行政院國家科學委員會專題研究計畫成果報告

網路醫學資訊系統之可擴充性與容錯機制

Scalability and Fault Tolerance Mechanisms for a Network Medical Information System

計畫編號： NSC88-2219-E-002-006
執行期限： 民國87年08月01日到88年07月31日
主持人：郭斯彥　　台灣大學電機工程學系教授

一、中文摘要(關鍵詞：元件物件模式、分散式物件元件模式、物件導向、可靠度、可用度、可擴充性、容錯機制。)

隨著寬頻網路技術日新月異的發展，醫學資訊系統的使用不再侷限於醫院本身，將醫學資訊系統架構於網路上已是一個不可避免的趨勢。利用分散式元件物件模式(DCOM)，可讓醫學資訊系統的程式碼發展具備物件導向及可重複使用的特性，配合新世代的寬頻網路環境，使得醫學資訊的傳遞更具即時性。

分散式元件物件模式是由微軟公司所提出，它延伸自元件物件模式(COM)，用來支援在各種網路環境上不同電腦間物件的通訊。藉著 DCOM，我們可以將已經在以 COM 為基礎的應用程式、元件、工具或知識輕易的轉移到標準化的分散式計算環境上。

由國外的研究論文中發現，隨著客戶端請求的數目增多，或是在伺服器上的物件數目增多，都會使延遲增長。所以本計畫擬將系統的可擴充性作深入研究，並提出與實現可改進效能的方案。其次，我們也將容錯的機制加以考慮來改進系統的可靠度，並增加可用性。我們將先建立容錯模型，找出處理錯誤的機制，提供一個以分散式元件物件模式為基礎的高可靠網路服務。

未來的網際網路應用將以分散式物件為基礎，工業界目前在這方面的進展非常快速，所以我們學術研究必須在這個熱門的領域投入人力與心力，而這個子計畫以可擴充性及容錯機制為課題，將大幅提高分散式計算環境的效能與可靠度。

英文摘要 (COM, DCOM, object oriented, reliability, availability, scalability, fault tolerance mechanism.)

It is due to the rapid development on broadband network technology that the medical information system is no longer just conformed to a medical center or a hospital itself. It is an inevitable trend to construct the medical information system on top of the Internet. With Distributed Component Object Model (DCOM), we make the source codes of medical information system become object oriented and reusable. Together with the next generation broadband network environment, the medical information spreads widely and will serve in a real-time manner.

DCOM is proposed by Microsoft. It extends Component Object Model (COM) to support communication among objects in different computers on the Internet. Because DCOM is a seamless evolution of COM, we can leverage our existing investment in DCOM-based applications, components, tools, and know-ledge to move into the world of standards-based distributed computing.

It has been shown that the latency increases when the number of requests

from clients or objects on the server increases. Thus we will focus on the scalability of the system, and propose and implement new mechanisms for better system performance. Then we will add fault tolerance mechanisms in the system to improve both the reliability and availability. We'll first setup proper fault model and then find out mechanisms of fault-handling to provide a DCOM-based, highly reliable network service.

It is foreseen that Internet applications will be based on distributed components in the near future. The progress on this topic is very fast in the industry, and we have to keep pace with it in the academia. This project focuses on the topics of scalability and fault tolerance mechanisms. We are sure that it will be of great help in increasing the performance and reliability of a distributed computing environment.

二、計畫緣由與目的

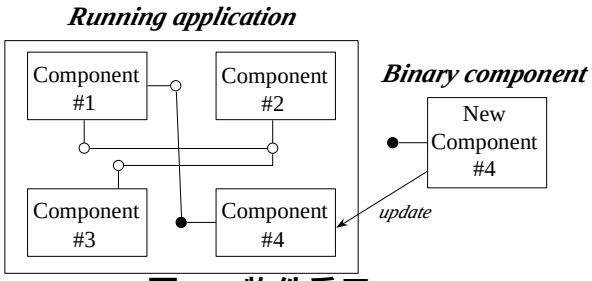
隨著 Java **程式語言和網際網路技術的發展與進步，我們希望將大又複雜的軟體應用程式分開成一系列已經預先建立、容易發展與了解的模組。這個使用元件的技術可以讓軟體的發展更快、更便宜。而微軟公司所發展的** DCOM **技術是一種可以讓這些軟體元件透過網路彼此通訊的技術。它有三個獨特點可以使它成為關鍵性技術。首先** DCOM **是建立在最被廣泛使用的元件技術上。其次** DCOM **可以與各種網路技術如** TCP/IP**、** Java **及** HTTP **相容。最後就是** DCOM **可以在很多作業系統平台上執行，使得在原有系統上的投資不會浪費。**

提供一個全方位的網路醫學資訊系統是項非常具挑戰性的工作，因為它必須是不會停下來的服務，否則將會對醫生或病人造成重大影響，甚至危害生命安全。另外效能的表現對於此系統的成功也是非常重要的，要是不能滿足醫生或病人的期待，他們便會尋求其它方法。再者可擴充性也是必須包括的，因為系統必須能夠隨著醫院及病人之增加，處理大量同時連線的客戶請求。

原有的網路醫學資訊系統最缺乏的是在客戶端的發展，若是沒有改善並加強使用者介面，則與系統間的交談無法得到改進。而改進會受限於原來已存在的系統架構，若不加思索的修改必需完全改寫原有的程式，這就限制網路醫學資訊系統的發展。所以，我們將設計的主要目標設定在如何提供簡單且有效的存取介面，如何將目前最新的多層式分散式技術應用於醫學資訊系統，並提供方法平穩轉移以主機為中心的程式到分散式計算的程式。進而提供可擴充性及容錯機制，使得客戶與伺服器都能感到效能提升及可靠度提高。

我們在新架構中主要是將應用區分為三個方面：使用者服務、系統服務及資料服務。將使用者介面服務與系統服務分開可以使得不同前端技術皆可使用並同時共享及重複使用相同的系統服務如圖一，達到使用者的活動可利用不同的交易組合來存取系統服務。要達到分散式服務，必須將應用程式分開成許多的邏輯元件，利用這些元件的重用及組合來形成新程式或延伸已存在的應用程式。

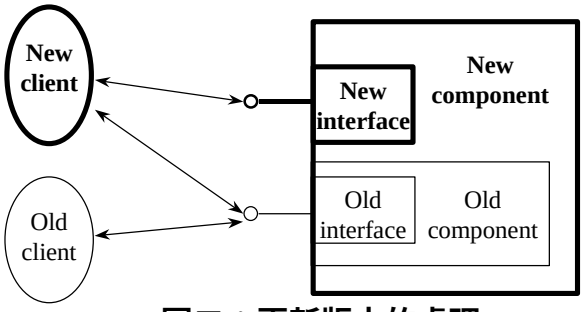
- Run-time software reuse of binary components



圖一：物件重用

COM **的方法是基於下列三個需求。首先，任何一個** IID (Interface ID)**必須是不變的。第二是使用相同的** CLSID (Class ID)**必須支援已存在的介面。最後，任何客戶必須利用詢問介面的方法與伺服器交談。如此才能允許客戶端與伺服器端獨立發展軟體。假設在特定的**

機器上，伺服器軟體在客戶端軟體更新之前更新，因為新的伺服器支援所有舊有的介面，舊客戶仍然可以保有所有的指標並正常工作。當客戶軟體更新時，新客戶將會詢問新的介面 ID 來享受新機能。相反地假如客戶端軟體先更新，則新客戶將試著詢問舊伺服器而得到失敗的結果。當然新的客戶端程式必須被設計成能夠處理這種找不到新的界面時的錯誤，然後新客戶端程式需以舊有的機能來處理這個失誤。這樣才不會使的程式當掉。圖二表示更新版本處理的架構。經由這種機制，伺服端只要讓客戶端知道有哪些界面就可以了，真正的實作階段就可以分別獨立發展而不會發生問題。此外，軟體的更新也可以分別獨立進行，而不會造成問題。



圖二：更新版本的處理

物件導向程式設計之所以受歡迎的一個原因，就是程式設計者可以在不同的專案中共用程式碼。因為重複發展相似的程式碼及演算法是一件浪費時間、金錢的事，所以我們可以看出程式碼的共用有許多好處。可是程式碼的重複使用需要仔細的計畫和考量，否則可能只是一廂情願。

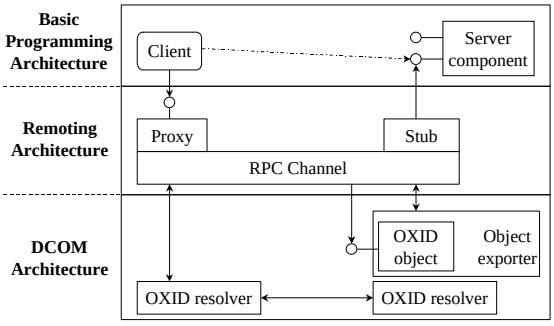
對於大多數形態的程式碼重複使用的狀況來說，還有一個很嚴重的問題，那就是程式碼的原始設計者與想要重複使用程式碼的人必須使用相同的一種程式語言。COM 定義了二進位形式的標準，可以解決這個問題。在撰寫 COM 程式的時候，我們只要定義 COM 的界面並且不可隨意改變此界面，真正實作時所使用的程式語言則不受到限制。達到程式碼可以真正重複使用的目的。

三、研究方法與成果

本子計畫於第一年，成果如下：

我們簡單的比較了 DCOM 與 CORBA 的不同點，這兩個不同分散式物件模型(Distributed Object Model)基本上可以看成是如圖三這樣的三層式的架構，圖中 DCOM Architecture 的部份因為不同平台而有不同的稱呼，也可以稱為 Wire Protocol Architecture 以避免因為不同物件模型上面稱呼的不同。關於詳細的名稱與各層的架構簡圖可以參閱參考文獻[1]。

最上層的部份稱為基本程式設計界面架構，這一層是發展客戶端與伺服端程式的程式設計師所看得到的部份。程式設計師只要知道客戶端與伺服端之間的溝通界面即可。每次需要使用到物件的時候，就經由詢問(Query)界面的方式來獲得某個特定的界面或功能。所以不管對方的實作細節是如何，只要它的界面維持不變，我們都可以由相同的界面獲得我們所需要的服務。所以界面訂定之前就必須要詳細的規劃與考慮。一旦訂定了界面且公佈了，就不可以再隨意更改界面。若需要新的功能我們可以考慮增加支援新的界面，而不可以在舊的界面上面動手腳。在這一層中，用來實作的程式語言不一定要限制是 C++。不過由於 C++的參考資料比較多，所以我們選擇以 C++當作實作的程式語言。



圖三：三層式的 DCOM 架構

在現今的作業系統中，不同行程之間的資料是處在不同的位址空間中。這

些不同的行程可以是在同一部電腦中或者是跨越網路連結的不同電腦中。不管是哪一種方式，我們都需要一種特別的方式或協定才能夠使用在不同行程之間的資料。作業系統通常會提供某種程序間(inter-process)溝通機制，COM 就提供了這樣的一個可以讓不同程序間的資料溝通的機制。

這個機制就是由圖中的中間這一層來達成。它稱為遠端架構層(Remoting Architecture)，它使得不同行程(Processes)之間的界面內的指標或物件的參考(References)變得有意義。程式設計師可以直接將指標傳入界面中當作參數，然後接收端在接收到參數後就可以直接對此指標做的資料做存取(Dereference)，而不需要擔心指標指向的地方是不合法的位址空間。將資料由某一個行程打包好，然後傳遞給另外一個行程並解包的這個動作叫做 Marshaling。

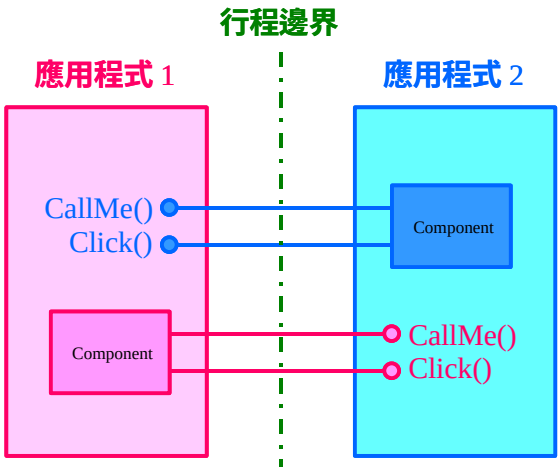
DCOM 使用的是 DCE(Distributed Computing Environment)的遠端程序呼叫(Remote Procedure Call)。為了要達成 Marshaling 的工作，我們必須要實作圖中的 Proxy/Stub 的程式碼部份。撰寫這部份的程式碼需要利用到一種叫做 IDL(Interface Description Language)的語言來撰寫界面。這個語言的語法與 C 語言有點類似。不過由於 IDL 語言沒有太大的一致性，且相關文件也不多。所以這個部份的參考資料不是很足夠。還好對於我們的需求來說，找到的資料已經夠用了。撰寫好了 IDL 程式碼之後，我們可以利用編譯器將這些程式碼編譯成 Proxy/Stub。編譯好的 Proxy/Stub 需要在 Registry 裡面註冊。這樣子 COM 才可以經由系統的協助尋找到需要的 Proxy/Stub 所在的位置。

我們希望以後能對 Marshaling 的部份做修改，期望達到多點傳輸的功能，或者增加一些容錯的機制在這個部份。讓發生錯誤的時候可以有方法可以回復，或者自動消除錯誤繼續執行。在撰寫程式的時候，我們大概只需要了解到這裡即可。除非我們需要針對 Marshaling 做改善或者增進一些特別的功能，否則圖中最底下的那一層，基本上對於一般的程式設計師來說是透明看不到的。除非有非常特別的需求，否則最底層被更動的機會應該不大。

我們實際實作了兩個可以互相溝通的應用程式。在我們所實作的程式中，兩個應用程式可以互相傳遞資料，並且將結果顯示出來。當我們在第一個程式中更改資料的時候，我們將被更動的部份經由特定界面通知第二個程式。第二個程式在接收到這個通知以後，就知道對方如何更改了資料，然後可以顯示給使用者知道。由於我們設計這兩個測試用的應用程式的時候，是要讓這兩個應用程式既是客戶端也是伺服端。所以任何一個應用程式都可以發出連線的要求給另外一個應用程式，而不需要特定某一個應用程式才可以發出連線的要求。為了達成這個要求，我們設計了如圖四的物件與界面。如圖所示，當任何一個測試程式被啟動的時候，它可以經由呼叫對方所公開出來的界面裡面的 CallMe()函示通知對方要求連線。也就是說，任何一個應用程式只知道對方所公開出來的界面與功能，詳細的實作細節是包含在另外一個程式裡面。這也就是為什麼圖中的 CallMe()與 Click()函式是畫進另外一個程式中的原因了。

起初我們遇到程式即使已經結束了，但仍然佔據系統資源不放的問題。後來我們針對這個問題改變了我們的程式結束時的流程。也就是，當我們要結束程式的時候，我們需要經由任何一個測試程式通知另外一個測試程式程式，告訴它我們已經不再需要它的物件與服務了。這個時候這兩個測試程式就可以將它們的資源釋放掉。如果沒有經由選擇選單達成這個目的，我們不允許程式結束，並且顯示對話框通知使用者結束程式的步驟。這個結束程式的方法雖然

多了一個步驟，但它可以解決程式佔據系統資源不放的問題。



圖四：既是客戶端與伺服端的模型

我們在一台 Windows 95 與 Windows 98 上面分別安裝了 DCOM for Windows 95 與 DCOM for Windows 98，並且在這兩台機器上面個別安裝了我們所撰寫的應用程式。這兩個應用程式在我們的測試下，運作非常的良好。由於我們實作的部份是使用 COM 的物件模型，往後需要再加入其他功能的時候，我們可以輕易地將物件逐漸地加入系統中。當加入的物件越來越多，系統功能越來用複雜的時候，物件導向模型的優點就會越來越明顯。

四、結論與討論

分散式計算已經成為主流，這是因為在高速網路的進步及網際網路的蓬勃發展下，加上物件導向程式也已成為發展可重用性軟體的主流，分散式物件結合這兩大趨勢而漸漸受歡迎。愈來愈多的系統軟體建立成分散式物件的應用程式讓彼此間能分享許多共同的目標，這個計畫主要的目的是研究 COM/DCOM 的特徵並加以應用在醫學網路資訊系統中，以達成可擴充性及容錯機制。

分散式元件物件模型的特徵包括了介面及實作，支援物件具有多重介面，語言中立，即時二元軟體重用，位置透明化，可延伸架構，轉向，版本管理及伺服器生命周期管理。也就是說以 COM/DCOM 為發展分散式程式的平台，則研究者及研發者可以專心於對他們程式比較重要的課題，而不需將大部分的努力投資在建立支援的基礎建設。

藉由本子計畫的執行，我們於計畫執行上獲得了許多寶貴的經驗。這些經驗包括 DCOM 與 CORBA 程式設計界面、軟體元件的發展模式、分散式物件應用程式的撰寫、及研究上分析與比較方法。同時也謝謝國科會給予我們執行此計畫的機會。

五、參考文獻

[1] P. Emerald Chung, Yennun Huang, and Shalini Yajnik, “DCOM and CORBA Side by Side, Step by Step, and Layer by Layer”, C++ Report, Jan, 1998.

[2] D. Rogerson, “Inside COM”, Redmond, Washington: *Microsoft Press*, 1996.

[3] Microsoft Corporation and Digital Equipment Corp., “The Component Object Model Specification”, http://www.microsoft.com/oledev/olecom/title.htm, Oct. 1995.

[4] COM/DCOM Resources, http://www.research.att.com/~ymwang/resources/resources.htm

[5] Y. M. Wang, “Introduction to COM/DCOM”, tutorial slides, http://www.research.att.com/~ymwang/slides/COMHTML/ppframe.htm, 1997

[6] Guy Eddon and Henry Eddon, “Inside Distributed COM”, Microsoft Press, 1998.

[7] A. Avizienis, “The N-version approach to fault-tolerant software”, *IEEE Trans. Software Eng.*, Vol. SE-11, No. 12, pp. 1491-1501, Dec. 1985.

[8] J. Siegel, “COBRA Fundamentals and Programming”, John Wiley & Sons, 1996.

[9] R. Grimes, “Professional DCOM Programming”, Olton, Birmingham, Canada: *Wrox Press*, 1997.