ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: INFORMATYKA z. 34 Nr kol. 1381

Krzysztof ZIÓŁKOWSKI, Andrzej KOWALCZYK, Grzegorz HRYŃ
Politechnika Śląska, Instytut Informatyki

WYBRANE METODY TWORZENIA APLIKACJI W ROZPROSZONYM ŚRODOWISKU SIECI INTERNET / INTRANET

Streszczenie. Artykuł opisuje trendy rozwojowe współczesnych systemów komputerowych opartych na systemach operacyjnych Microsoft Windows i sieci Internet/intranet. Opisano interfejsy programowe: COM, DCOM i technologię ActiveX, które, zdaniem autorów, będą podstawowym sposobem tworzenia oprogramowania w najbliższym czasie.

SELECTED METHODS OF APLICATION DEVELOPMENT IN DISTRIBUTED INTERNET / INTRANET ENVIRONMENT

Summary. This paper describes present development of computer systems based on Microsoft Windows and Internet/intranet. According to authors, described here interfaces: COM, DCOM and ActiveX technology will be basic ones in application programming soon.

1. Kierunki rozwoju systemów komputerowych

Bardzo dynamiczny rozwój technologii informatycznych, szczególnie tych związanych z siecią Internet, spowodował w ostatnich latach powstanie wielu metod i mechanizmów przeznaczonych do tworzenia oprogramowania użytkowego. Metody te ewoluowały z wcześniejszych rozwiązań, które czasami do tworzenia oprogramowania nie były pierwotnie przeznaczone (np. HTML). Jednocześnie umocniła się pozycja systemów operacyjnych Microsoft Windows, które w chwili obecnej są praktycznie jedynym liczącym się na rynku środowiskiem użytkownika.

1.1. Zatarcie różnic pomiędzy sieciami LAN i WAN

Oprogramowanie współczesnych sieci komputerowych uległo znacznej zmianie w stosunku do stanu sprzed lat kilku. Można stwierdzić, że na poziomie interfejsów programowych dostępnych dla twórców aplikacji nie ma różnicy, czy program ma pracować w sieci lokalnej, czy rozległej, różnice polegają tylko na szybkości transmisji. Widać wyraźną migrację technologii powstałych dla sieci Internet do sieci lokalnych (intranet) i odwrotnie, rozwiązania dostępne dawniej tylko w sieciach lokalnych lub nawet tylko na jednym komputerze mogą być obecnie stosowane w sieci Internet (np. DCOM).

1.2. Tworzenie aplikacji w oparciu o WWW

Język HTML i usługa znana jako WWW w sieci Internet powstały z myślą o publikowaniu informacji. W chwili obecnej jednak zarówno przeglądarki, jak i serwery WWW posiadają mechanizmy, które umożliwiają tworzenie w oparciu o nie skomplikowanych i wydajnych systemów informatycznych.

Przeglądarki WWW rozbudowano o możliwości:

a) wykonywania skryptów zamieszczonych w postaci źródłowej na stronach WWW (Java Script i Visual Basic Script),

b) wykonywania kodu binarnego specjalnie przygotowanych programów - komponentów (ActiveX components, Java Applets),

dzieki czemu możliwe jest tworzenie praktycznie dowolnie rozbudowanego interfejsu użytkownika na stronach WWW.

Serwery WWW rozbudowano o mechanizmy takie, jak:

a) wykonywanie programów użytkownika i publikowanie wyników ich pracy na stronach WWW (CGI, IIS API, IDC) [1],

b) wykonywanie skryptów na serwerze i umieszczanie komponentów wykonywanych na serwerze (Active Server Pages) [3],

które umożliwiają stosowanie serwerów WWW jako uniwersalnego narzędzia do prezentacji danych, niezależnie od źródła ich pochodzenia.

Język HTML uzupełniony o możliwość wykonywania skryptów Visual Basic Script i Java Script w połączeniu z komponentami, w których można zaimplementować dowolnie skomplikowany algorytm, czy nie ograniczony składnią HTML interfejs użytkownika, jest narzędziem umożliwiającym tworzenie aplikacji, które są bardzo konkurencyjne w stosunku do oprogramowania tworzonego tradycyjnymi metodami.

1.3. Oprogramowanie z wbudowanym dostępem do usług internetowych

Innym trendem rozwojowym jest korzystanie z usług internetowych przez aplikacje o charakterze biurowym. Nie chodzi tylko o to, że edytor tekstu może czytać dane bezpośrednio z serwera WWW, czy zapisywać stworzony dokument w formacie HTML. Możliwe technicznie są rozwiązania, takie jak program finansowo-księgowy czytający kursy walut bezpośrednio z serwera banku, program dokonujący automatycznie uaktualnienia wersji (ang. upgrade) przez sieć, czy sprawdzający legalność użytkowania tą samą drogą. Interfejsem programowym umożliwiającym w prosty sposób dostęp do podstawowych usług internetowych jest Internet API. Interfejs ten ma wejść w skład podstawowego zestawu funkcji (Win32 API) następnych wersji systemów Microsoft Windows [1].

Istnieje również wiele komponentów, które pozwalają na korzystanie w programach użytkowych z technologii internetowych, np.: komponenty wyświetlające strony w języku HTML.

1.4. Architektura rozproszona zamiast klient-serwer

Stosowanie powyższych rozwiązań doprowadziło do sytuacji, w której klasyczna architektura klient-serwer została zastąpiona przez bardziej rozproszony model systemu. Wprawdzie w poszczególnych częściach składowych systemów dalej można wyróżnić klientów i serwery usług, ale taki system składa się z wielu części wzajemnie powiązanych taką relacją. Najprostszy przykład takiego systemu (3 poziomy) to sytuacja, gdy użytkownicy korzystają za pośrednictwem przeglądarki z serwera WWW, który jest klientem serwera bazy danych [3].

2. Technologia ActiveX

Zdaniem autorów zbiór metod i interfejsów programowych określanych przez firmę Microsoft jako ActiveX stanie się szybko podstawowym środowiskiem dla tworzenia oprogramowania w systemach Microsoft Windows, a być może również innych (UNIX). Należy tutaj zaznaczyć, że termin "Technologia ActiveX" ma raczej znaczenie marketingowe niż techniczne, lecz pod taką nazwą rozwiązania te są najszerzej znane, dlatego autorzy też używają tego terminu.

Technologia *ActiveX* jest oparta na COM[1] *(ang. Component Object Model)* - definicję obiektowego modelu tworzenia oprogramowania, niezależnego od narzędzi programistycznych (języków programowania), jest to rozwinięcie i uogólnienie wcześniejszych interfejsów: OLE, OLE2 i OLE Automation.

2.1. ActiveX na stacji roboczej

Technologia ActiveX na stacji roboczej to przede wszystkim ActiveX Controls (*.ocx*), używane dla rozszerzenia możliwości interfejsu użytkownika i przetwarzania danych na stronach WWW. Ponieważ jednak najnowsza wersja przeglądarki WWW firmy Microsoft (Internet Explorer 4.x) jest jednocześnie programem typu powłoka (*ang. shell*) w systemie Microsoft Windows, komponenty ActiveX mogą służyć do rozszerzania funkcjonalności interfejsu użytkownika w ogóle.

Jeżeli na stronie WWW znajduje się komponent, jest on automatycznie instalowany w systemie klienta w momencie ładowania tej strony z serwera WWW. Sposób takiej zdalnej instalacji (jego interfejs programowy) jest opisany w Microsoft Active Internet Platform. Interfejs ten obejmuje takie zagadnienia, jak:

a) transfer komponentów poprzez sieć (sposoby kompresji, łączenie komponentów w pakiety itp.),
b) instalacja w systemie klienta (bezpieczeństwo - certyfikaty autentyczności),
c) sprawdzanie licencji użytkownika.

Podstawowym problemem nie do końca rozwiązanym, jeżeli chodzi o używanie ActiveX Controls w sieci Internet, jest zapewnienie bezpieczeństwa. Komponent jest przecież programem, który może wykonywać dowolne operacje w pamięci operacyjnej i dyskowej komputera, na którym pracuje. Obecne rozwiązanie tego problemu polega na rejestrowaniu i wydawaniu certyfikatów na komponenty przez specjalnie wyznaczone serwery w sieci, przy czy wymagana jest świadoma współpraca użytkownika. O ile istnieje interfejs programowy zapewniający sprawdzanie certyfikatów, to strona organizacyjna pozostawia wiele do życzenia.

Powyższy problem w znikomym stopniu dotyczy sieci lokalnych, dlatego technologia ActiveX szczególnie dobrze pasuje do systemów sieci typu intranet. W takich sieciach problemy związane z możliwością nie kontrolowanego załadowania komponentu, który wykonuje jakieś niepożądane operacje, jest marginalna, a możliwość automatycznej instalacji i uaktualniania oprogramowania klienta ułatwia administrowanie wieloma stacjami.

---

[1] COM jest opisany w dalszej części opracowania.

Komponenty mogą być oczywiście używane to tworzenia aplikacji działających lokalnie na stacji roboczej i nie związanych w żaden sposób z Internetem czy przeglądarkami WWW.

2.2. ActiveX po stronie serwera

Specyfikacja Active Server Pages umożliwia umieszczanie na stronach WWW komponentów ActiveX, których kod jest wykonywany przez serwer (Internet Information Server), takie komponenty w odróżnieniu od komponentów wykonywanych przez przeglądarki nie mogą jednak posiadać reprezentacji graficznej. Możliwe za to jest stworzenie współpracujących par komponentów działających po stronie użytkownika i serwera.

Wraz z pojawieniem się Microsoft Transaction Server (MTP) zaistniała możliwość używania komponentów ActiveX do tworzenia rozproszonego oprogramowania oferującego szeroko rozumiane usługi, a nie tylko związane z serwerem WWW. MTS może współpracować z innymi serwerami np. SQL serwerem i zapewnia transakcyjne wykonywanie operacji, które są zdefiniowane przez programistów tworzących komponenty.

Programista piszący komponent nie musi rozwiązywać problemów synchronizacji procesów i zależności danych pomiędzy różnymi klientami. Komponent pracujący na MTS powinien obsługiwać jedną konkretną operację dla jednego klienta. Operacja taka może zakończyć się sukcesem lub nie powieść się, wtedy - zgodnie z regułami przetwarzania transakcyjnego - system przywracany jest do stanu sprzed operacji.

MTP zapewnia mechanizmy do zarządzania wieloma komponentami tworzącymi konkretne rozwiązanie oraz mechanizmy kontroli dostępu i praw użytkowników.

3. Component Object Model (COM)

COM jest to interfejs programowy do komunikacji i wymiany danych między procesami (*ang. InterProcess Communication - IPC* [2]) w środowisku Microsoft Windows. Interfejs ten wprowadza obiektowe metody tworzenia oprogramowania do systemu operacyjnego, umożliwiając tworzenie lepiej skonstruowanych i modularnych aplikacji niezależnie od użytych narzędzi programistycznych.

Komponent COM (ang. COM component) jest to program wykonywalny w środowisku systemów operacyjnych Microsoft Windows, zawarty w zbiorze dyskowym o rozszerzeniu .exe, .dll lub .ocx, który udostępnia innym programom obiekty możliwe do wykorzystania. Program, który używa komponentu lub komponentów, jest nazywany *klientem*. Klient może korzystać z komponentu w ten sposób, że tworzy obiekty będące wcieleniami *klas (ang. class)* zdefiniowanych w tym komponencie, a następnie wywołuje *metody (ang. method)*, używa

atrybutów (ang. properties) i otrzymuje *zdarzenia (ang. events)* wygenerowane z tak utworzonych obiektów.

Informacja o klasach zdefiniowanych w komponencie jest zawarta w *type library*. *Type library* jest to tekstowy opis klas w specjalnym języku przypominającym deklaracje w języku C++. W gotowym (skompilowanym) komponencie *type library* mieści się w zasobach *(ang. resource)* komponentu i klienci mają do niej dostęp w trakcie wykonywania programu. Odwołania do klas, metod, atrybutów i zdarzeń wykonywane są poprzez ich unikalne identyfikatory lub nazwy.

Każda klasa udostępniana przez komponent posiada co najmniej jeden *interfejs - interfejs domyślny*, w skład którego wchodzą atrybuty klasy i wybrane metody. Klasa może mieć zdefiniowanych więcej interfejsów. Interfejsy i klasy są rozpoznawane poprzez unikalny identyfikator *GUID (Globally Unique Identifier)*, który jest 128-bitową liczbą (16 bajtów) generowaną przez specjalny algorytm zdefiniowany w Open Software Foundation (OSF) Distributed Computing Environment (DCE) - zestawie standardów dla przetwarzania rozproszonego. Użycie klasy i interfejsu jest możliwe po zarejestrowaniu jej w rejestrze systemowym *(ang. registry)*.

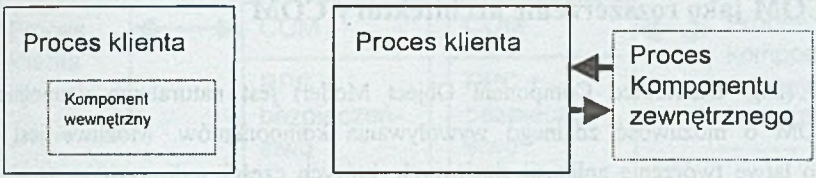
Taki sposób definicji interfejsu programowego klas umożliwia tworzenie nowych wersji komponentów, które jednocześnie zapewniają obsługę klientów znających jedynie niektóre (starsze) interfejsy komponentu (kompatybilność wstecz).

Komponenty możemy podzielić na dwie grupy w zależności od tego, w jaki sposób są związane z klientem:

a) Komponenty wewnętrzne (*ang. in process*) – takie, które współdzielą zasoby procesu klienta (.dll i .ocx);

b) Komponenty zewnętrzne *(ang. out of process)* – czyli takie, które są wykonywane w procesie oddzielnym od procesu klienta (.exe).

Komunikacja z komponentami wewnętrznymi jest najszybszą możliwą metodą (wywoływania funkcji i dostęp do zmiennych w tej samej przestrzeni adresowej), ale jest to okupione współdzieleniem jednego wątku i wszystkich zasobów programu przez klienta i komponent (co jest niekorzystne, jeśli chodzi o bezpieczeństwo danych i może być niekorzystne wydajnościowo w systemach wieloprocesorowych).



Rys. 1. Komponenty COM wewnętrzne i zewnętrzne
Fig. 1. In process and out of process COM components

Komponenty zawarte w zbiorach *.ocx* są to tzw. *COM Controls*, jest to taki rodzaj komponentu, który zwykle posiada graficzną reprezentację na ekranie, może obsługiwać interakcję z użytkownikiem i który może pracować jedynie wewnątrz przystosowanego do tego programu bazowego - tzw. *container application* (np. Microsoft Internet Explorer). Komponent taki zwykle sam rejestruje się w *rejestrze systemowym*.

Nie ma istotnych ograniczeń w zakresie tworzenia komponentów *COM*, w których skład wchodzą inne komponenty *COM*.

Komponenty COM można tworzyć korzystając z dowolnego narzędzia programistycznego umożliwiającego używanie wywołań systemowych systemu Windows 95/NT (Win32 API) i produkującego zbiory wynikowe .dll i .exe. Jest to jednak dosyć uciążliwa droga i w praktyce korzysta się z narzędzi ułatwiających ten proces. Istnieje stale rosnąca grupa oprogramowania, które ukrywa szczegóły implementacyjne interfejsu COM i pozwala programistom na proste tworzenie komponentów. W chwili obecnej są to:

a) Microsoft Visual Basic (wersja Visual Basic Custom Control Edition umożliwiająca kompilację do zbioru .ocx jest dostępna za darmo w sieci Internet - http://www.microsoft.com);
b) Microsoft Visula C++ z obiektowymi bibliotekami MFC (Microsoft Fundation Class) i ATL (Active Template Library);
c) Microsoft Visual J++ (Java);
d) Borland Delphi 3.

Należy się spodziewać, że w najbliższym czasie producenci języków programowania udostępnią również inne produkty.

4. DCOM jako rozszerzenie architektury COM

DCOM (ang. Distributed Component Object Model) jest naturalnym uzupełnieniem modelu COM o możliwość zdalnego wywoływania komponentów. Możliwe jest więc stosunkowo łatwe tworzenie aplikacji sieciowych, których części wykonywane na różnych komputerach mogą się ze sobą komunikować tak, jakby działały lokalnie. Uzyskano więc rozproszenie (*ang. distribution*) komponentów (czyli w zasadzie aplikacji) w sieci, dzięki czemu możliwe jest współdzielenie mocy obliczeniowej wielu komputerów.

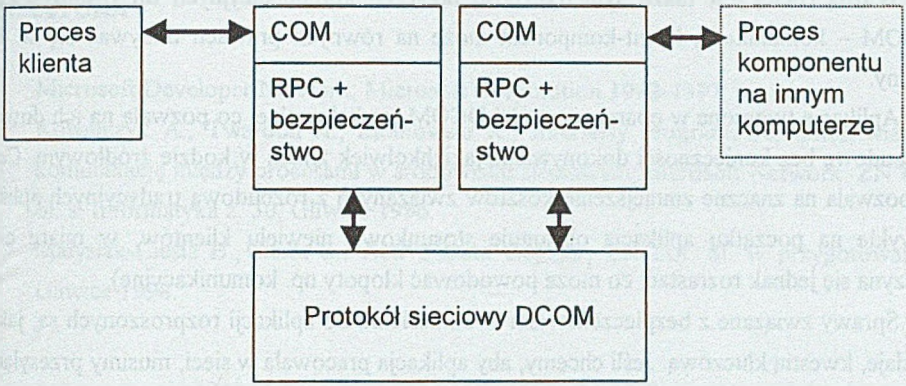
Istotne jest również to, że DCOM sam troszczy się o sprawy związane z niskopoziomową obsługą protokołów sieciowych (fakt rozproszenia aplikacji jest z punktu widzenia programisty przeźroczysty), pozwalając skupić się na tworzeniu aplikacji.

4.1. Architektura modelu DCOM

Ponieważ klient COM nie zna fizycznego położenia komponentu i odwołuje się do niego za pośrednictwem odpowiednich wpisów w rejestrze systemowym (*ang. registry*), nic nie stoi na przeszkodzie, aby komponent znajdował się naprawdę na innym komputerze w sieci.

Jak już powiedziano, DCOM w sposób naturalny i przeźroczysty dla samej aplikacji rozszerza model COM o elementy sieciowe – jeśli więc podczas komunikacji procesu klienta z procesem komponentu zewnętrznego okaże się, że obydwa procesy są realizowane na dwóch komputerach, to COM będzie w dalszym ciągu odpowiedzialny za stworzenie odpowiednich interfejsów, a DCOM zapewni im komunikację w ten sposób, że żaden z procesów nie będzie odczuwał faktu zmiany komunikacji lokalnej na sieciową (gdyby procesy były realizowane lokalnie, tzn. na tym samym komputerze, to również musiałyby się ze sobą w jakiś sposób komunikować).

Poniższy rysunek pokazuje w sposób schematyczny zmianę (wydłużenie się) drogi komunikacyjnej pomiędzy procesami klienta i komponentu w przypadku ich sieciowej realizacji. Należy tylko dodać, że w celu uzyskania dostępu do sieci interfejs COM, generując pakiety, które są zgodne z wymogami DCOM i mogą być przesyłane siecią, wykorzystuje technikę RPC (*ang. Remote Procedure Call*) [2] oraz odpowiednie mechanizmy bezpieczeństwa.



Rys. 2. Komunikacja procesów klienta i komponentu w realizacji sieciowej
Fig. 2. Client and component interprocess communication in network realization

4.2. Podstawowe zalety modelu DCOM

Chyba najistotniejszą zaletą modelu DCOM jest to, że pozwala on na dowolnie dalekie rozpraszanie tworzonych aplikacji bez konieczności wnikania w mechanizmy ich komunikacji. Co więcej, jeśli aplikacja korzystająca już ze standardu COM ma zostać rozproszona, to nie są wymagane jakiekolwiek zmiany jej kodu źródłowego (procesy klienta i komponentów w dalszym ciągu będą się komunikować) ani nawet ponowna jej kompilacja, a jedynie niezbędne zmiany interfejsu komunikacyjnego.

Kolejną zaletą podejścia DCOM w kontekście aplikacji rozproszonych (czasami pomiędzy wieloma krajami) jest ich niezależność językowa: ponieważ procesy komunikują się ze sobą poprzez ściśle określone interfejsy, wszystko to, co nie dotyczy sposobu komunikacji, może być tworzone w dowolnej wersji językowej, co znacznie obniża koszty, które w tradycyjnym podejściu są związane z tworzeniem różnych wersji językowych.

Model COM zakłada też bardzo efektywne wykorzystanie mechanizmów sieciowych min. poprzez utrzymywanie dla każdego komponentu odpowiedniego licznika, wskazującego na liczbę klientów, z którymi jest on w danej chwili połączony. Jeśli klient połączenie przerywa lub jeśli dzieje się to z innych przyczyn, np. awaria sieci[1], to licznik jest dekrementowany. W chwili gdy licznik osiągnie zero, komponent zwalnia zasoby.

[1] Aby wykrywać sytuacje wadliwego funkcjonowania sieci i wynikające z nich przerwy w łączności, DCOM monitoruje utrzymywane pomiędzy klientem a komponentem połączenia (przesyłane są pakiety kontrolne) i w razie ich zerwania licznik odpowiedniego komponentu jest dekrementowany.

Ważną cechą jest także symetryczność połączeń komunikacyjnych oferowanych przez DCOM – komunikacja klient-komponent może na równych prawach odbywać się w dwie strony.

Aplikacje tworzone w oparciu o model DCOM są skalowalne, co pozwala na ich dowolną rozbudowę bez konieczności dokonywania jakichkolwiek zmian w kodzie źródłowym. Cecha ta pozwala na znaczne zmniejszenie kosztów związanych z rozbudową tradycyjnych aplikacji (zwykle na początku aplikacja obsługuje stosunkowo niewielu klientów, w miarę czasu zaczyna się jednak rozrastać, co może powodować kłopoty np. komunikacyjne).

Sprawy związane z bezpieczeństwem w odniesieniu do aplikacji rozproszonych są, jak się wydaje, kwestią kluczową. Jeśli chcemy, aby aplikacja pracowała w sieci, musimy przesyłanym danym zapewnić całkowite bezpieczeństwo. Ponieważ w tej kwestii model DCOM został oparty na mechanizmach bezpieczeństwa proponowanych przez Open Software Foundation (OSF) Distributed Computing Environment (DCE), wydaje się, że będzie on spełniał odpowiednie wymagania.

W chwili obecnej dostępne są wersje DCOM dla środowisk Microsoft Windows NT 4.0, Microsoft Windows 95 oraz Sun Solaris, a wersje dla innych systemów operacyjnych są w przygotowaniu i prawdopodobnie będą dostępne w chwili ukazania się tego opracowania. Niewątpliwą wadą tego interfejsu jest mała (na razie) ilość narzędzi programistycznych umożliwiających jego przeźroczyste, dla twórcy komponentów, stosowanie (praktycznie tylko Visual Basic, inne narzędzia np. kompilatory C/C++ wymagają wiedzy nt. DCOM).

5. Podsumowanie

Interfejs COM w chwili obecnej jest już szeroko używany w wielu dziedzinach programowania w Microsoft Windows. Oparta na COM technologia ActiveX zdobywa coraz większą popularność w zastosowaniach internetowych i intranetowych. Wydaje się, że dominacja systemów Microsoft Windows jako środowiska użytkownika i coraz większa popularność Windows NT jako platformy serwera spowoduje, że rozwiązanie to będzie szerzej stosowane niż konkurencyjne komponenty w języku Java.

Interfejs DCOM nie jest jeszcze powszechnie stosowaną metodą, ale popularność COM wśród twórców oprogramowania na pewno spowoduje zwrócenie ich uwagi w kierunku tej technologii.

Chociaż elegancja interfejsu czy łatwość programowania są wielkościami niemierzalnymi, to autorzy opierając się na własnym doświadczeniu programistycznym uważają prezentowane tutaj metody za godne polecenia.

LITERATURA

[1] Microsoft Developer Network, Microsoft Corporation 1992-1997.

[2] Kowalczyk A., Twardoń A., Ziółkowski K.: Interfejsy programowe, umożliwiające komunikację między procesami w środowisku sieciowym Microsoft Network. ZN Pol. Śl. s. Informatyka z. 30, Gliwice 1996.

[3] Małysiak-Cieśla B., Cieśla S.: Active Data Objects, ZN Pol. Śl. w przygotowaniu, Gliwice 1998.

Recenzent: Dr inż. Ryszard Winiarczyk

Wpłynęło do Redakcji 24 grudnia 1997 r.

Abstract

The paper is an overview of program interfaces COM, DCOM and technology named ActiveX and they usage in application programming in Microsoft Windows and Internet environment.

Chapter one is the authors' view of present trends in application programming covering the following: LAN to WAN and WAN to LAN technology migration; HTML based applications; standalone applications with Internet access; evolution of client-server architecture into more distributed systems.

Next chapter describes possible usage of ActiveX technology on Internet/intranet workstation and server. It's remarked that ActiveX technology is more.

COM interface lying under ActiveX is described in next chapter. This is object model of software components cooperation, on operating system level. Differences of in process and out of process components are remarked (Fig. 1). Currently available software tools for component development are listed.

DCOM - the distributed version of COM is described in chapter four. DCOM architecture (Fig 2) and possible advantages of this interface are considered.

Last chapter contains conclusion that, in authors' opinion, described solutions are well done and that their usage will increase in near future.