УДК 681.3.06

С.В. Кур'ята, аспір.
Житомирський інженерно-технологічний інститут

ВИКОРИСТАННЯ OLE АВТОМАТИЗАЦІЇ У ПРИКЛАДНІЙ ПРОГРАМНІЙ СИСТЕМІ "DSR OPEN LAB 1.0"

(Представлено к.т.н., доц. М.М. Колодницьким)

Розглянуто особливості використання технології ActiveX, зокрема OLE Автоматизації, у прикладній програмній системі "DSR Open Lab 1.0". Наведено загальну схему обміну інформацією між компонентами системи, приклади програмної реалізації контролю компонентів системи та обміну інформації між ними за допомогою OLE Автоматизації.

В роботах [1, 2, 3] було представлено прикладну програмну систему (ППС) "DSR Open Lab 1.0" (Dynamical system research open laboratory), її архітектуру та інтерфейс користувача. Головним призначенням ППС "DSR Open Lab 1.0" є навчальний процес. У роботі [4] описана класифікація (типологія) математичних моделей, які охоплює програмна система: абстрактні множини, числові системи, функціональні відношення, векторний простір, алгебраїчні системи, поліноми, логічні дискретні моделі, звичайні диференціальні рівняння тощо. Такий широкий спектр математичних структур зумовлює специфіку інтерфейсу користувача ППС "DSR Open Lab 1.0", яка полягає в тому, що необхідно мати засоби опису не однієї, а цілого ряду тематичних моделей [3].

В першому наближенні, архітектуру інтерфейсу ППС "DSR Open Lab 1.0" можна представити як таку, що складається з кількох основних компонентів: основного каркаса системи (framework) та навігатора для математичних моделей. Однією з головних частин навігатора математичних моделей є майстер опису математичної моделі, який допомагає користувачеві у формі, що максимально наближена до предметної області, описати характеристики задачі, яку потрібно розв'язати або дослідити. Майстер опису математичної моделі побудований з кількох програмно незалежних компонентів – серверів. До його складу входять: компонента Equation Builder для введення опису математичної моделі задачі, сервер додаткових вхідних даних, сервер вибору завдань на аналіз (вибору видів вихідних результатів), сервер коментарів.

При створенні майстра опису математичної моделі та основного каркаса системи використовувалась технологія ActiveX. З точки зору ActiveX технології, всі перераховані вище сервери реалізовані як Active Document Servers. Основний каркас системи реалізований як Active Document Container. Він відповідає за виконання загальних функцій, таких як надання необхідних даних для компонентів системи, організація та контроль обміну даними, взаємодії між компонентами, збереження та відкриття файлів, із якими працює користувач, управління панелями інструментів тощо.

Сервери опису математичної моделі програмно вставлені (embedded) в основний каркас системи за допомогою технології OLE (Object Linking and Embedding), яка грунтується на моделі компонентного об'єкта (Component Object Model, COM).

Як інструментальний засіб використовувався пакет Microsoft Visual Studio 6.0, зокрема компілятор Visual C++ та бібліотека MFC.

Для реалізації контролю компонентів системи та обміну інформації між ними була використана OLE автоматизація. Загальна схема обміну інформацією між складовими частинами системи наведена на рис. 1. Серверові опису математичної моделі задачі [5, 6] від головного каркаса системи передається інформація про множину панелей інструментів, яку необхідно надати користувачеві, інформація про знаходження даного сервера в контейнері, встановлення ознаки модифікації документа. У свою чергу, головний каркас системи від даного сервера отримує введені користувачем дані та ознаку модифікації даних. Серверам додаткових вхідних даних та вибору завдань на аналіз передаються дані про види аналізів, види вихідних результатів, введені користувачем змінні, вказується необхідність поновлення даних. Серверам відображення результатів обчислень передається інформація про проведені розрахунки.

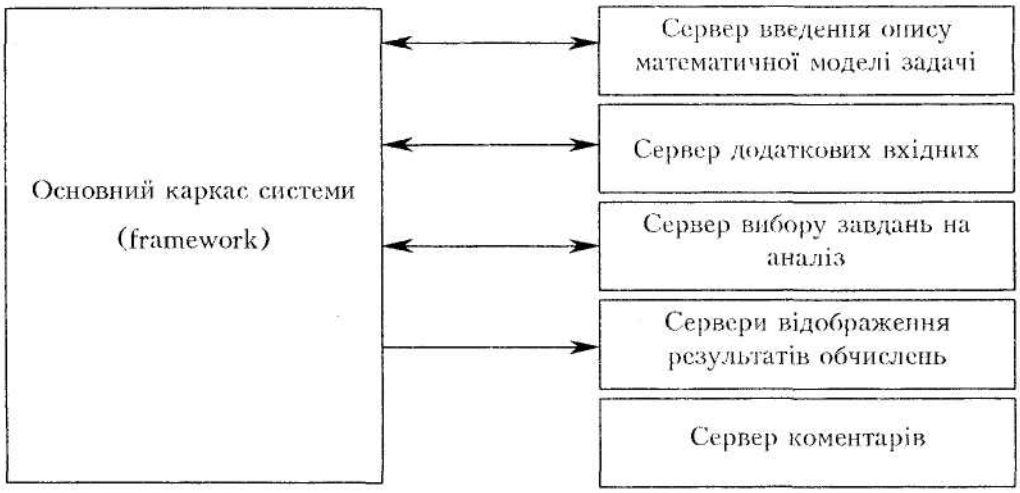

Рис. 1. Загальна схема обміну інформацією між компонентами системи

Розглянемо програмну модель використання OLE Автоматизації. Кожен із серверів в основному каркасі системи (контейнері) представлений своїм класом автоматизації. Такими класами є:

- `CDSRPage2Auto`;
- `CDSREquBAuto`;
- `CDSRPage3Auto`;
- `CGraphSrv`.

Усі перераховані вище класи автоматизації (`CDSREquBAuto`, `CDSRPage2Auto`, `CDSRPage3Auto`, `CGraphSrv`) як базовий клас мають MFC `COleDispatchDriver`, що представляє клієнта OLE Автоматизації.

Для звертання до методів OLE Автоматизації серверів використовується функція-член класу `COleDispatchDriver InvokeHelper(...)`. Наприклад, для класу `CDSREquBAuto` реалізація методу `AddToBuilderBuffer(...)` має вид представлений на рис. 2.

```
void CDSREquBAuto::AddToBuilderBuffer(LPCTSTR buff)
{
        static BYTE parms[] = VTS_BSTR;
        InvokeHelper(0x6, DISPATCH_METHOD, VT_EMPTY, NULL, parms, buff);
}
```

Рис. 2. Приклад використання функції `InvokeHelper`

Розглянемо призначення та реалізацію кожного класу автоматизації. Клас `CDSRPage2Auto` призначений для обміну інформацією із сервером додаткових вхідних даних. Він інкапсулює набір функцій для доступу до методів OLE Автоматизації сервера. Список та опис функцій, що інкапсулюються цим класом наведено в табл. 1.

Таблиця 1

Методи класу `CDSRPage2Auto`.

Назва методів	Опис методів
`SetAttributes`	Встановлює атрибути сервера для доступу до баз даних. Параметри: long id_task – номер поточної задачі, long id_model – номер моделі розв'язаної задачі, long id_tab – номер закладки, long cooperative – номер кооперативності.
`Refresh`	Поновлює вміст сервера.
`GetModifiedAttr`	Повертає `TRUE`, якщо документ сервера був змінений.
`SetModifiedAttr`	Встановлює ознаку модифікації сервера. Параметри: `BOOL bModify` – ознака модифікації сервера.

Клас `CDSREquBAuto` призначений для обміну інформацією зі серверами опису математичної моделі задачі та відображення результатів. Він інкапсулює набір методів (табл. 2) для керування поведінкою зазначених серверів, для отримання введених користувачем даних і виведення результатів обчислень.

Таблиця 2

Методи класу `CDSREquBAuto`

Назва методу	Опис методу
`GetIsInDSRContainer`	Повертає `TRUE`, якщо сервер вставлений в основний каркас системи.
`SetIsInDSRContainer`	Встановлює властивість сервера, який є вставленим в основний каркас системи.
`GetPaperWidth`	Повертає встановлену ширину папера.
`SetPaperWidth`	Встановлює ширину папера. Параметри: `long width` – ширина папера.
`GetBuilderBuffer`	Повертає вміст буфера сервера (введену інформацію).
`IsDocModified`	Повертає `TRUE`, якщо документ сервера був змінений.
`SetToolBarsSet`	Встановлює множину панелей інструментів, що надаються користувачеві. Параметри: `long barSet` – номер панелі інструментів.
`AddToBuilderBuffer`	Виводить інформацію в буфер сервера. Параметри: `LPCTSTR buff` – інформація в текстовому виді.
`ClearAll`	Очищає вміст сервера.
`SetModify`	Встановлює ознаку модифікації сервера. Параметри: `BOOL bModify` – ознака модифікації сервера.

Клас `CDSRPage3Auto` призначений для обміну інформацією зі сервером вибору завдань на аналіз. Набір функцій, що інкапсулює даний клас для доступу до методів OLE Автоматизації сервера, співпадає з множиною функцій класу `CDSRPage2Auto` і, тому він тут не наводиться.

`CGraphSrv` призначений для обміну інформацією зі сервером відображення результатів обчислень. Він інкапсулює набір методів (табл. 3) для керування поведінкою сервера та для виведення результатів обчислень.

Таблиця 3

Методи класу `CGraphSrv`.

Назва методу	Опис методу
`AddRow`	Додає рядки вікон для перегляду результатів. Параметри: `long nCount` – кількість рядків.
`AddCol`	Додає стовпці вікон для перегляду результатів. Параметри: `long nCount` – кількість стовпців.
`DeleteRow`	Вилучає вікна перегляду результатів, що знаходиться в даному рядку. Параметри: `long nRow` – номер рядка.
`DeleteCol`	Вилучає вікна перегляду результатів, що знаходиться в даному стовпці. Параметри: `long nCol` – номер стовпця.
`DeletePane`	Видаляє вміст вікна. Параметри: `long nRow, long nColumn` – координати вікна перегляду результатів.
`ClearAll`	Очищає вміст сервера.
`DeleteGraph`	Видаляє графік із вікна. Параметри: `long nRow, long nCol` – координати вікна перегляду результатів, `long nGraph` – номер графіка.
`DeleteAllGraph`	Видаляє усі графіки з вікна. Параметри: `long nRow, long nCol` – координати вікна.

`AddGraph2DDecGD`
`AddGraph2DdecGD_I2R`
`AddGraph2DdecGD_I2I`
	Додає 2D графік явно заданої функції у декартових координатах.

`AddGraph2DDecGDPar`
`AddGraph2DDecGDPar_I2R`
`AddGraph2DDecGDPar_I2I`
	Додає 2D графік параметрично заданої функції у декартових координатах.

`AddGraph2DDecGDImp`
`AddGraph2DDecGDImp_I2R`
`AddGraph2DDecGDImp_I2I`
	Додає 2D графік неявно заданої функції у декартових координатах.

`AddGraph2DDecDiscr`	Додає 2D графік дискретної явно заданої функції у декартових координатах.
`AddGraph2DDecDiscr_I2R`	Додає 2D графік дискретної явно заданої функції у декартових координатах.
`AddGraph2DDecDiscr_I2I`	Додає 2D графік дискретної явно заданої функції у декартових координатах.

Усі перераховані функції класу `CGraphSrv`, що додають графіки функцій мають однакові параметри: `long nRow, long nCol` – координати вікна, `LPCTSTR lpszName` – назва графіка, `long nSize` – кількість точок розбиття інтервалу обчислення функції, `VARIANT* arr_1` – масив координат на осі X, `VARIANT* arr_2` – масив координат на осі Y. Тут масиви

координат осей X та Y інкапсулюються в тип даних OLE Автоматизації VARIANT. Робота з типом VARIANT потребує від програміста додаткових зусиль та уважності. Тому для полегшення роботи з цим типом даних були розроблені шаблонні функції `Vector2Variant` та `Variant2Vector`. Функція `Vector2Variant` як параметра, займає масив координат та повертає покажчик на об'єкт типу VARIANT, що зберігає в собі масив точок. Друга функція `Variant2Vector` виконує зворотне перетворення.

Отже, використання сучасної технології ActiveX, зокрема OLE Автоматизації, у прикладній програмній системі "DSR Open Lab 1.0", надає можливість розробити достатньо ефективну схему керування компонентами, обміну інформацією між основним каркасом (контейнером) системи та серверами. Крім того, значно спрощується задача створення гнучкого, універсального та інтуїтивно зрозумілого інтерфейсу користувача.

ЛІТЕРАТУРА:

1. *Kolodnytsky M., Ivanitsky I., Kovalchuk A., Kuryata S., Levitsky V.* "DSR Open Lab 1.0" – software system for simulation // Proceedings of the 21st International Conference on Information Technology Interfaces, Pula, Chroatia, 1999. – P. 31.
2. *Колодницький М.М.* Тривимірна компонентна архітектура прикладної програмної системи "DSR Open Lab 1.0" як втілення концепцій реінженерії // Теорія програмування. – 1998. – № 4. – С. 37–45.
3. *Колодницький М. Ковальчук А. Кур'ята С.* Типологія та архітектура інтерфейсу користувача прикладної програмної системи "DSR Open Lab 1.0" // *Проблеми програмування*, № 3–4, Київ, 1999.
4. *Колодницький М.М.* Типологія математичних моделей технічних систем. Частина 2. // Вісник ЖІТІ. – 1998.– № 7. – С. 208–218.
5. *Кур'ята С., Іванщький І., Рожик О.* Особливості реалізації підсистеми редагування виразів в програмному комплексі "DSR Open Lab 1.0" // Праці 1-ї національної науково-практичної конференції студентів та аспірантів "Системний аналіз та інформаційні технології", 28–29 червня 1999 р. – Київ, 1999. – С. 76–77.
6. *Кур'ята С.В.* Особливості реалізації багатопоточних обчислень у підсистемі редагування виразів "Equation Builder" програмного комплекса "DSR Open Lab 1.0" // Вісник ЖІТІ. – 2000. – № 14.

КУР'ЯТА Сергій Валерійович – аспірант Житомирського інженерно-технологічного інституту.

Наукові інтереси:

- комп'ютерні інформаційні технології;
- моделювання і розв'язок задач за допомогою обчислювальної техніки;
- математичне моделювання нелінійних ланцюгів.

Подано 20.09.2000.