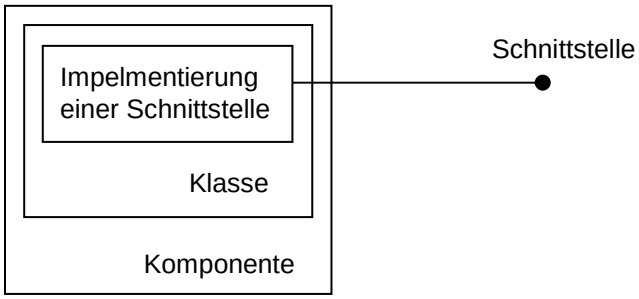
COM – Component Object Model

Das Ziel von COM ist die Zerlegung von Softwaresystemen in wiederverwendbare Bausteine (Komponenten). Diese werden binär als „dynamisch ladbare Bibliotheken“ (DLL) oder ausführbare Programme (EXE) gespeichert und Ausgetauscht. COM standardisiert Mechanismen, mit denen die in diesen Komponenten implementierten Klassen angesprochen und von jedermann verwendet werden können. Das aufrufende Programm wird dabei als Client bezeichnet, für den die Komponente als Server Funktionalität bereitstellt.

Für jede Komponente muß zuvor genau festgelegt werden, welche Funktionalitäten die jeweilige Komponente zur Verfügung stellt. Dies erfolgt durch die Definition von Schnittstellen.



Für einige Anwendungsgebiete hat man bereits fertige Schnittstellen eingeführt, die in *Component Frameworks* standardisiert werden (z.B. Definition von Benutzerelemente und ActiveX).

Die Definition von COM-Schnittstellen erfolgt in einer eigenen Sprache, *der Microsoft Interface Definition Language (MIDL*). Neben dieser alphanumerischen Darstellung können die Informationen auch binär in einer *Typenbibliothek (TLB)* abgelegt werden. Die Konvertierung erfolgt mit dem von VC++ bereitgestellten Programm MIDL.EXE. Obwohl es keine Rolle spielte welche der beiden Formate verwendet werden, bieten Typenbibliotheken den Vorteil, daß sie als Ressource an die Komponente angehängt und mit dieser verbreitet werden können.

Autor Dr. Uwe Altenschmidt
Datei com.doc

LITTLE softwarehouse GmbH
Zentnerstr. 19, D-80789 München
www.littlesoft.com

Die Definition einer Schnittstelle kann ähnlich wie die Deklaration einer Klasse in C++ betrachtet werden. Das Schlüsselwort *CLASS* wird jedoch durch *INTERFACE* ersetzt. Die Schnittstelle repräsentiert eine abstrakte Klasse, deren Methoden erst nach einer Vererbung implementiert werden dürfen. Attribute können nicht deklariert werden. Alle Schnittstellen müssen mindestens die Methode der Schnittstelle *IUnknown* (durch COM-Standard vorgegeben) enthalten.

Die Deklaration der einzelnen Methoden erfolgt ähnlich wie in C++, jedoch liefern alle Methoden einen Fehlercode vom Datentyp *HRESULT*. Weiterhin muß für jeden Parameter ein *[in], [out] oder [inout]* angegeben werden. Für die Rückgabe von Ergebnissen empfehlen sich Zeigervariablen.

Die Schnittstellen müssen nun den eigentlichen COM-Klassen zugeordnet werden, von denen sie später implementiert werden sollen. Hierfür nutzt man das Schlüsselwort *COCLASS*. Durch das Schlüsselwort *LIBRARY* werden die Deklarationen zu einer Typenbibliothek zusammengefaßt. Mit *IMPORT und IMPORTLIB* können andere Dateien bzw. Typenbibliotheken eingebunden werden.

Für jede Schnittstelle, Klasse und Typenbibliothek muß ein entsprechender Indentifikator angegeben werden (z.B. [uuid]).

Eine fertiggestellte Schnittstellendefinition ist als ein Vertrag zu betrachten. Das Hinzufügen von Methoden ist nur durch die Definition einer neuen Schnittstelle möglich. Durch Vererbung kann diese aus einer vorhandenen hervorgehen.

Eine COM-Klasse kann praktisch eine beliebige Anzahl von Schnittstellen unterstützen. Um eine Befragung nach entsprechenden Implementierungen zu ermöglichen, muß stets eine Implementierung der Basisschnittstelle *IUnknown* vorhanden sein. Da alle Schnittstellen die Methoden von *IUnknown* erben, ist durch Polymorphie sichergestellt, daß für jede Klasse eine eindeutige Implementierung der Basisschnittstelle vorhanden ist. Als zweites Aufgabengebiet fällt der Schnittstelle *IUnknown* die Referenzenzählung zu. Client-Programme können die Zeiger auf eine Schnittstelle untereinander austauschen. Dadurch entsteht die Frage, ab wann ein COM-Objekt oder eine Komponente nicht mehr benötigt wird. Und aus dem Speicher entfernt werden kann. Jeder Client signalisiert durch den Aufruf von *AddRef*, daß er die Schnittstelle einer Klasse unter Beschlag nimmt. Wird der Zeiger nicht mehr benötigt, wird dies durch *Release* mitgeteilt. *AddRef* und *Release* sind mit einer Zählervariablen umzusetzen.

Die Instanz einer COM-Klasse kann in verschiedenen Zusammenhängen nicht direkt erzeugt werden (z.B. einzige Einsprungstelle ist *main*). Ein entsprechender Mechanismus zum Erzeugen von Instanzen muß deshalb dem COM beim Programmstart mitgeteilt werden. Dies wird durch das Konzept der *Klassenfabriken* realisiert.

Klassenfabriken sind gewöhnliche COM-Klassen, die eine spezielle Schnittstelle mit dem Namen IClassFactory unterstützen. Für jede echte COM-Klasse muß eine solche Klassenfabrik bereitgestellt werden. Die Implementierung der Klassenfabriken sollte durch parametrisierbare Klassenfabrik erfolgen.

Hat das Client-Programm erst einmal einen Zeiger auf *IClassFactory* erhalten, können beliebig viele Instanzen der entsprechenden Klasse hergestellt werden (Methode *CreateInstance*).

Natürlich müssen die Klassenfabriken und die entsprechenden COM-Klassen zu einer Komponenete zusammengefasst werden. Dabei müssen drei verschiedene Typen von COM-Komponenten berücksichtigt werden:

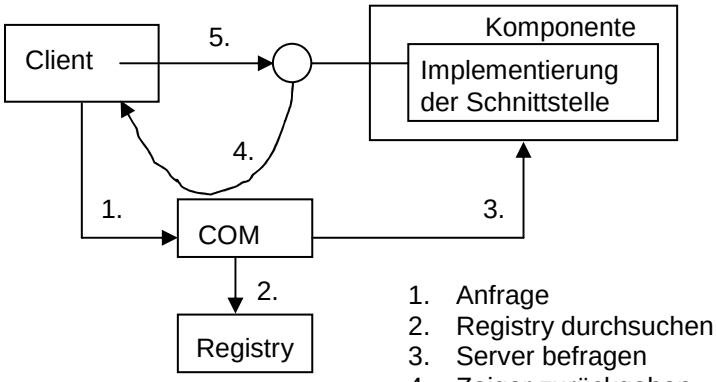
- In-Process-Server: als DLL umgestzt
- Local Server: als EXE umgesetzt
- Remote Server: erweitert das Konzept eines Local Servers auf Netzwerkanwendungen

Für ausführbare Pragramme muß ein anderer Mechanismus angewandt werden , als von DLLs. Nach Start der Komponente muß diese alle Klassenfabriken durch Aufruf der Funktion *CoRegisterClassObject* bei COM anmelden. Fragt ein Client nach einer Klasse, so liefert COM automatisch den entsprechenden Zeiger auf die Klassenfabrik zurück. Wird der Server beendet, ist ein Abmelden durch Aufruf von *CoRevokeClassObject* notwendig.

Nach der Fertigstellung der Komponenten bleibt die Frage, woher COM weiß, von welcher Komponente eine geforderte Klasse bereitgestellt werden soll. Dabei kommt der Registrierung (Registry eine zentrale Rolle zu. Alle für COM und ActiveX notwendigen Informationen werden unter dem Schlüssel *HKEY_CLASS_ROOT* eingetragen.

Die Entwicklung einer Client-Anwendung muß natürlich den umgekehrten Weg gehen. Zunächst wird die COM-Bibliothek in den Speicher geladen. Anschließend wird ein Zeiger auf die Klassenfabrik erfragt. COM durchsucht die Registry nach einer

Eintragung und lädt die gefundenen Komponenten in den Speicher. Dann wird der In-Process-Server nach einem Zeiger befragt, und dieser wird an den Client zurückgegeben. Im nächsten Schritt werden Methoden der Klassenfabrik aufgerufen und so die eigentliche Instanz der Klasse erzeugt. Abschließend wird das Objekt freigegeben und die COM-Bibliothek aus dem Speicher entfernt.



1. Anfrage
2. Registry durchsuchen
3. Server befragen
4. Zeiger zurückgeben
5. Interface aufrufen

Wird Software-Entwicklung durch COM einfacher ?

Es ist zu beachten, dass VB die Implementierung der notwendigen Mechanismen quasi automatisch erledigt. Auch für C++ existieren fertige Klassenbibliotheken, die entsprechende Implementierungen weitgehend von selbst bereitstellen. Für größere Projekte bieten sowohl MFC als auch die „Active Template Library“ (ATL) entsprechende Hilfestellungen.

Seine eigentliche Stärke entfaltet COM erst bei der Erstellung grafischer Benutzerelemente. Als Basis dient hierfür die von ActiveX standardisierte Schnittstellenarchitektur. Auch Verbunddokumente, die in fremden Container-Applikationen bearbeitet werden können, sind dadurch möglich. Diese als OLE bekannte Technik ist inzwischen mit ActiveX eng verwachsen. Auch in Netzwerken eröffnen sich durch das DCOM viele neue Wege.

Quelle:

L. Röder (1999) Komponentenentwicklung mit dem „Component Object Model“, OBJEKTspektrum 3, 18-25