing and training needs	人員的能力與所需要的訓練
Schedule	測試的時程
Risks and contingencies	潛在問題與風險
Approvals	文件的認可

（四）軟體測試技術

軟體測試是軟體品質管理中最實際的行動，軟體測試是有組織性的、步驟性的、以及計劃性的。就測試基本模式而言，軟體測試可分為兩大路線，白箱測試（White Box Testing）與黑箱測試（Black Box Testing）。 這兩種方式的測試方向是不同的，白箱測試是以測試的深度為主，而黑箱測試是以測試的廣度為主。

■ 白箱測試（White-Box Testing）

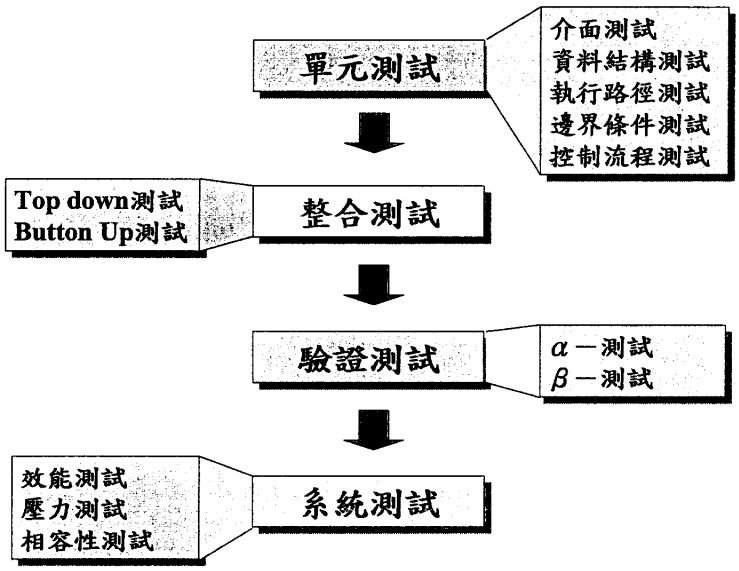
白箱測試也稱為結構性測試（Structural Testing），有時候因為牽涉到內部機密的問題，這種測試路線大都是在公司內部進行很少委外給其他公司或個人。 嚴格來說，白箱測試有兩大層面；資料流程面（Data Flow Coverage）以及控制流程面（Control Flow Coverage），資料流程面就是測試資料在系統的進出入於程式內所經過的流程，控制流程面就是測試程式在執行過程中每個階段的流程。

■ 黑箱測試（Black-Box Testing）

測試人員並不需要對軟體的結構性有足夠深層的瞭解，所進行的測試是著重於軟體的功能面，所以也有人稱之為功能測試。 這樣的測試除了在自己公司內部進行之外，同樣的也可以委外給其他人員或是公司去執行。 為了要控制黑箱測試的執行，測試人員必須要按照測試案例（Test Cases）來逐一進行，所以 Test Cases 設計的好壞就會直接影響到測試結果。

（五）軟體測試方法

軟體測試可以簡單的分為：單元測試→整合測試→驗證測試→系統測試。



[軟體測試方法]

- 單元測試主要考慮個別模組的功能、介面及控制流程正確
- 整合測試主要考慮模組間的功能驗證和架構協調
- 驗證測試說明程式是否符合軟體需求規格
- 系統測試則在驗證此軟體於實際環境中是否可運作正常

單元測試

單元測試必須包含如下子測試：介面測試、資料結構測試、執行路徑測試、邊界條件測試及控制流程測試。

■ 介面測試

介面測試亦即流經模組介面的資料測試，測試人員應檢查區域資料結構以保證臨時儲存的資料在演算執行的所有步驟中能保持一致性，並避免資料不正確地作為參數進入模組或 return 出模組，在此測試中，測試人員應確認：

- 輸入參數的數量是否等於引數的數量
- 參數與引數的屬性是否匹配
- 函數的數值屬性和引數順序是否正確
- 傳送給被調用的引數數量是否等於參數的數量
- 傳送給被調用模組的引數屬性是否等於參數屬性
- 模組中整體變數的定義是否保持一致

當模組執行外部I/O時，測試人員應進行另外之介面測試並確認：

- OPEN/CLOSE 敘述是否正確
- 檔案屬性是否正確
- 檔案在使用前是否被開啟
- 格式宣告與 I/O 敘述是否匹配
- 緩衝區大小與記錄大小是否匹配
- I/0 錯誤是否被處理

■ 資料結構測試

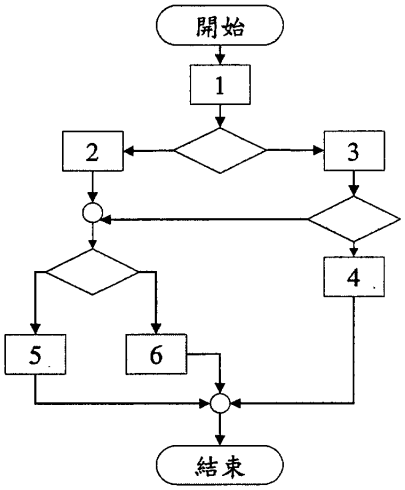
模組的區域資料結構往往是執行錯誤之來源，測試人員應設計測試案例確認模組是否有如下之錯誤：

- 不正確或不一致的 type
- 不正確的變數名稱
- 錯誤的初始化值或預設值
- 下溢，上溢，位址出界
- 不一致的資料型態

■ 執行路徑測試

執行路徑測試即是檢查控制結構所有獨立路徑，期在發現錯誤的計

算、不正確的compare或不合適的控制流程，測試人員必須將模組中所有敘述至少執行一次。如下圖，計有5個Case（1-2-5、1-2-6、1-3-4、1-3-5、1-3-6）需被執行。



[執行路徑測試案例]

■ 控制流程測試

一般程式撰寫時，控制流程的變動經常發生在比較條件之後，亦即比較和控制流程相互緊密結合的，測試人員應針對控制流程中所有可能之比較測試逐一進行測試，如下列敘述：

if（C）then { E1 } else { E2 }

若C 為C1.op.C2，則要測2 個cases。

若C 為(C1.op.C2).op.(C3.op.C4)，則要測4 個cases。

透過控制流程測試，測試人員應仔細發現程式是否有如下之錯誤：

- □ 不同資料型態的比較
- □ 不正確的邏輯運算子和優先級

- 由於精準度的錯誤，使得等號不成立
- 不正確的比較和變數
- 不正確或不存在的迴圈結束
- 不正確地更改迴圈變數

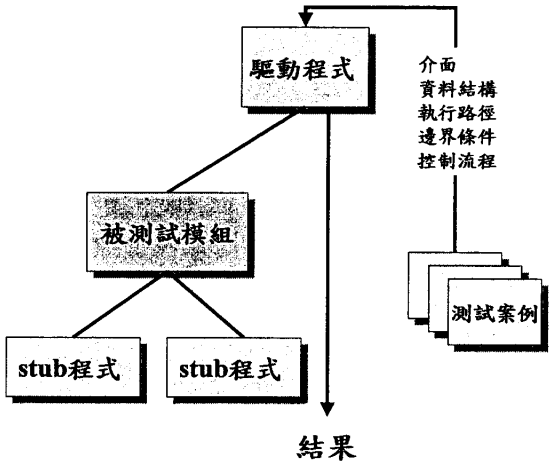
■ 邊界測試

邊界測試是單元測試步驟中最後工作，測試人員應運用控制流程條件中剛好大於、等於或小於最大值或最小值的測試例子來進行所謂邊界測試，因為在此條件數值邊界最容易發生錯誤，另外，錯誤亦經常發生在：（1）在矩陣處理運算時，一個 n 維陣列的第 n 個元素被處理時。（2）在迴圈處理時，發生在具有 i 次迴圈的第 i 次重複被啟動時。

單元測試程序

上面介紹了單元測試所需包含之子測試，接下來說明單元測試程序：

由於模組不是一個獨立程式，欲進行測試時呼叫此模組之程式（上游模組），或此模組欲呼叫之程式（下游模組），可能皆尚未被開發出來，故在此時點，開發人員必須因此須先行模擬上游模組設計出一主程式（此處稱為驅動程式），另為其模擬下游模組設計出一副程式（此處稱為 stub 程式），並如下圖測試環境進行測試程序，此驅動程式及 stub 程式屬程式測試時之額外軟體，主要是配合前述單元測試所需之各項子測試所需。



整合測試

由於較具規模之系統開發，程式開發團隊一定是由多個開發人員組合而成，在開發階段過半，亦即各自完成所配之模組後，就必須面臨系統整合的問題，整合測試即是確認在將模組堆疊後，整體系統運作能達到預期的功能。整合測試大部分用黑箱測試來進行，測試方法又可分為 Top down 及 Bottom up 兩種。

■ Top down 之整合測試

模組按照控制層次由上而下整合，它開始於主控模組(主程式)。主控模組的子模組被按深度優先或廣度優先的方式置入整個結構中。若低階層的模組尚未完成，則須以stub程式代替，供上游模組呼叫，此時以Top down方式測試較佳。

Top down整合測試程序應按照以下步驟執行：

- 主控模組為測試驅動模組，所有子模組當作 stub 模組
- 用實際模組逐一替代 stub 模組
- 每一模組被整合時進行測試
- 隨著每一測試整合結束，額外的 stub 模組被實際模組取代

■ Bottom up 之整合測試

由下而上的整合測試，從程式結構中最低層的模組開始進行構造和測試。由於模組是由下而上整合的，因此對於一個給定層次的子模組所需的處理總是可用的，所以不需要stub模組。若模組為一般用途的utility，且常為其他模組所呼叫時，則以Bottom up方式較佳。

Bottom up 整合測試程序應按照以下步驟執行：

- 低層的模組被合併成執行特定軟體子功能的一群模組
- 撰寫一個驅動程式，協調測試例子的輸入和輸出
- 測試每一組模組
- 刪除驅動模組，在程式結構中向上合併模組群

驗證測試

整合測試的完畢後，代表可能之介面錯誤皆已被查出並修正，軟體已夠資格被組裝成套裝軟體，此時進入軟體測試的驗證測試。驗證測試又稱為先期測試，主用於驗證開發出來的系統是否滿足使用者的需求規格，此使用者之需求規格在專案開發時，應已明確定義在軟體需求規格中。驗證測試一般可區分為 α －測試及 β －測試。

■ α －測試：

- 由使用者在開發者端進行的測試
- 軟體公司以本身的成員擔任使用者，先期進行的測試

■ β －測試：

- 由客戶在使用者端，進行的先期測試
- 裝機測試
- 檢查系統的完整性
- 檢查系統功能性是否受到工作環境的影響

系統測試

系統測試通常稱為 QA Testing，就是整合整體系統運作所應支援的額外相關之人力（如網管人員、硬體人員）及資源（如伺服器、網路設備、IP）所進行之測試，此與上述單元測試或整合測試等不同，單元測試或整合測試旨在發覺系統程式寫作上之錯誤，一個程式邏輯正確且單元測試或整合測試都合格之系統，在面臨真實環境試煉時往往還是問題百出，此時問題可能不在系統程式上，而是所安裝之環境，例如系統所安裝之伺服器記憶體及硬碟是否充足？網路是否過於壅塞導致系統運作異常？這些種種，都是需要其他資訊人員運用其他資源加以確認的。在系統測試進行之前，測試人員必須先行建立一組評量標準之基準。

一般說來系統測試可分為如下三種：

- 效能測試

對於即時系統而言，如只提供所需功能，但沒確認效能的系統是無法被接受的。效能測試在檢測已整合的系統範圍內的軟體於使用時所呈現的狀態及所耗用的資源，例如：CPU 使用率、記憶體使用狀況、所佔網路頻寬等等。

- 壓力測試

測試軟體所能承受的極限程度，在此模擬之測試環境下，大量之資料在瞬間輸入系統，測試系統是否仍能正常運作。壓力測試既然被設計成在非正常情況下檢驗程式，測試人員應一併檢驗系統於資源運用上能夠承擔之極限數值。

- 相容性測試

因為不同的使用者會有不同的作業環境，最常見的即是作業系統不同，此測試就是要確保此系統在不同的環境都可以正常運作。

肆、建議

如心得部分中說明，我們可利用 Web Services 進行企業內應用程式整合（EAI），或利用 Web Services 進行企業間 B2B 交易，甚至藉以提供外界軟體服務，分析這三項應用所提供之服務對象是由內而外的，是由少數之公司同仁而推廣至 Internet 上未知之群眾的，故如何將內部現有之資訊系統或正在開發之系統逐步導向符合 Web Services 這種應用程式標準平台，以達整合簡化之目標，實為資訊同仁當慎思進行之方向。由於本公司資訊系統眾多，似可先擇非關鍵類資訊系統進行應用程式架構轉換，除可降低風險外，亦可藉此培訓相關同仁熟習所需技術。

本公司主機使用 IBM DB2 資料庫，在整體暨相容性考量 Websphere（以 Java 為應用程式平台）應為一不錯之選擇。目前市場上最常見之爭執，即是該用.Net 或是 Java 作為 Web Services 之應用程式平台，依本人淺見，若專案性質屬於中小型的應用程式，則.NET 較為恰當，其優異的 UI 反應速度及與 Windows 的整合度，可以較快速的建構出這類 UI 比重較高的應用程式，Java 因為多年來在 Enterprise Application 領域的經營，其產品的功能性與穩定則應較.Net 為佳，亦即兩者各有所擅，建議公司在人力配置許可之前提下，配合潮流建立 Web Services 作業環境，引進統一之開發工具平台，加速應用程式開發。