

DCOM

Vladimír Sklenář¹, Václav Snášel²

1 Úvod

Základní otázka softwarového inženýrství je problém produktivity práce. Její řešení můžeme spojovat s možností znovupoužití existujících řešení. Znovupoužitím na úrovni návrhu architektury systému se zabývají návrhové vzory. OOP využívá pro znovupoužití dědičnost. Ta je většinou omezena na znovupoužití zdrojového textu v konkrétním programovacím jazyku. Dalším krokem je znovupoužití na úrovni vykonatelného kódu. To se snaží řešit softwarové komponenty. Cílem je přiblížit tvorbu softwarových aplikací k práci návrhářů hardware a umožnit jejich sestavování z existujících součástí (komponent), které budou dostupné na trhu. To se týká nejenom vlastního vytvoření aplikace, ale i její údržby a dalšího rozvoje. Ta by byla založena na pouhé výměně součástí, to je komponent. V současné době existuje několik systému, které se tyto představy snaží realizovat. Jedním z nich je DCOM.

2 Co je DCOM

Component Object Model (COM) je specifikace, která určuje způsob vytváření softwarových komponent. Poskytuje standard umožňující jejich vzájemnou spolupráci a zaměnitelnost. COM komponenta obsahuje vykonatelný kód ve formě dynamické knihovny (DLL) nebo spustitelného souboru (EXE). Tento kód realizuje služby, které komponenta formou přesně stanoveného rozhraní nabízí svému okolí. COM především určuje binární rozhraní, které musí jednotlivé objekty splňovat, a proto jsou komponenty nezávislé na programovacím jazyk, ve kterém byly vyvinuty. Od svého počátku byl COM navrhován tak, aby podporoval i distribuované zpracování. Aby tedy umožňoval vytváření objektů na jiném počítači a volání jejich metod způsobem, který je shodný s využíváním lokálních objektů. Tato možnost však v prvních verzích nebyla implementována. Byla zahrnuta až v Distributed COM (DCOM), který byl uveden v roce 1996. DCOM je tedy rozšíření COM o podporu distribuovaného zpracování. Vše co platí pro COM proto platí i pro DCOM. COM objekty i klienti mohou být okamžitě používány v prostředí DCOM. DCOM navíc přináší některé nové prvky, které především podporují efektivnost a bezpečnost práce v síťovém prostředí.

2.1 Historie

COM byl vyvíjen s cílem umožnit vyvíjení aplikací, které by byly flexibilní a přizpůsobitelné potřebám uživatele. Původní záměr byla podpora technologie vytváření dokumentů OLE. První verze OLE byla založena na mechanismu dynamické výměny dat (DDE). Při návrhu druhé verze OLE již její architekti chápali problematiku složených dokumentů jako speciální případ obecnějšího problému: jak zajistit, aby různé softwarové komponenty mohly vzájemně spolupracovat. OLE 2 bylo tedy založeno na COM. Od té doby se COM stal v podstatě standardem pro vývoj komponent a málokterá aplikace pro Windows nebo Windows NT jej nevyužívá. V roce 1996 s příchodem Windows NT 4.0 byl COM doplněn o možnost přístupu ke komponentám na jiných počítačích. V současné době jsou vytvářeny implementace DCOM i na jiných platformách než Win32.

¹ RNDr. Vladimír Sklenář, KMI PřF UP, Tomkova 40, Olomouc, (068) 412210,
e-mail: vladimir.sklenar@upol.cz

² RNDr. Václav Snášel CSc., KMI PřF UP, Tomkova 40, Olomouc, (068) 412210,
e-mail: vaclav.snasel@upol.cz

3 Rozhraní

Jedním ze základních principů uplatněných při návrhu COMu je princip zapouzdření. Objekt je chápán jako černá skříňka, se kterou lze komunikovat pouze pomocí rozhraní, to je seznamu funkcí, které je schopen vykonat. Jinak řečeno, pokud chceme objekt využívat, musíme znát jeho rozhraní. Jeden objekt přitom může mít poskytovat více rozhraní (obvykle má minimálně dvě). Je tedy nutné každé rozhraní jednoznačným způsobem zapsat a zpřístupnit všem případným klientům. K tomu je určen jazyk IDL (Interface Definition Language). COM IDL je čistě deklarativní jazyk, který vychází ze syntaxe jazyka C. Win32 SDK obsahuje překladač tohoto jazyka (MIDL.EXE). Jeho úkolem je vygenerovat soubory obsahující hlavičkové soubory C++ pro abstraktní třídy odpovídající rozhraním v IDL. MIDL dále generuje soubory se zdrojovým textem vazebních procedur (proxy, stub) a knihovnu typů. Ta obsahuje informace o rozhraní v binární podobě.

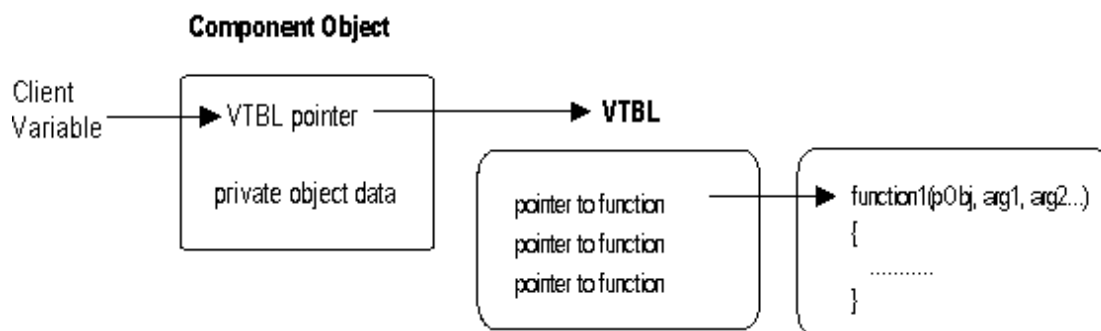
Aspekty, které nemají analogii v jazyce C, se v IDL zapisují pomocí atributů. Ty jsou uzavřeny hranatých závorkách []. S pomocí atributů zaznamenáme např. informaci jednoznačně identifikující dané rozhraní nebo informaci o charakteru jednotlivých parametrů metod (t.j. zda jsou vstupní nebo výstupní). Následuje příklad deklarace jednoduchého rozhraní

```
[ object, uuid(f1b66600-1cb7-11d0-a7e5-444553540000) ]
interface IData : IUnknown {
    import "unknwn.idl";
    HRESULT SetData([in] BYTE* buffer, [in] long size);
    HRESULT GetData([out] BYTE** buffer, [out] long* size);
}
```

IDL tedy mimo jiné podporuje základní datové typy a jednoduchou dědičnost. Jedná se však pouze o dědičnost rozhraní. Implementační dědičnost není v rámci COM podporována. Každé rozhraní musí mít ve své hlavičce dva atributy. Atribut [object] určuje, že se jedná o COM rozhraní. Druhý atribut obsahuje řetězec jednoznačně identifikující dané rozhraní. Pro tyto účely se v rámci COM používá GUID (Globally Unique Identifiers). Je to 128 bitové číslo, pro které je garantována jednoznačnost v čase i prostoru. Je-li GUID použito pro označení rozhraní, pak se nazývá IID. Součástí COM jsou nástroje, které GUID vygenerují.

4 Přístup k rozhraní

Pro přístup k rozhraní poskytne COM klientu ukazatel na tabulku ukazatelů označovanou jako virtuální tabulka (vtable). Její struktura odpovídá tabulce, kterou generují překladače MS Visual C++ pro implementaci virtuálních funkcí. Jednotlivé položky této tabulky odkazují na implementace metod v některém z objektů podporujících toto rozhraní.



5 Rozhraní IUnknown

Každé COM rozhraní musí být potomkem (přímým nebo nepřímým) rozhraní IUnknown. To obsahuje pouze tři metody: QueryInterface, AddRef a Release. Tyto tedy musí implementovat každý COM objekt. Úkolem metody QueryInterface je poskytnout klientu přístup ke všem rozhraním, které daný objekt podporuje, popř. ohlásit, že objekt dané rozhraní nepodporuje. Tato metoda má klíčovou úlohu při správě verzí objektu. V COM platí pravidlo, že jednou zveřejněné rozhraní nelze modifikovat. Pokud jej chceme např. rozšířit o nové metody, musíme vytvořit zcela nové rozhraní. COM objekt může podporovat více rozhraní, to znamená, že i různé verze téže aplikace. Voláním QueryInterface může klient zjistit se kterou verzí právě pracuje a podle toho upravit své chování.

Metody AddRef a Release zajišťují evidenci počtu klientů využívajících dané rozhraní. Funkce AddRef se volá v okamžiku, kdy klient získá přístup k objektu a zajistí zvýšení čítače o jedničku. Funkce Release se volá v okamžiku, kdy klient ukončuje práci s rozhraním a její úlohou je naopak snížit čítač odkazů, a v okamžiku, kdy dosáhne nulové hodnoty, objekt zruší. Je pochopitelné, že postup vyžaduje korektní chování všech zúčastněných.

6 Třídy

Každý COM objekt je instancí třídy a každá třída je jednoznačně identifikována GUID, které se označuje zkratkou CLSID. COM využívá CLSID pro zaznamenání odkazu na server obsahující binární kód třídy. Jeden server může obsahovat více tříd. COM chápe třídu jako konkrétní implementaci skupiny rozhraní. Dané rozhraní může být implementováno více třídami. Třída vždy obsahuje objekt třídy, který slouží jako vstupní bod do implementace třídy. Často tento objekt slouží pro vytváření nových instancí (Class factory).

7 Servery

Každý COM objekt je implementován pomocí serveru. Server obsahuje kód metod jednotlivých rozhraní. COM rozlišuje tři základní druhy serverů:

- vnitřní (in-process) – objekty jsou implementovány v DLL a jsou vykonávány v témže adresovém prostoru jako klient.
- lokální (local) – objekty jsou vykonávány jako samostatný proces na témže stroji jako klient.
- vzdálený (remote) – objekty jsou vykonávány na jiném stroji než klient.

V registru operačního systému se zaznamenávají odkazy na jednotlivé servery. Změna umístění serveru pak znamená pouze změnu tohoto odkazu.

8 Vytvoření a inicializace objektu

Při žádosti o vytvoření objektu zadá klient IID rozhraní, které chce zpřístupnit a CLSID požadované implementace. COM vyhledá v registru odkaz na příslušný server, zajistí jeho spuštění a poté si vyžádá ukazatel na požadované rozhraní. Ten pak předá klientu. Takto vytvořený objekt neobsahuje žádná data. První krokem klienta tedy ve většině případu je zajištění korektního naplnění stavu objektu.

9 Znovupoužití

COM nepodporuje dědičnost implementace. Tvůrci COM ji nepovažují za vhodný nástroj pro heterogenní prostředí. Stejného efektu se dosahuje tím, že objekt může mít více rozhraní a je podporováno zapouzdření rozhraní vnitřní komponenty a jeho zpřístupnění klientu. Znovupoužití je tak dosaženo spíše vlastněním než dědičností. COM poskytuje dva mechanismy s jejichž pomocí může jeden objekt využívat druhý: delegování (containment/delegation) a agregaci (agregation). Objekt, který je využíván se označuje jako vnitřní (inner object) a objekt, který jej využívá jako vnější (outer object). V případě delegování se vnější objekt chová jako klient vnitřního objektu a používá služeb vnitřního objektu pro implementování svých vlastních funkcí. Při použití agregace zpřístupní vnější objekt některá rozhraní vnitřního objektu tak jako kdyby byly implementovány přímo vnějším objektem. Na rozdíl od delegování vyžaduje implementování agregace podporu od vnitřního objektu. Jde především o způsob implementování rozhraní IUnknown.

10 DCOM

vše co platí pro COM platí i pro DCOM. COM objekty i klienti mohou být okamžitě používány v prostředí DCOM. DCOM však přináší některé nové prvky. Chcete-li zajistit, aby interakce mezi klientem a objektem byla efektivní a bezpečná, pak je potřebné dopsat některé úseky kódu.

Příkladem problému, který v prostředí jednoho počítače nebylo nutné řešit je havárie serveru nebo klienta. V tom případě je nutné zajistit, aby příslušné objekty byly informovány a mohly tak na vzniklou situaci reagovat. DCOM tento problém řeší mechanismem pinkání. Zajišťuje, že klient a objekt (přesněji jejich proxy a stub) si periodicky zasílají zprávy. Pokud zpráva opakovaně nedorazí, pak se předpokládá, že příslušná strana havarovala.

10.1 Zabezpečení přístupu k vzdáleným objektům

V případě, že by žádným způsobem nebyl zabezpečen přístup k vzdáleným objektům, mohlo by velmi jednoduše dojít k vytvoření nebo použití objektu neautorizovaným procesem nebo uživatelem, což by mohlo vést k ovlivnění bezpečnosti celého systému. Z tohoto důvodu také DCOM ve spolupráci s ORPC podporuje zabezpečení přístupu k objektům.

Existují v zásadě dvě oblasti zabezpečení přístupu k objektům. V první řadě se jedná o určení práv, kdo může který vzdálený objekt vytvořit, tzv. zabezpečení aktivace objektu. V druhé řadě pak jde o omezení přístupu k již existujícímu vzdálenému objektu, tzv. zabezpečení volání objektu. V DCOM je možno pracovat s oběma úrovněmi zabezpečení.

10.1.1 Zabezpečení aktivace objektu

Zabezpečení aktivace objektu je určeno nastavením položek registru konkrétního počítače. Nejširší možné nastavení může povolit nebo zcela zakázat použití daného stroje pro spouštění distribuovaných objektů. V případě, že je spouštění distribuovaných objektů zcela zakázáno, žádní vzdálení klienti nemohou na tomto stroji vytvořit objekt, ani se nemohou k již existujícímu objektu připojit.

Dále je možné používat zabezpečení na úrovni jednotlivých tříd objektů. V tomto případě je možné nastavit jisté implicitní zabezpečení pro aktivaci a přístup k instancím tříd a dále pak pro jednotlivé třídy použít seznam přístupových práv. V tomto případě je však třeba identifikovat klienta, který se pokouší o přístup k objektu.

DCOM umožňuje, aby nově vytvořený objekt získal oprávnění speciálního uživatele, podobně, jako služby Windows NT. Dále je možné, aby objekt získal oprávnění shodné s oprávněním klienta, který jej vytvořil, a konečně je také možné, aby oprávnění objektu bylo shodné s oprávněním interaktivního uživatele, pro kterého je objekt vytvořen.

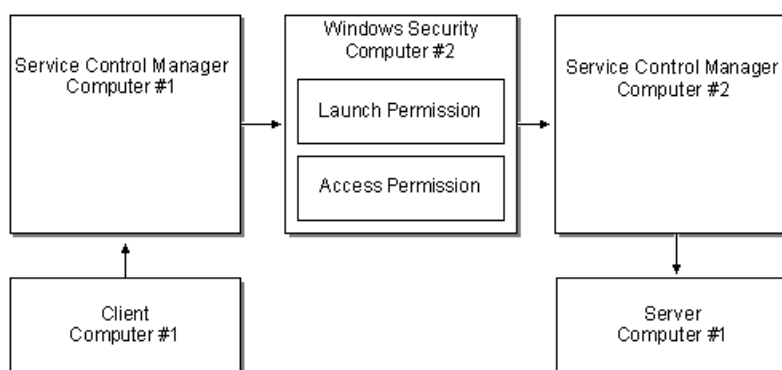
1.1.2 Zabezpečení volání objektu

Po tom, co je objekt vytvořen, čeká na žádosti od vzdálených klientů. V tomto případě je vhodné pomocí nějakého mechanismu zabezpečit volání objektu před neoprávněným přístupem. K tomuto se používá následujících služeb:

- Ověření věrohodnosti – zjišťuje, zda ten, kdo se za někoho vydává, jím skutečně je.
- Ověření oprávnění – zjišťuje, zda klient má pro danou akci oprávnění.
- Zabezpečení integrity dat – zajišťuje, že data nebyla při přenosu po síti modifikována.
- Zabezpečení utajení dat – zajišťuje, že data nemohou být při přenosu čtena.

Tyto služby je možné zabezpečit různými mechanismy. Z tohoto důvodu je DCOM konstruován takovým způsobem, aby dopředu žádný z mechanismů nevylučoval. V současné době je podporován jen mechanismus zahrnutý ve Windows NT, který podporuje všechny čtyři služby.

Podobně jako v případě zabezpečení aktivace objektu, i v případě zabezpečení volání existují dvě úrovně bezpečnosti. Lze totiž nastavit automatické zabezpečení volání metod rozhraní pro konkrétní proces a dále je možné nastavit zabezpečení pro každý interface objektu jednotlivě. V případě automatického zabezpečení nezáleží na tom, zda objekt pracuje ve funkci klienta nebo serveru. Voláním jediné funkce knihovny DCOM je možné nastavit seznam přístupových práv pro všechny objekty běžící v daném procesu, úroveň věrohodnosti procesu, tj. jak často se má provádět ověření věrohodnosti klienta, a seznam služeb, které se mají používat pro ověření věrohodnosti. Výhodou tohoto postupu je, že v rámci jediného volání jsou nastaveny základní vlastnosti bezpečnosti volání objektů. Pokud se neprovede ani toto jednoduché volání, používá se implicitní nastavení bezpečnosti pro konkrétní systém.



Zabezpečení na úrovni interface objektu je mnohem mocnější. Oproti automatickému zabezpečení se v tomto případě rozlišuje mezi klientem a serverem. Pro nastavení úrovně oprávnění klienta se používá interface IClientSecurity, zatímco pro nastavení serveru je třeba získat ukazatel na IServerSecurity. Pomocí těchto rozhraní pak lze modifikovat práva pro používání jednotlivých interface. Ve skutečnosti se však nastavují práva jen pro proxy resp. stub konkrétního interface.

Pokaždé, když klient volá server, dochází nejprve k ověření věrohodnosti klienta. Toto ověření provádí COM automaticky a využívá přitom zabezpečení integrity a utajení dat. V případě, že ověření věrohodnosti dopadne bezchybně, následuje ověření úrovně oprávnění klienta vzhledem k požadované akci. Teprve potom následuje vlastní volání metody serveru.

11 Nástroje pro vývoj aplikací

Pro tvorbu COM serveru poskytuje Microsoft několik nástrojů. API funkce obsažené v COM knihovně Win32 SDK. Jejich využití vyžaduje napsání poměrně velkého množství kódu, je proto hodně pracné a také pravděpodobnost zanesení chyb je větší. Na druhou stranu ovšem poskytuje nejvyšší možnosti řízení aplikace. Druhou možností je využití knihovny MFC. Ta byla navržena především pro tvorbu programů s grafickým rozhraním. Protože i uživatelské rozhraní obsahuje rysy COM, byly metody pro podporu COM rozhraní doplněny do většiny tříd MFC. Vývojové prostředí navíc poskytuje nástroje pro snadnou tvorbu standardních rozhraní. S jejich podporou nemusí mít programátor žádné podrobné znalosti o COM. Avšak výsledná aplikace je velká. Třetí možnost je knihovna ATL. Je to knihovna šablon, navržena především pro tvorbu COM serveru. Vývojové prostředí zajistí vygenerování výchozích zdrojových souborů. Je podporována tvorba všech typů serverů (inproc, local, NT service) i rozhraní (standard, dispatch, dual).

12 Další perspektivy COM a DCOM

V současné době firma Microsoft pracuje na nové architektuře, která byla nazvána Windows Distributed interNet Applications (DNA). Tato architektura má především sloužit pro jednodušší tvorbu aplikací, které budou schopny plně využít integrace osobního počítače a Internetu. Cílem je vytvoření infrastruktury aplikace, která umožní vývojářům transparentní používání všech služeb jak lokálních tak rozsáhlých sítí.

Součástí Windows DNA je i následník COM, s příhodným označením COM+. COM+ je zpětně kompatibilní s COM, takže dříve napsané komponenty budou spolupracovat bez nejmenších problémů. COM+ však navíc poskytuje pro nově vytvářené komponenty mnohem lepší infrastrukturu. V současnosti musí vývojář strávit mnoho času tím, aby napsal kód komponenty, který nemá přímou souvislost s funkcí komponenty, ale spíše s funkcí COM. COM+ tuto rutinní činnost odstraňuje tím, že implementaci základních prvků COM, jako jsou interface IUnknown, IDispatch, IPersist, class factory a dalších obsluhuje ve vlastní režii. Dále COM+ umožňuje používání atributů pro modifikaci chování komponent, rozšiřuje již existující, avšak ne vždy plně implementované služby, jako například transakční zpracování, poskytuje nový mechanismus pro distribuci událostí na základě jejich publikování a objednání, umožní vyvolání metod na základě asynchronního modelu a mnoho dalšího.

Literatura

- [1] Chappel D.: *Understanding ActiveX and OLE*. Microsoft Press, 1996
- [2] Box D.: *Essential COM*. Addison-Wesley, 1998
- [3] Grimes R.: *Professional DCOM Programming*. Wrox Press, 1997
- [4] Rogerson D.: *Inside COM*, Microsoft Press, 1997

