УДК 004.451.56

© 2005р. С.В. Мельничук, В.І. Федорук, П.М. Благодір

Чернівецький національний університет ім. Ю.Федьковича, Чернівці

СИСТЕМА УНІВЕРСАЛЬНОГО ДОСТУПУ ДО WINDOWS-АПЛІКАЦІЙ

Розроблено універсальний інтерфейс доступу до екранних зображень, який дозволяє з мінімальними затратами керувати аплікаціями, звертаючись до елементів їх графічного інтерфейсу. Проаналізовано існуючі підходи до створення такого інтерфейсу та проблеми, які виникають за його використанняі. Розроблена програмна архітектура системи, реалізоване її ядро.

The universal interface for access to the screen images, which allows with minimum expenses to manage appliqués, is developed, addressing the elements of their graphic interface. Analyzed existent approaches to creation of such interface, problems that arise up at his use. Program architecture of the system is developed, its kernel is realized.

На даний час у світі інформатики важливим є стан програмного забезпечення комп'ютерних систем. Старі системи вже не можуть забезпечити потреб користувачів. Користувачі вимагають більше зручностей, сервісу, можливостей. Постає проблема оновлення програмного забезпечення, яка дуже часто розв'язується шляхом простого переписування систем із врахуванням нових вимог. Такий підхід вимагає величезних затрат коштів і часу.

Інша проблема полягає в тому, що робота із програмним забезпеченням дуже часто вимагає великої кількості рутинних операцій, які повинна виконувати людина. Одна з вимог до сучасних програмних комплексів – забезпечення зручності й простоти роботи, автоматизації рутинних операцій.

До нових напрямків на ділянці такого програмного забезпечення належить розробка системи діагностики, тестування, відлагодження і автоматизації програмного забезпечення. Цей напрямок досить молодий і має великі перспективи. У зв'язку із цим важливою є задача розробки інтерфейсу універсального доступу до екранних зображень. Такий інтерфейс дозволить із мінімальними затратами керувати аплікаціями, звертаючись до елементів їх графічного інтерфейсу.

Серед систем, призначених для автоматизації та інтеграції аплікацій, найбільш сучасна система *Automate* 5.0. Ція система має ряд переваг перед системами, призначених для автоматизації та інтеграції аплікацій:

- добре розроблений інтерфейс;
- оновлення версій разом з системою;
- широкі можливості застосування.

Водночас їй притаманні такі гнедоліки:

- висока собівартість системи;
- відсутність засобів віддаленого доступу.

Реалізація доступу до екранних зображень базується на тому, що всі компоненти й елементи операційної системи *Windows* мають асоційований з ними дескриптор, використовуючи який можна отримати і задати певну інформацію для системи. Дескриптор є цілим числом і саме він, і тільки він, однозначно визначає елемент операційної системи: вікно, елемент графічного інтерфейсу, повідомлення, файл, або пристрій введення/виведення.

Використовуючи дескриптор, можна відіслати повідомлення у чергу повідомлень *Windows*, зчитати або встановити дані, виконати підтримувані дії (наприклад, емулювати натискання кнопки, посилаючи повідомлення *WM_LBUTTON DOWN* у чергу повідомлень). Дескриптор сталий лише під час існування елемента операційної системи і генерується при його створенні. Дескриптор кнопки буде іншим, якщо перезапустити вікно. У зв'язку з цим постає проблема пошуку елемента у процесі роботи. Дескриптор не може бути критерієм порівняння під час пошуку, він – лише засіб доступу до елемента. Тому необхідно обрати інші ознаки (критерії) для пошуку елементів в операційній системі. Такими критеріями можуть бути:

- заголовок (наприклад, статичний текст "*Ok*" на кнопці діалогу);
- клас елемента (може бути лише допоміжним критерієм, оскільки багато різних елементів можуть мати однаковий клас);
- розміри (можливі проблеми для елементів з однаковим розміром);
- відносна позиція елемента (можливі проблеми, наприклад, при переміщенні вікон на екрані);
- дочірні елементи (елемент може бути контейнером для дочірніх елементів).

Спрощена схема взаємодії компонент системи наведена на рис. 1.

Доступ до графічного інтерфейсу користувача оригінальної аплікації забезпечує бібліотека проксі-компонент, *COM*-інтерфейси якої зображені на рис. 2. З об'єктів цієї бібліотеки в режимі дизайну будується модель графічного інтерфейсу оригінальної аплікації.

Для надсилання даних елемента графічного інтерфейсу оригінальної аплікації використовується система повідомлень операційної системи, а також низькорівневі *API*-функції.

Отже, очевидно, що для пошуку елементів графічного інтерфейсу повинен використовуватись досить продуманий алгоритм пошуку і розпізнавання. Жоден з перелічених вище критеріїв не є достатнім для однозначної ідентифікації. Достатньою може бути лише певна їх комбінація. Більше того, в деяких ситуаціях певні критерії можуть зашкодити успішному розпізнаванню елемента (наприклад, позиція і розмір елемента за різного налаштування операційної системи).

Нами розроблена система, яка дозволяє виділяти складові частини графічного інтерфейсу користувача оригінальної аплікації, емулювати роботу з аплікацією за допомогою скриптів автоматизації. Головними можливостями програмного продукту є:

- автоматизація аплікацій за допомогою скриптів;
- створення "мостів" між аплікаціями для організації передачі даних;
- розширення функціональних можливостей аплікацій із закритим вихідним кодом;
- редагування *GUI* (графічного інтерфейсу користувача) *Windows* аплікацій;
- автоматизація процесу тестування графічного інтерфейсу аплікацій.

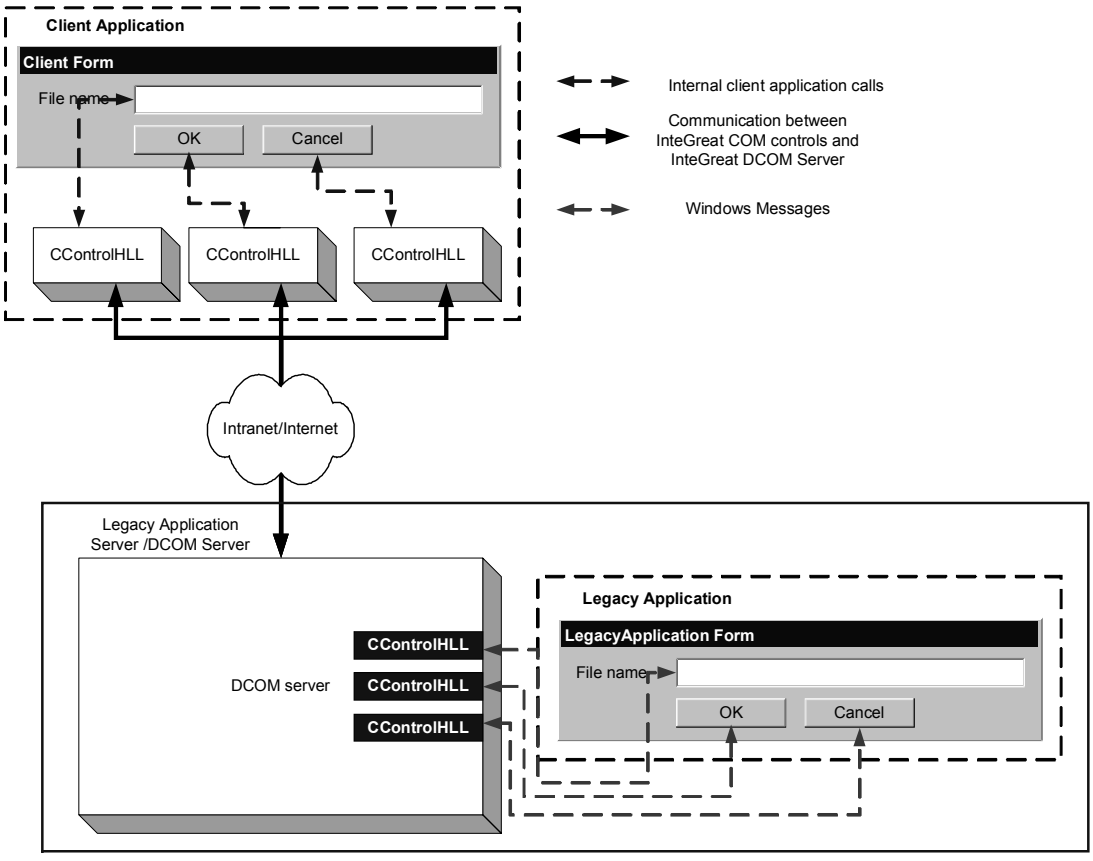

Рис. 1. Спрощена схема взаємодії компонент системи

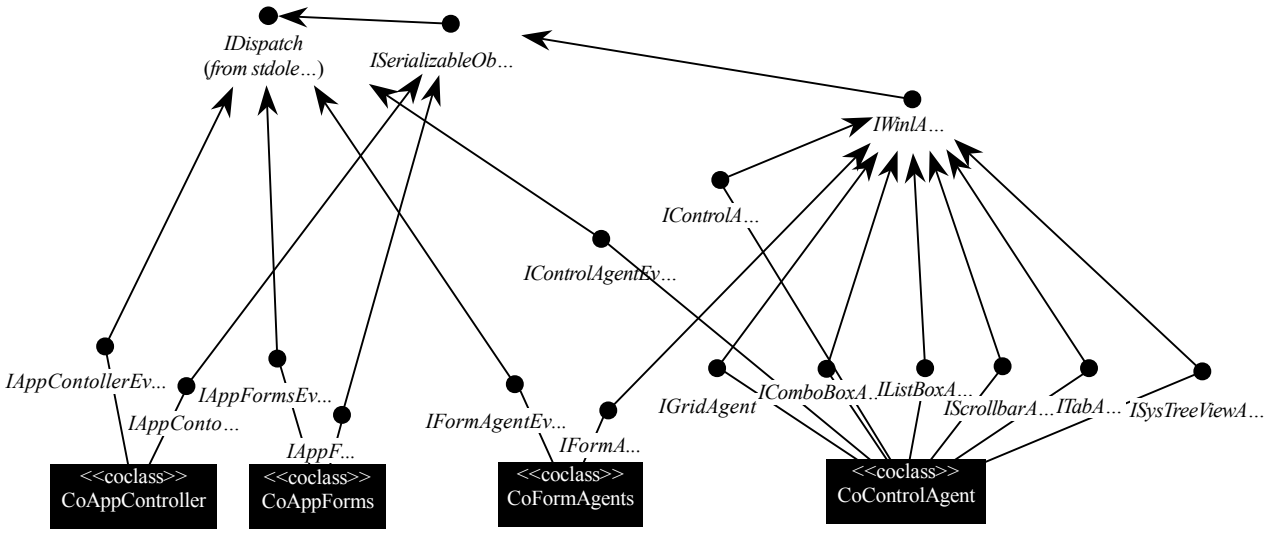

Рис. 2. Діаграма *COM*-інтерфейсів бібліотеки проксі-компонент

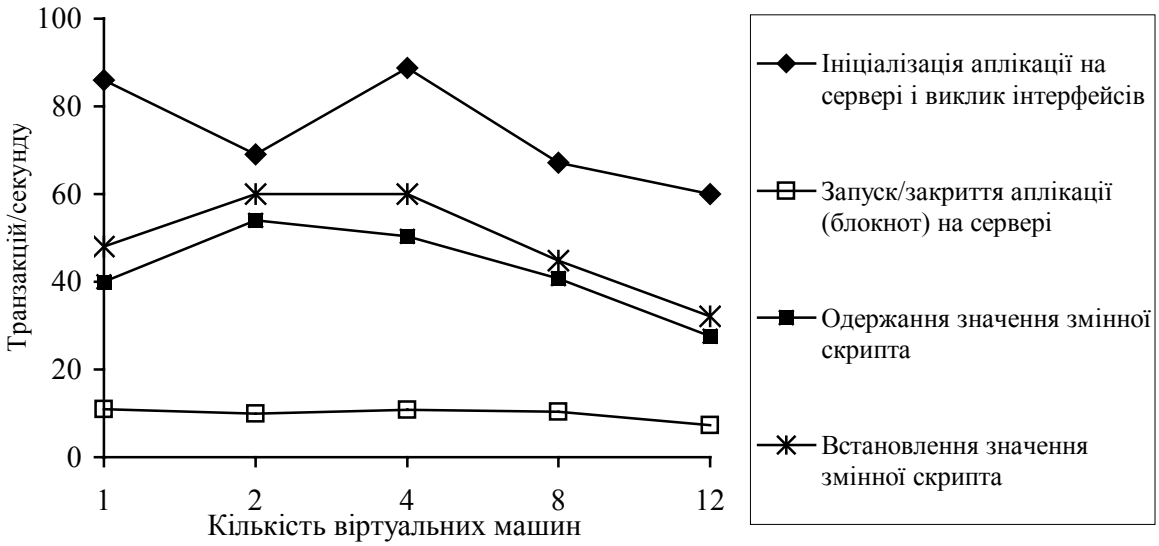

Рис. 3. Тестування швидкодії сервера

У цій роботі вперше так широко використано технологію *XML* (*Extensible Markup Language*) для подання даних у форматі, придатному для запису у потік. Мова *XML* використовується для обміну даними між аплікаціями. Аплікація, яка повинна передати дані, створює *XML*-документ, який вибудовується або безпосередньо у процесі роботи аплікації, або внаслідок застосування спеціальних інструментальних засобів. Програма, якій адресовані дані, використовує спеціальну компоненту конвертації (*XML-parser*), яка перетворює *XML* у формат, придатний для подальшої обробки. Зауважимо, що описана схема не залежить ні від мови програмування, ні від платформи, на якій вона виконується.

Для оцінки продуктивності системи було проведено тестування роботи програми з використанням віддаленого доступу, що був емульований програмою *WMVare*. Всі тестові операції виконані з використанням 100 Мбіт мережевого ввімкнення сервера – ОС *Windo*ws 2000 *Server*. Конфігурація комп'ютера – P4 1800 МГц, 1024 Мбайт ОЗУ. Для кожної віртуальної машини виділялося по 64 Мбайти ОЗУ. Результати тестування системи подано на рис. 3.

В системі реалізовані такі інтерфейси:

- *IcontrolAgent,* наслідуваний від *IWinAgent;*
- *IcontrolAgentEvents* – базовий інтерфейс, від

якого успадковуються всі елементи даної системи;

- *Ico*mboBoxAgent – інтерфейс роботи елемента *ComboBox*;
- *I*gridAgent – інтерфейс роботи елемента Grid;
- *IlistBoxAgent* – інтерфейс роботи елемента *ListBox*;
- *IscrollbarAgent* – інтерфейс роботи елемента *ScrollBar*;
- *ItabAgent* – інтерфейс роботи табуляції;
- *Isys*TreeViewAgent – інтерфейс роботи елемента *TreeView*.

Внутрішня реалізація об'єкта інкапсульована в агрегованому класі *CDefaultControl*, який і здійснює функціональність роботи з елементами графічного інтерфейсу. Від цього класу наслідуються класи, що відповідають за специфічну функціональність конкретного типу елементів. Ієрархію класів віконних елементів наведено на рис. 4. Система побудована з використанням трирівневої моделі. Вся найважливіша функціональність інкапсульована в *COM*-компонентах.

Найнижчому рівню відповідає компонента розбору і доступу до оригінальної аплікації. Це ядро системи. Воно забезпечує безпосередній зв'язок з оригінальною аплікацією.

Середньому рівню відповідають компоненти інтерпретатора скриптів і *DCOM*-сервера. Вони не тільки використовують інтерфейси проксі-об'єктів для взаємодії з ними, але й надають більш високорівневі інтерфейси керування клієнтам. Наприклад, команда скрипта використовує інтерфейс проксі-об'єкта елемента, застосовуючи певну послідовність дій над ним для власного виконання.

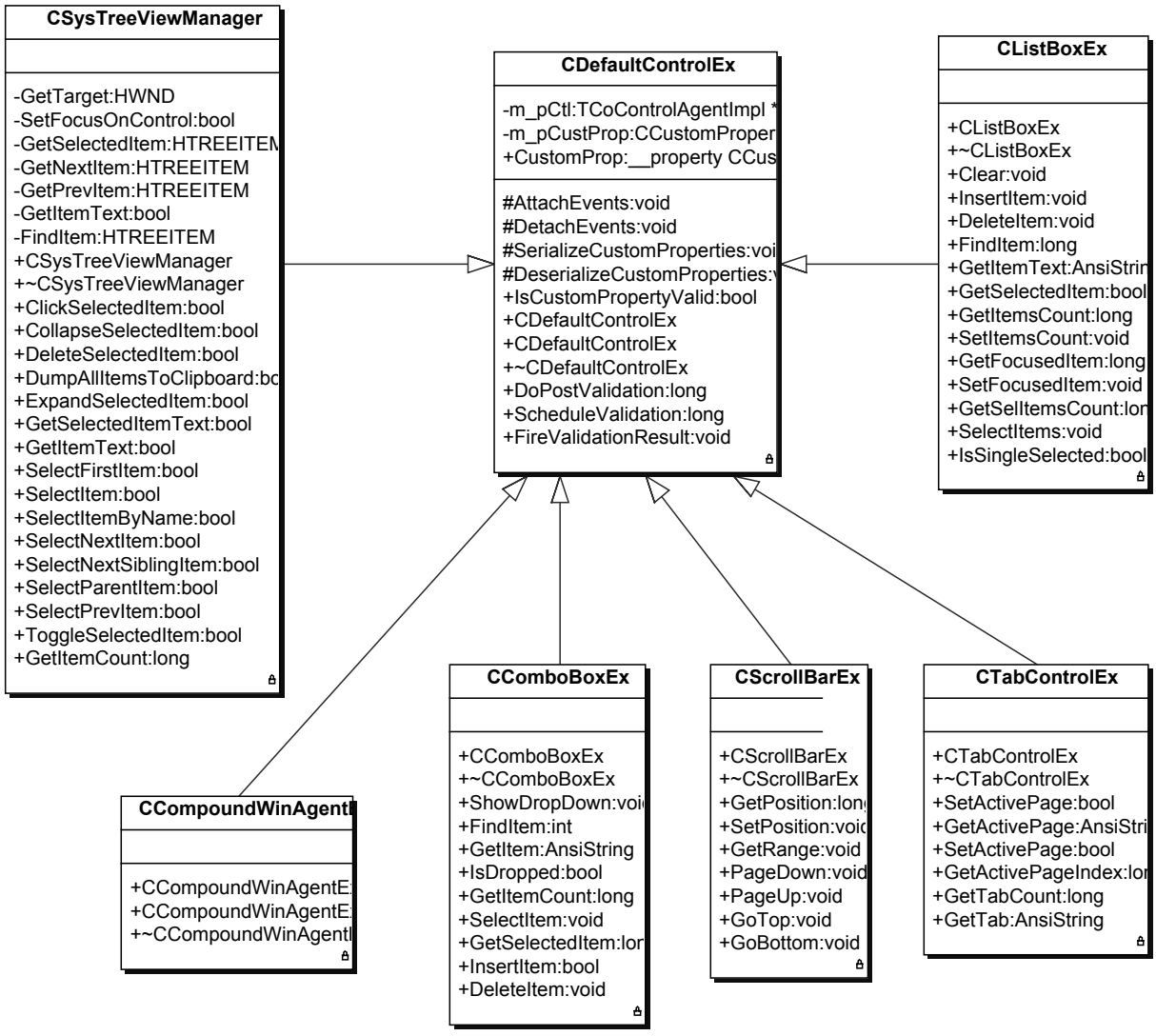

Рис. 4. Ієрархія класів віконних елементів

Дизайнер скриптів і клієнти *DCOM*-сервера можна віднести до найвищого, третього рівня системи. Використання цього рівня доступу надає зручні інтерфейси кінцевому користувачу.

Отже, інтеграцію з оригінальною аплікацією можна здійснювати на будь-якому з трьох рівнів. Доступ на найнижчому рівні надає повний набір методів доступу до об'єктів графічного інтерфейсу. На середньому рівні можлива інтеграція з використанням віддаленого доступу до аплікації як для автоматизації (запуск сервером скриптів автоматизації), так і для відображення інтерфейсу користувача на віддалений термінал. На цьому рівні можлива деяка спеціалізація доступу до аплікації. Так, використовуючи мову високого рівня (*Java*, *C++*, *C#*, *Delphi* та ін.), можна відтворити весь інтерфейс оригінальної аплікації або необхідну її частину, відкинувши непотрібні елементи і, у разі необхідності, додаючи нові. Прикладом цього може бути відображення необхідної частини графічного інтерфейсу через *Web*-інтерфейс із використанням *Java*-аплетів.

Базова архітектура системи наведена на рис. 5.

Отже, розроблену систему можна розглядати як один із нових підходів до розробки програмних засобів автоматизації роботи з *Windows*-аплікаціями.

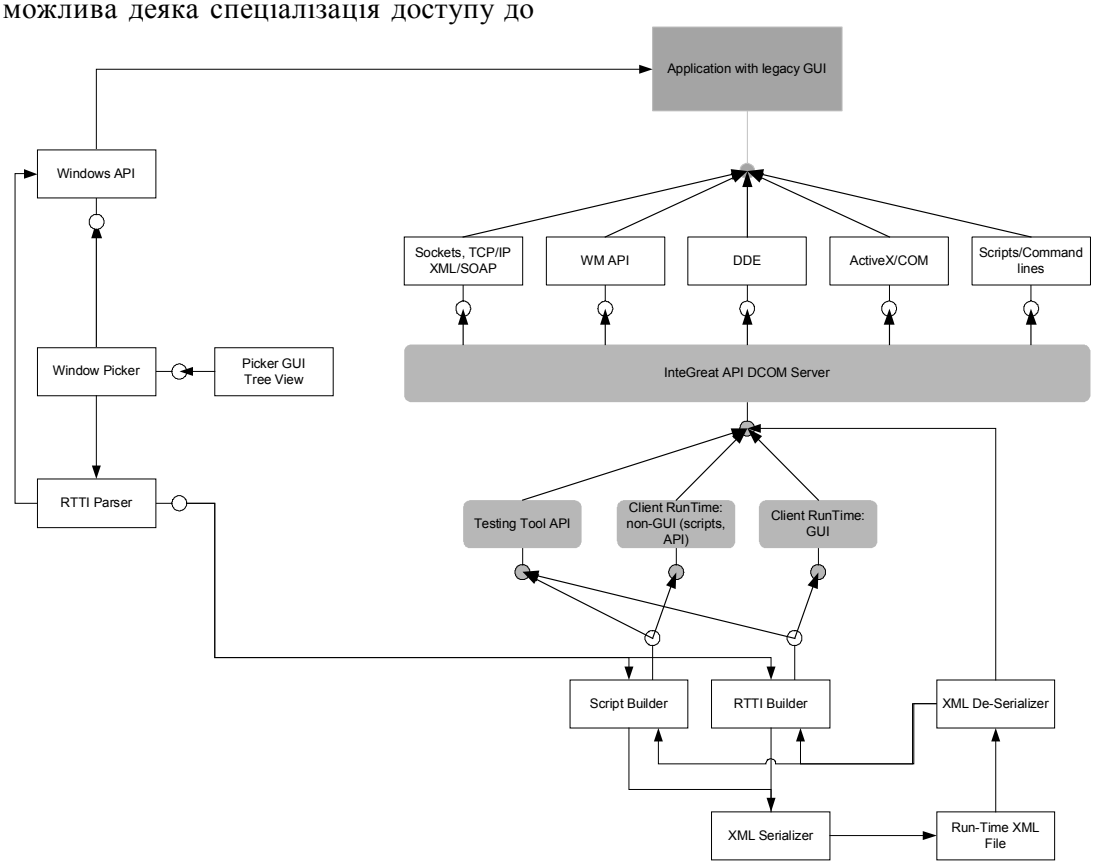

Рис. 5. Базова архітектура системи

СПИСОК ЛІТЕРАТУРИ

1. *Рихтер Д.* Windows для профессионалов: создание эффективных Win32 приложений с учетом специфики 64-разрядной версии Windows. – М.: Русская редакция, 2001.
2. *Рихтер Д., Кларк Д.* Программирование серверных приложений для Microsoft Windows 2000. Мастер-класс. – М.: Русская редакция, 2001.
3. *Ганеев Р.* Проектирование интерфейса пользователя средствами Win32 API. – М.: Горячая линия – Телеком, 2001.
4. *Неббет Г.* Справочник по базовым функциям API Windows NT/2000. – М.: Вильямс, 2002.
5. *Троелсен Э.* Модель COM и применение ATL 3.0. - Спб.: BHV, 2001.
6. *Оберг Д.* Технология Com+. Основы и программирование. – М.: Вильямс, 2000.
7. *Мартин Д., Бирбек М., Лозген Б. и др.* XML для профессионалов. – М.: Лори, 2001.
8. *Архангельский А.* Программирование в C++ Builder 6 (с заворотом). – М.: Бином, 2003.