Component Object Model (COM) es una plataforma de Microsoft para componentes de software introducida por dicha empresa en 1993. Esta plataforma es utilizada para permitir la comunicación entre procesos y la creación dinámica de objetos, en cualquier lenguaje de programación que soporte dicha tecnología. El término COM es a menudo usado en el mundo del desarrollo de software como un término que abarca las tecnologías OLE, OLE Automation, ActiveX, COM+ y DCOM. Si bien COM fue introducido en 1993, Microsoft no hizo énfasis en el nombre COM hasta 1997.

Esencialmente COM es una manera de implementar objetos neutrales con respecto al lenguaje, de manera que pueden ser usados en entornos distintos de aquel en que fueron creados, a través de fronteras entre máquinas. Para componentes bien creados, COM permite la reutilización de objetos sin conocimiento de su implementación interna, porque fuerza a los implementadores de componentes a proveer interfaces bien definidas que están separados de la implementación. Las diferentes semánticas de reserva de memoria están acomodadas haciendo a los objetos responsables de su propia creación y destrucción por medio del contador de referencias. Se puede hacer *casting* entre distintas interfaces de un objeto por medio de la función QueryInterface(). El método preferido de herencia en COM es la creación de subobjetos a los que se delegan las llamadas a métodos (llamado agregación).

Aunque estas tecnologías han sido implementadas en muchas plataformas, son principalmente usadas con Microsoft Windows. Se espera que COM sea sustituido, al menos en un cierto grado, por Microsoft.NET, y soporte para Web Services a través de Windows Communication Foundation (WCF). DCOM en red usa formatos binarios propietarios, mientras que WCF usa mensajes SOAP basados en XML. COM también compite con CORBA y Java Beans como sistema de componentes de software.

Los programadores COM construyen software usando componentes de software COM-aware. Diferentes tipos de componentes son identificados por tipo de IDs (CLSIDs), los cuales son identificadores globales únicos, o GUIDs. Cada componente COM revela su funcionalidad por medio de una o más interfaces. Las diferentes interfaces soportadas por un componente son distinguidas las unas de las otros usando interfaces IDs (IIDs), que son también GUIDs.

Las Interfaces COM tienen implementaciones en varios lenguajes, tales como C, C++, Visual Basic, y varios de los lenguajes de script implementados en la plataforma Windows. Todo acceso a los componentes se realiza a través de los métodos de las interfaces. Esto permite que técnicas como inter-proceso, incluso programación entre ordenadores (este último mediante el apoyo de DCOM).

Interfaces

Todos los componentes COM deben implementar, al menos, la interfaz estándar IUnknown, y así todas las interfaces COM son derivadas de IUnknown. La interfaz IUnkown consta de tres métodos: AddRef() y Release(), que implementan conteo de referencias y control del ciclo de vida de las interfaces; y QueryInterface(), que por especificar un IID permite a una llamada recuperar las referencias a las diferentes interfaces que el componente implementa. El efecto de QueryInterface() es similar a dynamic_cast <> en C + + o casting en C # y Java.

Las interfaces de un componente COM es necesario que muestren las propiedades reflexiva, simétrica y transitiva. La propiedad reflexiva se refiere a la capacidad para que la llamada al QueryInterface(), dada una interfaz con el ID de la interfaz, devuelva la misma instancia de la interfaz. La propiedad simétrica hace referencia a que cuando la interfaz B es recuperada de la interfaz A por el QueryInterface(), la interfaz A es asimismo recuperable de la B. La propiedad transitiva es similar a la simétrica, pero requiere que si la interfaz B puede conseguirse de la A, y la C a su vez de la B, entonces la interfaz C debería poder conseguirse de la A.

Una interfaz consta de un puntero a una función virtual que contiene una lista de punteros a las funciones que implementa la función declarada en la interfaz, en el mismo orden que fueron declaradas en la interfaz. Esta técnica de paso de punteros de función de estructuras es muy parecida a la utilizada por OLE 1.0 para comunicar con su sistema de librerías.

COM especifica muchas otras interfaces estándar usadas para permitir comunicación entre componentes. Por ejemplo, una de ellas es IStream, que está puesta para los componentes que tienen semántica de flujo de datos (un componente FileStream usado para leer y escribir

archivos). Tiene los esperados métodos Read y Write para realizar las lecturas y escrituras de flujo. Otra interfaz estándar es IOleObject, que esta puesta para componentes que esperan ser enlazados o empotrados en un contenedor. IOleObject contiene métodos que permiten a los interesados determinar el tamaño de los límites de un componente rectángulo, si el componente soporta operaciones como 'Open', 'Save' y así sucesivamente.

Clases

Una clase en COM se denomina coclass, que es la forma contraída de Component Object class. Una coclass es la forma de COM de definir una clase en el sentido orientado a objetos independiente del lenguaje. Una coclass suministra una implementación concreta de una o más interfaces. En COM, tales implementaciones concretas pueden ser escritas en cualquier lenguaje de programación que soporte desarrollo de componentes COM, como C++, Visual Basic, etc.

Una de las mayores contribuciones de COM al mundo de desarrollo Windows es la toma de conciencia del concepto de separación de interfaz de implementación. Esta toma de conciencia ha influido, sin duda, en el camino programadores al construir los sistemas de hoy. Una extensión de este concepto fundamental es el concepto de una interfaz, múltiples implementaciones. Esto significa que en tiempo de ejecución, una aplicación puede elegir la instanciación de una interfaz de una de las diferentes implementaciones concretas.

Lenguaje de definición de interfaces y bibliotecas de tipos

Las bibliotecas de tipos contienen metadatos que representan los tipos COM. Sin embargo, estos tipos deben ser primero descritos usando el Microsoft Interface Definition Language. Esto es la práctica común en el desarrollo de un componente COM, por ejemplo al empezar con la definición de tipos usando IDL. Un archivo IDL es lo que proporciona COM que permite a los desarrolladores definir las clases orientadas a objetos, interfaces, estructuras, enumeraciones y otros tipos definidos por el usuario en un lenguaje de forma independiente. COM IDL es similar en apariencia a las declaraciones de C / C + + con la adición de palabras clave como «interface» y «library» para la definición de interfaces y de las colecciones de las

clases, respectivamente. IDL también requiere el uso de los atributos entre corchetes antes de las declaraciones para proporcionar información adicional, como el GUID de las interfaces y la relación entre los parámetros de puntero y longitud de los campos.

El archivo IDL es compilado por el compilador MIDL en un par de formas para utilizarlos en varios lenguajes. Para C/C++, el compilador MIDL genera una cabecera independiente del compilador que contiene definiciones de estructuras que emparejan las funciones virtuales de la declaración de interfaces y un archivo C que contiene declaraciones de la interfaz GUIDs. El código fuente C++ para un modulo Proxy puede también ser generado por el compilador MIDL. Este Proxy contiene métodos stubs para convertir llamadas COM en RPC, esto permitido por el DCOM.

Un archivo IDL puede también ser compilado por el compilador MIDL en una librería de tipos (archivo .TLB). Los metadatos binarios contenidos dentro de la librería tiene significado para ser procesados por los compiladores del lenguaje y entornos de tiempo de ejecución (VB, Delphi, el .NET CLR, etc.). El resultado final de tal procesado TLB es que la construcción específica de cada lenguaje están producidas a lo que representa la clase COM definida en la .TLB (y en defnitiva el que se definió en el archivo de origen IDL).

Conteo de referencias

La más importante de todas las interfaz COM, es decir, IUnknown (de la que todas las interfaces COM debe ser derivadas), admite dos conceptos principales: la exploración características a través del método QueryInterface, y la gestión del ciclo de vida del objeto mediante la inclusión de AddRef () y Release (). Conteo de referencias y exploración de característica que se aplican a los objetos (no a cada interfaz de un objeto) y, por tanto, debe tener una implementación centralizada.

Las especificaciones COM requieren de una técnica llamada conteo de referencias para garantizar que los distintos objetos están vivos mientras haya clientes que han adquirido el acceso a uno o más de sus interfaces y, por el contrario, que el mismo objeto esté correctamente eliminados cuando todo código que usa el objeto haya terminado con él y ya

no lo requiere. Un objeto COM es el responsable de la liberación de su propia memoria una vez que su contador de referencias se reduce a cero.

Para su ejecución, un objeto COM generalmente mantiene un valor que se utiliza de referencia para el conteo. Cuando es llamado AddRef () a través de cualquiera de las interfaces del objeto, este valor se incrementa. Cuando se llama a Release(), este número entero se decrementa. AddRef () y Release () son los únicos medios por los que un cliente de un objeto COM es capaz de influir en su ciclo de vida. El valor interno sigue siendo un miembro privado del objeto COM y nunca será accesible directamente.

Instanciación

COM normaliza el proceso de instanciación (es decir, la creación) de objetos COM, al exigir la utilización de la clase Factories. Para cada objeto COM que se creó, dos parámetros asociados deben existir:

- Una clase ID.
- Una clase Factory.

Cada clase o CoClass COM debe estar asociada con una única ID de clase (un GUID). También debe ser asociada con su propia clase Factory (que se logra mediante el uso de un registro centralizado). Una clase Factory es en sí misma un objeto COM. Es un objeto que debe exponer la IClassFactory o IClassFactory2 (este último con la soporte licencias de apoyo) interfaz. La responsabilidad de dicho objeto es la creación de otros objetos.

Una clase Factory suele estar contenida en el mismo código ejecutable (es decir, el servidor de código) como el propio objeto COM. Cuando una clase Factory está llamada a crear un objeto objetivo, este objetivo del objeto es el id de clase que debe ser proporcionado. Así es como la clase Factory sabe qué clase de objeto instancia.

Una sola clase Factory objeto puede crear objetos de más de una clase. Es decir, dos objetos de diferente ids de clase pueden ser creados por la misma clase de Factory objeto. Sin

embargo, esto es transparente para el sistema de COM.

Delegando la responsabilidad de creación de un objeto en objeto separado, se consigue un mayor nivel de abstracción y el desarrollador tiene una mayor flexibilidad.

A fin de que las aplicaciones cliente puedan adquirir las clases de objetos Factory, los servidores COM debe exponerlos adecuadamente. Una clase Factory está expuesta de forma diferente, en función de la naturaleza del código del servidor. Un servidor que está basado en DLL debe exportar una función global DllGetClassObject (). Un servidor EXE que está basado en los registros de clases Factory en tiempo de ejecución a través de la función de la API de Windows de CoRegisterClassObject ().

Programación

COM es un estándar binario y puede ser desarrollado en cualquier lenguaje de programación capaz de comprender e implementar sus tipos de datos binarios definidos e interfaces.

Bibliotecas en tiempo de ejecución (en situaciones extremas, los programadores) son las responsables de que entren y salgan del entorno COM, instanciación y conteo de referencias de objetos COM, objetos para consultar información sobre la versión, la codificación de aprovechar las avanzadas versiones de objetos.

Transparencia de aplicaciones y red

Los Objetos COM pueden ser instanciados y referenciados en un proceso, a través de las fronteras de un proceso dentro de equipo y, a través de una red, usando la tecnología DCOM. Salir del proceso y de los objetos remotos puede utilizar serialización para enviar las llamadas a los métodos y valores de retorno hacia atrás y hacia delante. La serialización es invisible para el objeto y el código usando el objeto.

Críticas

Mensaje de bombeo

Cuando un STA se inicializa que crea una ventana oculta que se utiliza para la inter-apartamento y entre procesos de enrutamiento de mensajes. Esta ventana debe tener su cola de mensajes regularmente bombeado. Esta construcción se conoce como un mensaje bomba. En versiones anteriores de Windows no hacerlo podría causar en todo el sistema bloqueos. Este problema es especialmente desagradable porque algunas APIs de Windows inicializa COM como parte de su aplicación, lo que provoca una fuga de los detalles de implementación.

Conteo de referencias

El conteo de referencias dentro de COM puede causar problemas si dos o más objetos están referenciados circularmente. El diseño de una aplicación debe tener esto en cuenta para que los objetos no se queden huérfanos.

Los objetos pueden también dejar activa la cuenta referencias si el COM «caso sumidero» es el modelo utilizado. Dado que el objeto que dispara el evento necesita una referencia al objeto para reaccionar al evento, el conteo de referencias a objeto nunca llega a cero.

DLL hell

Debido a la ubicación de cada uno de los componentes se almacenan en una ubicación de todo el sistema (el registro de Windows), puede haber una sola versión de un cierto componente instalado. Por lo tanto, COM sufre seriamente del DLL hell, en que dos o más aplicaciones requieren diferentes versiones de un mismo componente.

Windows XP introduce un nuevo modo de registro de objetos COM «Registro libre de COM». Este servicio hace posible que las aplicaciones que necesiten para instalar los objetos COM almacenaran toda la información de registro necesaria para COM registro en la solicitud del

directorio, en lugar de en el registro global, en donde, en rigor sólo una única solicitud se utilizan. DLL hell, se pueden evitar mediante Registro-COM libre, la única limitación que se requiere, al menos, Windows XP o posteriores versiones de Windows y que no debe utilizarse para EXE, COM o servidores en todo el sistema de componentes, tales como MDAC, MSXML, DirectX o Internet Explorer.