STUDIA INFORMATICA 2002
Volume 23 Number 2A (48)

Dariusz Rafał AUGUSTYN
Politechnika Śląska, Instytut Informatyki

MECHANIZM AUTOMATYCZNEJ KONWERSJI DWUWARSTWOWEGO SYSTEMU RAPORTOWEGO DO SYSTEMU TRÓJWARSTWOWEGO OPARTEGO NA TECHNOLOGII DCOM

Streszczenie. W artykule przedstawiona została koncepcja modułu programowego, dokonującego automatycznej konwersji dwuwarstwowego systemu raportowego do systemu w architekturze trójwarstwowej. W proponowanym rozwiązaniu przetwarzanie raportowe aplikacji systemu o architekturze dwuwarstwowej zostało przeniesione do metod rozproszonych obiektów raportowych, zaimplementowanych w technologii komponentów DCOM, uruchamianych w wybranych węzłach lokalnej sieci komputerowej.

THE MECHANISM OF AUTOMATIC CONVERSION FROM A TWO TIER-REPORT SYSTEM TO A THREE-TIER SYSTEM BASED ON DCOM TECHNOLOGY

Summary. This article presents an idea of a program module for the automatic conversion from a two-tier report system to a three-tier one. Report processing of applications based on the client-server architecture was moved into methods of distributed report objects implemented in DCOM technology activated in selected nodes of LAN.

1. Wstęp

Jednym z podstawowych zadań wielu systemów informatycznych jest wykonywanie raportów i zestawień w oparciu o zawartość bazy danych. Narzędzia tworzenia aplikacji pozwalają na tworzenie modułów programowych, służących do raportowania. Przykładem

takiego narzędzia programowego jest system SQLWindows/32 firmy Centura Software®. Tego typu narzędzia tworzenia aplikacji przeznaczone są głównie do wytwarzania systemów informatycznych działających w oparciu o architekturę klient-serwer i pracujących w sieciach lokalnych, gdzie zakłada się stosunkowo dużą przepustowość łącza. Dotyczy to również wytwarzanych w tych narzędziach modułów raportowych, działających właśnie w architekturze dwuwarstwowej.

Architektura klient-serwer staje się nieefektywna w sytuacji występowania znacznej ilości danych potrzebnych do realizacji raportu, pobieranych od oddalonego serwera bazy danych i transmitowanych przez sieć do aplikacji klienta. Rozwiązaniem problemu może być stworzenie systemu o architekturze trójwarstwowej. W takim systemie, dodatkowa, druga warstwa oprogramowania zajmuje się pobieraniem danych od serwera bazy danych, czyli warstwy pierwszej, dokonywaniem przeliczeń i agregacji danych oraz transmisji danych do aplikacji klienta, czyli warstwy trzeciej. W trójstopniowym przetwarzaniu możliwa jest konfiguracja systemu, w której oprogramowanie warstwy pierwszej - serwer bazy danych i warstwy drugiej - serwer raportowy uruchomione są na tym samym komputerze. Dzięki temu komunikacja, w której następuje intensywna transmisja danych, może być zrealizowana poprzez szybkie mechanizmy wewnątrzsystemowe (IPC - ang. interprocess communication), jak np. anonimowe potoki czy pamięć dzielona. Przez sieć komputerową do raportującej aplikacji klienta, której rola sprowadza się do prezentacji danych, transmitowane są jedynie dane o małym rozmiarze. Taka organizacja przetwarzania jest efektywna nawet dla przeciążonych sieci komputerowych bądź sieci o małych przepustowościach łącza.

Artykuł prezentuje programowy mechanizm umożliwiający automatyczną (prawie bezobsługową) konwersję aplikacji raportowych, działających w architekturze klient-serwer i wykonanych w SQLWindows/32, na aplikacje działające w architekturze trójwarstwowej. Mechanizm tworzenia dodatkowej warstwy oprogramowania przetwarzającego opiera się na technologii COM/DCOM (DCOM - ang. distributed component object model) [2]. Polega on na automatycznym rozdziale kodu klasycznej aplikacji raportowej (wykonanej w środowisku SQLWindows/32) na dwie niezależne części: część będącą klientem obiektu DCOM oraz część stanowiącą serwer DCOM, w którym odbywać się będzie zasadnicze przetwarzanie danych.

Prezentowane rozwiązanie pozwala na taką konwersję istniejącego dwuwarstwowego systemu raportowego, aby umożliwić użycie programu niezależnego technologicznie od pozostałych warstw przetwarzania raportowego w roli aplikacji raportującej. W proponowanym rozwiązaniu obiekty raportowe zaimplementowane są w języku SAL systemu SQLWindows/32, ze względu na fakt, że ich kod pochodzi z istniejącej wcześniej aplikacji, utworzonej w SQLWindows/32. Jednak dzięki zastosowaniu otwartego, binarnego

interfejsu usług obiektów raportowych DCOM, możliwa jest realizacja zadań raportowych poprzez aplikacje raportujące (warstwa trzecia), które nie są wykonane w środowisku SQLWindows/32. Właśnie ta otwartość interfejsu obiektów DCOM pozwala na uzyskanie uniwersalności proponowanego rozwiązania w stosunku do istniejącego rozwiązania w architekturze klient-serwer, gdzie prezentacja raportu i logika przetwarzania raportowego były zintegrowane i zaimplementowanie w tym samym języku programowania. W nowym rozwiązaniu praktycznie każda aplikacja, mogąca wykorzystać interfejs COM może się stać klientem zdalnych lub lokalnych obiektów raportowych.

Dla efektywnego zarządzania systemem raportowym stworzony został program administratora systemu raportowego, umożliwiający zmianę lokalizacji uruchomienia obiektów raportowych. Program administratora umożliwia z jednego miejsca (tzn. dowolnego węzła sieci komputerowej) realizację decyzji użytkownika, w którym węźle sieci będzie uruchamiany obiekt DCOM, będący serwerem raportowym dla wskazanej aplikacji klienta (klienta DCOM). Może być to dowolny węzeł sieci, gdzie zarejestrowany i udostępniony jest serwer raportowy w postaci obiektu DCOM. W szczególności może to być stacja robocza z uruchomioną aplikacją klienta (warstwa trzecia) albo komputer z uruchomionym serwerem bazy danych (warstwa pierwsza). Technicznie możliwe jest to dzięki programowej, zdalnej modyfikacji udostępnianych rejestrów systemu Windows 9x/2000/XP na stacjach roboczych (tam, gdzie uruchomiona jest aplikacja raportująca - warstwa trzecia).

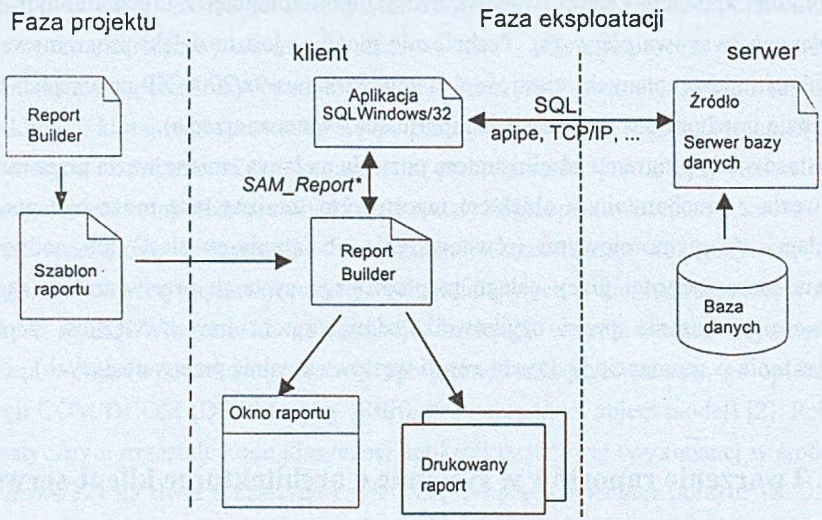
Zastosowanie programu administratora pozwala na łatwą zmianę węzła przetwarzającego, czyli węzła z uruchomionym obiektem raportowym. Zmiana taka może być podyktowana względami efektywnościowymi (równoważenie obciążenia w sieci) lub podwyższeniem poziomu niezawodności pracy całego rozproszonego systemu przetwarzania (zapewnienie możliwości wskazania przez użytkownika-administratora innych węzłów, zdolnych do przetwarzania w momencie wykrycia awarii węzłów aktualnie przetwarzających).

2. Tworzenie raportów w systemie o architekturze klient-serwer

W środowisku SQLWindows/32 podsystemem przeznaczonym do tworzenia raportów jest *Report Builder*. Umożliwia on projektowanie układu graficznego prezentowanych danych, ich wyświetlanie oraz drukowanie. Podsystem ten wykorzystywany jest zarówno w fazie projektu układu graficznego raportu przy tworzeniu aplikacji, jak również w fazie eksploatacji wdrożonej aplikacji. Rys. 1 przedstawia środowisko podsystemu raportowego SQLWindows/32 w obu tych fazach. *Report Builder* jest nie tylko narzędziem projektowania

raportu w trybie WYSIWYG [3], ale stanowi niezbędną część środowiska uruchomieniowego dla aplikacji tworzonych w SQLWindows/32.

W projekcie raportu definiuje się układ pól związanych z poszczególnymi danymi, określa się format prezentacji, definiuje proste funkcje operujące na danych (np. agregacje, sumowania, grupowanie). Projekt raportu przechowuje się w pliku w formie tzw. szablonu. Projekt raportu nie musi być skojarzony z żadnym źródłem danych. Wymiana danych pomiędzy raportem a źródłem danych (na ogół serwerem bazy danych) zachodzi za pośrednictwem aplikacji SQLWindows/32. W szablonie deklarowane są zmienne wejściowe, tzw. *input items*, poprzez które aplikacja dostarcza wartości dla wyświetlanych/drukowanych wierszy raportu. Aplikacja SQLWindows/32 komunikuje się z podsystemem raportowym za pomocą rodziny funkcji *SalReport**. Uruchomienie podsystemu raportowego następuje przez wywołanie funkcji *SalReportView()* lub *SalReportPrint()*. Jako parametry przyjmują one nazwy *input items* z szablonu raportu oraz odpowiadające im nazwy zmiennych w przestrzeni adresowej aplikacji SQLWindows/32. Aplikacja pełni rolę "serwera danych", udostępniającego podsystemowi raportowemu informacje, np. z bazy danych.



Rys. 1. *Report Builder* w fazie projektu raportu i fazie wdrożonego systemu, działającego w architekturze klient serwer

Fig. 1. *Report Builder* in project phase and in deployment phase of a system based on the client-server architecture

Podsystem *Report Builder* zgłasza zapotrzebowanie na dane za pomocą grupy komunikatów systemu MS Windows nazwanych *SAM_Report**: *SAM_ReportStart, SAM_ReportFetchInit, SAM_ReportFetchNext, SAM_ReportFinish*. Komunikaty kierowane

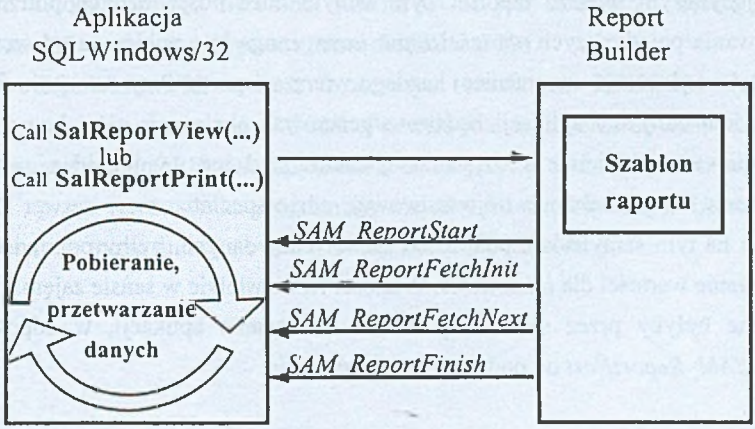
są do wskazanego okna aplikacji. Za pomocą tych komunikatów *Report Bulider* informuje aplikacje o zdarzeniach związanych z bieżącą obsługą raportu typu: rozpoczęcie/zakończenie inicjalizacji raportu, przejście na nową stronę, rozpoczęcie/zakończenie grupy danych itp. W ramach obsługi komunikatów, aplikacja może zmieniać zawartość własnych zmiennych, skojarzonych z *input items* raportu, czy poinformować *Report Buildera* o zakończeniu zbioru danych do uwzględnienia w raporcie.

Po poprawnym zainicjowaniu podsystemu raportowego, aplikacja otrzymuje komunikat *SAM_ReportStart*. W ramach obsługi tego komunikatu aplikacja powinna zainicjować źródło danych.

Następnie aplikacja otrzymuje komunikat *SAM_ReportFetchInit*, w ramach obsługi którego powinna wykonać inicjalizację potrzebnych zmiennych oraz zlecić do źródła danych ustawienie wskaźnika na początek zbioru danych.

Podsystem *Report Bulider*, tworząc nowe wiersze raportu (zbudowane z *input items*), zgłasza żądanie pobrania kolejnych danych poprzez sekwencyjne wysyłanie komunikatu *SAM_ReportFetchNext*. Aplikacja pobiera dane ze źródła i ustawia nowe wartości poszczególnych zmiennych, skojarzonych z *input items*. Jeżeli aplikacja odpowie wartością *false* w ramach synchronicznej obsługi komunikatu, oznaczać to będzie dla podsystemu raportowego, że zakończył się zbiór danych do raportowania.

W ramach zakończenia procesu tworzenia raportu jednorazowo wysyłany jest komunikat *SAM_ReportFinish*. W odpowiedzi aplikacja deinicjalizuje źródło danych, zwalnia zasoby itp.



Rys. 2. Komunikacja aplikacji SQLWindows/32 z podsystemem *Report Builder*
Fig. 2. Communication between a SQLWindows/32 application and the subsystem *Report Builder*

3. Motywacja

Opisany w rozdziale 2 model przetwarzania danych przy tworzeniu raportu [1], pokazany na rys. 1 i 2, jest efektywny dla typowego scenariusza działań, w którym raport wykonywany jest w oparciu o wynik realizacji jednego zapytania SQL. Wtedy, w ramach obsługi *SAM_ReportStart*, ma miejsce realizacja połączenia z bazą danych oraz przesłanie z aplikacji do serwera bazy danych treści zapytania SQL oraz przygotowanie zapytania (faza *prepare* w przetwarzaniu po stronie serwera bazy danych [3]). W ramach obsługi *SAM_ReportInit* następuje przesłanie rozkazu wykonania wcześniej przygotowanego zapytania SQL, czyli wysyłany jest rozkaz tworzenia wynikowego zbioru wierszy, spełniających kryteria zapytania (faza *execute* [3]). W ramach obsługi każdego komunikatu *SAM_ReportFetchNext* następuje pobranie kolejnego wiersza z utworzonego po stronie serwera bazy danych zbioru wynikowego dla zadanego zapytania SQL. Wartości pobranego wiersza ze zbioru wynikowego odpowiadają (lub prawie odpowiadają, tzn. są łatwo wyliczalne) wartościom wiersza sporządzanego raportu, czyli stanowią wartości wprost dla *input items*.

Jednak często zdarza się, że nie można oprzeć przetwarzania na jednym zapytaniu SQL, którego wartości w kolumnach wierszy wynikowych niemal odpowiadają wartościom *input items* raportu. Dla złożonych raportów zapytania SQL (może być ich kilka) są wielokrotnie zlecane dla każdego komunikatu *SAM_ReportFetchNext*, a więc dla każdego wiersza raportu (tzn. nie tylko jednorazowo dla *SAM_ReportInit*). Dopiero w oparciu o wyniki realizacji tych zapytań (tzn. po przetworzeniu wielu wierszy wynikowych) wyznaczane są wartości *input items* w pojedynczym wierszu raportu. Tym samym duże ilości danych, potrzebnych do wypracowywania pojedynczych wartości *input items*, mogą być pobierane od serwera bazy danych do aplikacji wraz z utworzeniem każdego wiersza raportu. Przy założeniu fizycznego oddalenia źródła danych i aplikacji będzie to generować obciążenie sieci komputerowej i może powodować opóźnienia w działaniu systemu. Stąd też wynika idea zastosowania oprogramowania w architekturze trójwarstwowej, gdzie specjalizowany serwer raportowy, uruchomiony na tym samym komputerze co serwer bazy danych, byłby odpowiedzialny za wypracowywanie wartości dla *input items*. Wartości te, niewielkie w sensie zajętości pamięci, transmitowane byłyby przez sieć komputerową na żądanie aplikacji, w odpowiedzi na komunikaty *SAM_ReportNext* od podsystemu raportowego.

4. Wykorzystanie technologii COM w środowisku SQLWindows/32

Serwer raportowy jest zbiorem niezależnych obiektów raportowych wykonanych w technologii COM [2]. COM definiuje binarny standard udostępnianych obiektów w postaci tzw. komponentów. Technologia COM działa w oparciu o model klient-serwer. Klientem jest każda aplikacja, która za pośrednictwem interfejsu potrafi korzystać z komponentów COM.

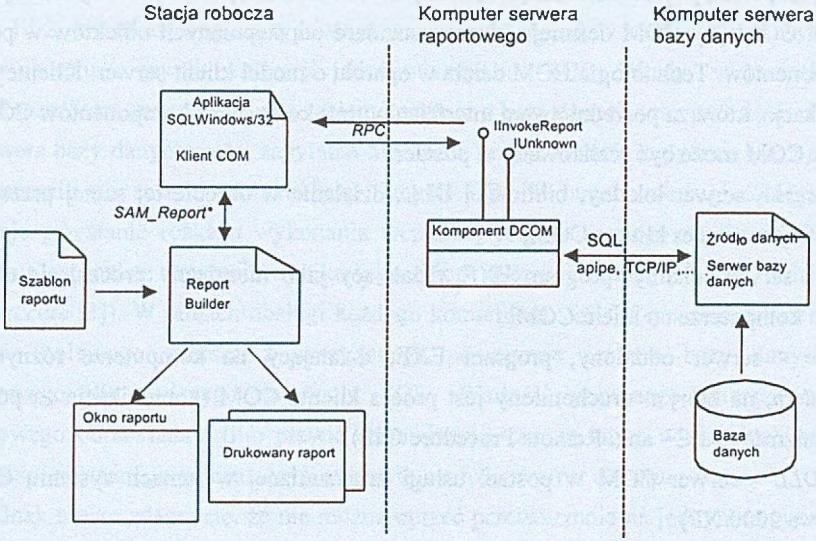
Serwer COM może być realizowany w postaci:

- *In-process* - serwer lokalny, biblioteka DLL, działanie w obrębie tej samej przestrzeni adresowej co proces klienta COM,
- *Local* - serwer lokalny, program EXE, działający jako niezależny proces, ale na tym samym komputerze co klient COM,
- *Remote* - serwer oddalony, program EXE, działający na komputerze różnym od komputera, na którym uruchomiony jest proces klienta COM (komunikacja za pomocą mechanizmów RPC - ang. Remote Procedure Call),
- *MTS DLL* - serwer COM w postaci usługi uruchamianej w ramach systemu COM+ (Widows 2000/XP).

System SQLWindows/32 (korzystając z danych o komponencie COM, umieszczonych w rejestrze systemowym i plikach **.tlb*, opisujących interfejs [2]) potrafi utworzyć odpowiednie klasy w języku SAL, powalające na tworzenie, wywoływanie metod, niszczenie obiektów COM. Klasy te stanowią jedynie wygodne opakowanie dla wywołań kierowanych do lokalnych lub zdalnych obiektów COM (wykorzystują interfejs klienta automatyzacji OLE). Mechanizm automatycznego tworzenia klienta COM jest integralną częścią środowiska SQLWindows/32. Jest to tzw. kreator klienta obiektów COM.

System SQLWidows/32 można rozszerzyć o moduł XSalCOM (niezależnego producenta), tzw. kreator serwera COM. Pozwala on na tworzenie komponentu COM (serwera automatyzacji OLE) na podstawie kodu w języku SAL dla wskazanej klasy (dla wybranych przez użytkownika-programistę metod i własności klasy). Utworzony przez XSalCOM serwer może funkcjonować w wersji *Local* lub *Remote*.

5. Tworzenie systemu raportowego w architekturze trójwarstwowej, działającego w oparciu o technologię DCOM



Rys. 3. Trójwarstwowy model systemu raportowego w technologii DCOM
Fig. 3. The three-tier architecture model of a report system in DCOM technology

Idea działania trójwarstwowego systemu raportowego pokazana została na rys. 3. Mechanizm automatycznego tworzenia systemu działającego według powyższego schematu jest realizowany w następujących krokach:

a) **Pobranie do środowiska SQLWindows/32 kodu aplikacji, działającej na zasadzie modelu klient-serwer.**

b) **Uruchomienie kreatora tworzącego kody źródłowe klienta i serwera obiektu raportowego.**

Przed uruchomieniem kreatora programista wskazuje okno aplikacji, uwrażliwione na komunikaty *SAM_Report**.

W wyniku działania kreatora powstają dwa niezależne pliki z kodami źródłowymi programów w języku SAL systemu SQLWindows/32.

Kreator klasy obiektu raportowego ze zmiennych aplikacji, odpowiadających zmiennym *input items*, tworzy właściwości tej klasy (*Properties*), a akcje zawarte w ramach obsługi każdego z czterech komunikatów *SAM_Report** umieszcza wewnątrz tworzonych metod *InvokeReport**. Kreator automatycznie dodaje do ciała metody

InvokeReportStart/InvokeReportFinish kod odpowiedzialny za połączenie/rozłączenie ze źródłem danych.

Kreator klienta aplikacji COM usuwa oryginalny kod akcji wykonywanych w ramach obsługi komunikatów *SAM_Report**. W ramach obsługi komunikatów *SAM_ReportStart* i *SAM_ReportFinish* wpisane zostają komendy SAL tworzące lub niszczące odpowiedni obiekt COM. Akcja związana z obsługą *SAM_ReporFetchNext* zredukowana zostaje do ustawienia zmiennych aplikacji poprzez pobranie wartości właściwości obiektu COM (wywołania metod *GetProperty**) oraz odpowiednie wywołanie metody *InvokeReportFetchNext* tego obiektu.

c) **Utworzenie komponentu COM dla wskazanej klasy obiektu serwera raportowego** na podstawie kodów źródłowego serwera raportowego, poprzez wywołanie kreatora XSalCOM

W wyniku działania XSalCOM, operującego na kodzie klasy utorzonej w punkcie *b*, powstaje samorejestrujący się komponent w postaci programu EXE. Utworzony komponent COM udostępnia interfejs *IInvokeReport* (z metodami: *GetProperty** i *InvokeReport**), którego wykorzystanie ilustruje rys. 3.

Komponent taki można łatwo rejestrować jako obiekt COM lub DCOM na dowolnym komputerze z systemem Windows 9x/2000/XP i dystrybuować go niezależnie od aplikacji klienta (ale wraz ze środowiskiem uruchomieniowym, dostarczanym z SQLWindows/32).

Realizacja automatycznego tworzenia kodu źródłowego klienta i serwera COM z punktu *b* możliwa jest dzięki zastosowaniu pakietu CDK (ang. Component Development Kit), umożliwiającego programowe sprawowanie kontroli nad działaniem środowiska SQLWindows/32.

Pakiet CDK stanowiący opcjonalne rozszerzenie SQLWindows/32 jest obiektowo zorientowaną biblioteką, udostępniającą:

- interfejs *Outline* (zestaw klas i metod do manipulacji na hierarchicznym kodzie źródłowym programu w języku SAL systemu SQLWindows/32),
- interfejs *Runtime Inspektor* (zestaw klas i metod określających stan uruchomionej aplikacji, statusy ekranowych elementów kontrolnych, pozwalających na tworzenie własnych programów do testowania),
- interfejs *Notification* (zestaw klas i metod dostarczających informacji o zdarzeniach generowanych przez system SQLWindows/32 podczas edycji kodu źródłowego przez użytkownika-programistę).

Program kreatora kodów klienta i serwera COM został napisany w języku SAL z wykorzystaniem metod interfejsu *Outline* pakietu CDK.

6. Administrowanie obiektami raportowymi

Komponenty raportowe DCOM mogą być umieszczone i zarejestrowane na stacjach roboczych, czyli tam, gdzie znajdują się aplikacje klienta, jak również w innych, wybranych węzłach sieci lokalnej (w szczególności na komputerze z uruchomionym serwerem bazy danych). Obecność komponentu na stacji roboczej pozwala na jego wykorzystanie w trybie lokalnego serwera COM.

W celu konfiguracji komponentu w trybie serwera zdalnego DCOM wykorzystany jest program systemowy *DCOMCnfg*. Pozwala na ustawienie kluczy rejestrów związanych z konfiguracją klienta DCOM, jak i serwera DCOM.

W celu konfiguracji klienta DCOM [2, 5] na stacji roboczej program *DCOMCnfg* umożliwia ustalenie nazwy zdalnego komputera z serwerem DCOM, do którego kierowane będą potem odwołania klienta. Program pozwala na ustawienie automatycznego żądania ze strony aplikacji klienta aktywacji instancji serwera DCOM (klucz *ActivateAtStorage*).

W konfiguracji serwera DCOM istotne jest ustawienie odpowiedniego poziomu bezpieczeństwa [2, 5] poprzez określenie:

- użytkowników (lista *ACL*), którzy mogą uruchomić proces serwera DCOM (klucz *LaunchPermission*),
- użytkownika (identyfikator *SID*), z prawami którego serwer DCOM będzie uruchomiony (klucz *RunAs*),
- użytkowników (lista *ACL*) mogących uzyskać dostęp do usług uruchomionego serwera DCOM (klucz *AccessPermission*).

W proponowanym rozwiązaniu systemu rozproszonego zakłada się węzły sieci mogące funkcjonować w roli serwerów raportowych są konfigurowane lokalnie, jednorazowo (z wykorzystaniem programów *DCOMCnfg* i *Regedit* albo *Regedt32*, który dodatkowo uwzględnia system uprawnień dostępu do poszczególnych gałęzi rejestrów systemu Windows 2000/XP).

Natomiast wartości w kluczach rejestrów stacji roboczych, dotyczące klientów DCOM, mogą być wielokrotnie, dowolnie zmieniane w trakcie eksploatacji systemu z jednego, wybranego węzła sieci. Możliwość centralnego zarządzania systemem rozproszonych obiektów raportowych sprowadza się głównie do zmiany wpisu o lokalizacji komponentu DCOM. Takie centralne zarządzanie staje się możliwe do realizacji po zainstalowaniu na stacjach roboczych usługi *Remote Registry service*, włączeniu opcji zdalnego administrowania i określeniu listy użytkowników, mających prawa do zamiany wartości kluczy w odpowiednich gałęziach rejestru.

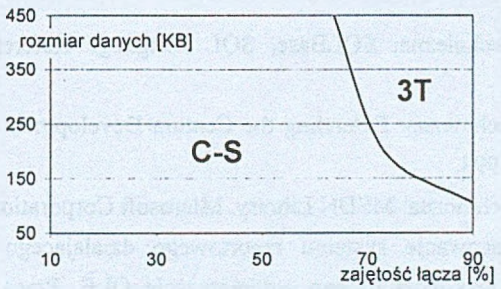
Dla ułatwienia zarządzania systemem raportowym stworzono specjalizowany program administratora. Program pozwala na logowanie do wybranego komputera (stacji roboczej), przegląd wpisów o własnościach zainstalowanych komponentów DCOM, z wyróżnieniem komponentów raportowych oraz edycję niektórych kluczy rejestru, związanych z wybranym komponentem. Podstawowa funkcja programu to zmiana lokalizacji uruchomienia komponentów raportowych, realizowana poprzez zdalną modyfikację klucza *RemoteServerName*, zawierającego nawę przetwarzającego węzła sieci w konwencji UNC, DNS lub adresu IP.

7. Zakończenie

W ramach projektu opisanego w niniejszym artykule stworzono narzędzia programowe przeznaczone do generacji kodu klienta i serwera raportowego oraz do administracji lokalizacją uruchomienia obiektów raportowych.

Systemy programowe zostały wykonane i przetestowane z wykorzystaniem środowiska programowego SQLWindows/32 v.1.5, Microsoft Visual C++ v.6.0 i serwera bazy danych SQLBase v.7.5. Część modułów programowych kreatora kodu źródłowego klienta i serwera COM oraz administratora komponentów DCOM wykonana została w ramach pracy dyplomowej [6].

W ramach projektu przeprowadzono testy, w których dokonano pomiarów czasów wykonania raportów, działających w systemie o architekturze klient-serwer (C-S) i trójwarstwowej (3T). Wyniki testów w postaci syntetycznej pokazano na rys. 4.



Rys. 4. Obszary efektywności dla architektury C-S i 3T w zależności od rozmiaru przesyłanych danych [KB] i współczynnika zajętości łącza [%] dla sieci Ethernet 100 Mb/s

Fig. 4. Areas of effectivity for the C-S and the 3T architecture depended on a size of transmitted data [KB] and an utilization coefficient [%] for Ethernet 100 Mb/s

Testy przeprowadzone zostały w sieci lokalnej Ethernet 100 Mb/s. Dokonano w nich symulacji obciążenia (mierzonego współczynnikiem zajętości łącza [%]), poprzez wprowadzanie kontrolowanej liczby pakietów zakłócających. Testy sytemu w architekturze C-S i 3T wykonano dla różnych raportów, których realizacja w architekturze C-S wymagała transmisji przez sieć do aplikacji klienta różnej ilości danych (o różnym rozmiarze [KB]).

Krzywa z rys. 4 wyznacza granicę efektywności stosowania rozwiązania w architekturze C-S albo 3T. Granica ta wyznacza obszary efektywności dla C-S albo 3T, tzn. takie zbiory par wartości (zajętość, rozmiar), dla których raport wykonywany jest szybciej w rozwiązaniu C-S albo 3T.

Rys. 4 pokazuje, że rozwiązanie C-S dla małych raportów, w sensie rozmiaru przesyłanych danych (poniżej 100KB w rozwiązaniu bazującym ma architekturze C-S), jest zdecydowanie lepsze dla każdej przepustowości łącza. Jednak dla zajętości łącza na poziomie 90%, odpowiadającej w przybliżeniu nieobciążonej sieci 10 Mb/s, rozwiązanie 3T dla raportów wymagających przesyłania powyżej 100 KB, staje się konkurencyjne w sensie czasu wykonania. Dla sieci o jeszcze niższych przepustowościach lub w sytuacji większych rozmiarów przesyłanych danych rozwiązanie 3T może być efektywniejsze od rozwiązania C-S.

LITERATURA

1. Ring B.: Developing with SQLWindows/32. Centura Software Corp., 1998.
2. Eddon G., Eddon H.: Inside Distributed COM. Mirosoft Press, Redmond, Washington 1998.
3. Dokumentacja techniczna: SQLBase, SQL Language Reference. Centura Sofware Corp., 1999.
4. Dokumentacja techniczna: Extending the Centura Development Environment. Centura Sofware Corp., 1999.
5. Dokumentacja techniczna: MSDN Library. Microsoft Corporation, 2001.
6. Kopek Z.: Opracowanie systemu raportowego działającego w oparciu o model COM/DCOM i wykorzystującego automatyzację OLE. Praca dyplomowa, Wydział Automatyki, Elektroniki i Informatyki Politechnika Śląska w Gliwicach, kierunek Informatyka, 2001.

Recenzent: Dr inż. Arkadiusz Sochan

Wpłynęło do Redakcji 11 kwietnia 2002 r.

Abstract

This paper presents a program module for an automatic conversion from a report system based on the client-server architecture (C-S) to a report system based on the three-tier architecture (3T). A report server is an additional data processing tier. In 3T systems, data processing is moved form a thick client application of C-S systems to bodies of report object's methods. A set of distributed report objects is the report server. Report objects are implemented as DCOM components and could be activated in any node of LAN.

The paper presents an idea of an administrator program, which could be run in any LAN node for a remote change of network location of activated report objects.

The purposed solutions are dedicated for application systems, which were developed using the SQLWindows/32 software environment.

The paper presents such conditions (values of network utilization and size of transmitted report data) that the 3T architecture is more efficient than the C-S one (has a better time of response).