ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В НАУКЕ, ОБРАЗОВАНИИ, КУЛЬТУРЕ, МЕДИЦИНЕ, ЭКОНОМИКЕ, ЭКОЛОГИИ, СОЦИОЛОГИИ

УДК 519.713

ПРОБЛЕМЫ ИСПОЛЬЗОВАНИЯ ИНТЕРФЕЙСНЫХ СРЕДСТВ ОДНИХ ПРИЛОЖЕНИЙ ДРУГИМИ

КОБЫЛИН А.М., КУЗЁМИН А.Я.

Обобщаются практические результаты использования COM (Component Object Model) технологии в различных программных средах. Представляются практические рекомендации, которые могут быть полезны для создания Internet приложений, отвечающих принципам самоорганизации. Чтение статьи удобно сочетать с наличием “под рукой” соответствующих программных средств.

1. Описание средств и постановка задачи

Существует непосредственная зависимость между сложностью задач систем практической информатики, их структурой и сложностью выполняемых ими функций. Структурная сложность приложений заложена в системах практической информатики, а функциональная – в той информации, которую несут решаемые системой задачи.

Согласно Н.Винеру [1] превышение количества воспринимаемой информации (функциональной сложности системы) над количеством внутренней (ее структурной сложности) после определённого порога может привести к развитию её в смысле самоорганизации.

Проблема создания современного, эффективного программного средства практической информатики, отвечающего принципам самоорганизации, может быть преодолена путем решения следующих задач:

– необходимо отобрать наиболее эффективные компоненты программных средств для повышения функциональной сложности проектируемой системы её структурной сложности;

– в стратегии программирования нужно осуществить доступ к компонентам, которые обеспечат развитие системы или её самоорганизацию;

– необходимо обеспечить процесс проектирования практической информатики методом создания компонентов и построения из них приложений на основе COM технологии [2-5].

Начнём с рассмотрения современных средств COM технологии [2-5]. Их реализацию авторы отработали для различных задач практической информатики.

Ознакомление и отработку практических навыков использования **OLE** и ActiveX технологии программирования можно выполнить в такой последовательности:

– изучение связывания и встраивания объектов **OLE** в среде программирования Delphi;

– изучение активизации связанных и встроенных в приложение объектов **OLE** в среде программирования Delphi;

– изучение связывания и встраивания объектов **OLE** в среде программирования Builder C++ и их активизация;

– создание элементов управления ActiveX в среде программирования Delphi;

– создание COM-объектов с использованием библиотеки активных шаблонов ATL среды разработки Visual C++;

– создание COM-объектов и включение их в другие приложения.

2. Современные средства COM технологии

Связыванием (linking) называют ассоциирование файла объекта **OLE** с контейнером **OLE**. Файл объекта никогда не сохраняется в контейнере, но контейнер **OLE** ссылается на файл. Файл объекта **OLE** должен быть создан до того, как на него можно будет ссылаться из контейнера.

Документ Word или файл растрового изображения (BMP-файл) не является файлом объектов **OLE**— это просто файл данных. Чтобы создать из документа файл объекта **OLE**, нужно указать, что сервер **OLE** должен создавать файл в формате объекта.

Одним из преимуществ связывания объектов является то, что множество пользователей, серверов **OLE** и приложений контейнеров могут получать доступ к одному документу. Следовательно, достаточно расположить данные в известном каталоге и обеспечить к ним доступ для всех приложений, которые эти данные используют. Когда данные объекта **OLE** изменяются, то изменения автоматически отражаются во всех контейнерах, содержащих ссылки на файл этого объекта, включая и ваш собственный контейнер.

Встраивание (embedding) объектов – это второй метод, используемый при работе с объектами **OLE**. При встраивании реальный объект сохраняется в вашем приложении и другие контейнеры **OLE** не имеют к нему доступа. Встроенные объекты **OLE** не хранятся в файлах – они сохраняются в контейнерах внутри приложений. Встраивание объектов гарантирует, что объект не будет случайно удален, изменен или поврежден, как это случается при хранении объектов во внешних файлах.

Преимущество встраивания состоит в хранении данных как части приложения, а недостатком такого способа хранения является то, что размер контейнера **OLE** вашего приложения увеличивается на размер вставленного объекта. Таким образом, чем больше размер объекта, тем больше размер вашего приложения.

Если во время выполнения приложения в объект **OLE** внесены изменения, которые необходимо сохранить до следующего запуска приложения, то с

помощью метода **SaveToFile** объект **OLE** необходимо записать в файл.

Компонент **TOIeContainer** инкапсулирует функции контейнера **OLE** в компоненте **VCL**, который значительно проще в использовании. К вставленным в контейнер серверам **OLE** пользователь может получить прямой доступ или доступ посредством программных средств.

В **Delphi 4** компонент **TOleContainer** был создан заново, чтобы обеспечить вставку активных документов в контейнер. Теперь во время разработки в контейнер можно вставлять активные документы из **Microsoft Word** и редактировать их непосредственно в контейнере. В пакет программ **Microsoft Office 97** вошли единственные на сегодняшний день приложения, которые манипулируют активными документами.

Компонент **TOIeContainer** позволяет связывать и встраивать объекты **OLE** в приложение. В этом компоненте скрыта большая часть сложных механизмов **OLE**, что позволяет пользователям вставлять объекты в приложение путем вызова метода **InsertObjectDialog**. Для встраивания объекта **OLE** программными средствами следует воспользоваться методом **CreateObject**, **CreateLinkToFile** или каким-либо другим методом создания связанных объектов **OLE**.

В классе **TOIeContainer** добавлено одно новое свойство— **AllowActiveDoc**. Оно используется для поддержки документов типа **ActiveDocument**. Если свойство **AllowActiveDoc** принимает значение **True**, то в контейнер можно вставлять активные документы.

Компонент **TOIeContainer** позволяет связывать и встраивать объекты **OLE** в приложении. В этом компоненте скрыта большая часть сложных механизмов **OLE**, что позволяет вставлять объекты в приложение путем вызова одного метода **InsertObjectDialog**.

Для автоматической инициализации встроенного объекта необходимо изменить свойство **AutoActivate** контейнера **OleContainer1** с **aaDoubleClick** на **aaGetFocus**.

Событие **OnObjectMove** является специфическим для класса **TOIeContainer**. Если необходимо предоставить пользователям возможность перемещать контейнер или изменять его размеры (перемещая или изменяя размеры пунктирной рамки вокруг объекта **OLE**), то это событие нужно обрабатывать. В обработчике события в записи **BoundsRect** нужно установить значения, полученные в параметре **Bounds** обработчика события, в противном случае контейнер не изменит положения и размера. Ниже показан код простого обработчика события **onObjectMove**.

```
procedure Tform1.OleContainer1ObjectMove
(OleContainer: OIeContainer; const Bounds: Trect);
 begin
 // Не допустить, чтобы размеры контейнера были
 // меньше размеров объекта.
 with OleContainer1 do
 begin
  if left <100 then Left := 100;
  if Height < 100 then Height := 100;
  if Top < 100 then Top := 100;
  if Width < 100 then Width := 100;
 end; // with
 OleContainer1.BoundsRect := Bounds;
 end; // Tforml.OleContainer1ObjectMove
```

Для создания объектов OLE программными средствами можно воспользоваться методом **CreateObject**, а для создания связанного объекта **OLE** – методом **CreateLinkToFile**.

Заметьте, что не каждый файл может быть вставлен в **TOleContainer**. Для вставки подходят только те файлы, которые контейнер распознает. Если в контейнер вставляется документ, то в **Windows** должна быть зарегистрирована ассоциация для загрузки сервера **OLE**, в противном случае нужно использовать файл объекта. В показанном ниже фрагменте кода выполняется проверка возможности вставки объекта. Если объект нельзя вставить, то выводится соответствующее сообщение.

```
Procedure TForml.ButtonlClick(Sender: TObject);
begin
 with OleContainer1 do
 if InsertObjectDialog then
  DoVerb(ovShow) ;
else
ShowMessage('An object MUST be specified');
end; // TForml.ButtonIClick
```

Для создания объекта **OLE** с помощью метода **CreateObject** нужно указать значение идентификатора программы (**ProgID**) и вид объекта (в виде пиктограммы, значение **True**) или сам объект (значение **False**).

Для создания объектов **OLE** на этапе выполнения в среде **Visual Basic 5** используется элемент управления **OLE**. Он предоставляет методы создания объектов, приведенных в табл.1.

Таблица 1

Метод элемента управления **OLE**	Применение
OLEobject.InsertObjDlg	Выводит стандартное диалоговое окно **OLE Insert Object** (Вставить объект), чтобы пользователь смог выбрать, какой **OLE**-объект создавать
OLEobject.CreatEmbed file [,type]	Создает код для внедренного объекта без вывода диалогового окна **OLE Insert Object**
OLEobject.CreatLink file	Создает связанный объект из файла без вывода диалогового окна **OLE Insert Object**

Для создания новых внедренных объектов метод **CreateEmbed** использует типы классов объектов **OLE**. В табл. 2 перечислены типы классов некоторых распространенных объектов **OLE**.

Используя код

```
Private Sub OLE1_DblClick()
OLE1.CreateEmbed "", "Тип класса"
' Создать новый внедренный объект OLE
End Sub
```

Таблица 2

Приложение	Тип класса
Microsoft Equation Editor	**Editor**
Microsoft Equation Draw	**MSDraw**
Документ **Microsoft Word** (любая версия)	**Word.Document**
Документ **Microsoft Word** (любая версия)	**Word.Document**
Документ **Microsoft Word** (версия7.0)	**Word.Document.7**
Рисунок **Microsoft Word** (версия7.0)	**Word.Picture.7**
Microsoft Note-it	**Note-it**
Microsoft WordArt	**WordArt**
Пакет (файл произвольного формата)	**Package**
Microsoft Paint (растровое изображение)	**Paint.Picture**
Microsoft Quick Recorder	**SoundRec**
Диаграмма **Microsoft Excel**	**Excel.Chart**
Лист **Microsoft Excel**	**Excel.Sheet**
Лист **Microsoft Excel** (версия7.0)	**Excel.Sheet.7**
Лист **Microsoft Excel** (версия7.0)	**Excel.Chart.7**

Builder C++ позволяет создавать элементы управления ActiveX различных типов на этапе проектирования. Работать с элементами ActiveX сравнительно просто. Для этого сначала нужно поместить элемент в библиотеку VCL, а затем — в форму. В данной работе будет рассмотрен пример вставки элемента ActiveX в модуль (package) и способ его использования.

Среда программирования Builder C++ поставляется с набором различных элементов управления ActiveX. Элементы управления OCX сгруппированы на странице OCX палитры компонентов. На странице Win95 собраны элементы управления OLE, которые заключены в модули-оболочки. Кроме того, несколько элементов ActiveX размещено на странице Internet. Как и в случае с компонентами VCL, для использования перечисленных элементов ActiveX достаточно вставить их через компонент TOLEContainer в форму. Между компонентами VCL и компонентами, созданными на базе OLE, есть одно большое различие - компоненты OLE можно использовать в любой среде и в любом языке программирования, т.е. в любой среде программирования, которая поддерживает элементы управления ActiveX.

Технология ActiveX компании Microsoft представляет собой технологию OCX, переделанную для обеспечения передачи элементов управления через службы Internet, в основном через World Wide Web, а также для использования в таких средах программирования, как Borland C++ Builder, Visual C++ и Visual Basic. Чтобы использовать эту возможность, Delphi предлагает с помощью среды DAX (Delphi ActiveX) заключить потомка класса TWinControl в элемент управления ActiveX. Ниже перечислены платформы, на которых сертифицированы для выполнения элементы управления ActiveX, поставляемые с Delphi:

Borland Delphi версий 2 и 3.
Borland C++ Builder.
Borland Paradox 8.
Borland IntraBuilder.
Microsoft Visual C++.
Microsoft Visual Basic версий 4 и 5.
Microsoft Internet Explorer 3.01.
Microsoft ActiveX Control Pad.
Microsoft FrontPage.

OCX-файл можно импортировать в любую среду разработки, поддерживающую элементы управления ActiveX, которые созданы в Delphi.

Метод InitializeControl переопределен для вставки элемента управления VCL, который представляет элемент ActiveX, и для выбора методов для каждого обработчика событий в файле реализации.

```
Procedure TPieX.InitializeControl;
 begin
  FDelphiControl := Control as TPie;
  FDelphiControl.OnClick := ClickEvent;
  FDelphiControl.OnDblClick := DblClickEvent;
 end.
```

Методы обработки события являются подходящими кандидатами для вставки специального кода настройки элемента управления ActiveX.

События диспинтерфейса должны быть связаны с каналами обработки событий (Event Sink) в методе EventSinkChanged. Процедура для реализации такого метода имеет следующий вид:

```
Procedure TPieX.EventSinkChanged(const EventSink:
IUnKnown);
 begin
  FEvents := EventSink as IPieXEvents;
end.
```

В конце файла PieImp1.pas вы увидите фрагмент кода, создающий специальный класс среды DAX TActiveXControlFactory. После создания этот класс регистрирует элемент управления ActiveX вместе с глобальным объектом сервера COM Delphi ComServer. В процессе регистрации серверу ComServer передается идентификатор класса CLSID и ссылка на класс, благодаря чему сервер знает, что нужно создать по запросу с выбранным идентификатором класса. Фрагмент этого кода имеет вид:

```
initialization
 TActiveXControlFactory.Create(
  ComServer,
```

```
    TPieX,
    TPie,
    Class_PieX, 1,
'(6DB6FBD0-96D3-11D0-A0C0-00C04FC29155)',
    0);
  end.
```

Уникальный идентификатор GUID, являющийся последним параметром конструктора

TActiveXControlFactory.Create,

используется в качестве значения лицензионного ключа. Можно спокойно изменить значение этого параметра, но в этом случае нужно внести соответствующие изменения в файл <Имя проекта.>.lic, который создан, чтобы различать лицензирование элемента управления ActiveX на этапах разработки и выполнения. Практически так создаётся элемент управления ActiveX.

3. Выводы

Излагаемые результаты исследования программных средств для реализации COM технологии показывают, с одной стороны, практические возможности интегрированных визуальных средств проектирования, а с другой – обеспечивают создание интерфейсов пользователя с признаками самоорганизации, что особенно важно при создании Internet приложений.

Читатель не только познакомился с современными средствами COM технологии, но и получил возможность, следуя рекомендациям, излагаемым в статье, самостоятельно выполнить заказ по разработке приложений практической информатики.

Литература. 1. *Кузнецова В.Л., Раков М.А.* Самоорганизация в технических системах. К.: Наук. думка,1987. 200 с. 2. *Чеппел Дэвид.* Технологии ActiveX и OLE: Пер. с англ. М: Издательский отдел "Русская Редакция" ТОО "Channel Trading Ltd.", 1997. 320 с. 3. *Роджерсон Дуйл.* Основы COM: Пер. с англ. М: Издательский отдел "Русская Редакция" ТОО "Channel Trading Ltd.", 1997. 376 с. 4. *Миллер Тодд, Пауэл Дэвид и др.* Использование Delphi 4. Специальное издание: Пер. с англ. К.: Диалектика, 1997. 768 с. 5. *МакКелви М., Мартинсон Р., Веб Дж., Ризельман Б.* Visual Basic 5: Пер. с англ. СПб.:BHV-Санкт-Петербург, 1998. 976 с.

Поступила в редколлегию 30.03.99

Рецензент: д-р техн. наук Сироджа И.Б.

Кобылин Анатолий Михайлович, канд. техн. наук, профессор кафедры информатики ХТУРЭ. Научные интересы: COM-технологии. Увлечения: садоводство. Адрес: Украина, 61726, Харьков, пр. Ленина, 14, тел. (0572) 40-94-19.

Кузёмин Александр Яковлевич, канд. техн. наук, доцент кафедры информатики ХТУРЭ. Научные интересы: системный анализ. Увлечения: туризм. Адрес: Украина, 61726, Харьков, пр. Ленина, 14, тел. (0572) 40-94-19.

УДК 681. 335.001.53

СТРУКТУРА ПРОСТРАНСТВА ПРЕОБРАЗОВАНИЙ ИНФОРМАЦИИ В МАТРИЧНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЯХ УПРАВЛЕНИЯ ПРОЕКТАМИ

ТЕСЛЯ Ю.Н.

Излагается подход к разработке структуры проектно-ориентированной информационной технологии управления в функционально-ориентированных управленческих структурах. Предлагается матричная схема декомпозиции информационной среды систем управления проектами и организациями с последующим построением пространства преобразований в матричной информационной технологии управления проектами.

Необходимость создания значительного экономического потенциала у отечественных предприятий требует поиска новых форм и методов реализации крупных государственных программ, методов, которые могут применяться в сложившихся в Украине условиях. Одним из эффективных современных направлений в области совершенствования производственной деятельности сложных организационно-технических систем является методология Управления Проектами [1].

Сущность данной методологии состоит в правильной координации моделей, оборудования материалов, финансовых средств и графиков для выполнения определенного проекта в заданные сроки, в пределах бюджета и удовлетворения условий заказчика (потребителя). Переход отечественных предприятий к новым условиям хозяйствования, изменение форм собственности, отношений с государственными органами, появление новых экономических структур, свойственных странам рыночной экономики, нестабильность законодательства повлияли на системы планирования, учета и отчетности, выдвинули ряд требований к организации проектно-ориентированной деятельности. Изменились условия финансирования народного хозяйства.

Финансовые трудности резко сталкиваются с практической невозможностью эффективного функционирования в новых условиях некомпьютеризированного предприятия.

В технологическом плане разработку современных информационных систем и технологий характеризует ориентация на [2]:

– использование опробованных на практике методологий проектирования, таких как SSADM,IDEF, Merise,Dafne,NIAM;

– применение клиент-серверных WWW- и Java-технологий;

– создание программного обеспечения (ПО) с независимых компонентов и, как последствие, применение ПО объединения типа OLE, ActiveX, CORBA Plus;

– использование инструментальных сред разработки программных систем;

– применение средств концептуального моделирования и CASE-технологий создания ПО.

Главная современная проблема информатизации при реализации сложных народнохозяйственных проектов связана с информационным наполнением среды инструментальных программных средств. Опыт показывает, что существующие программные средства в полной мере и эффективно позволяют решать задачи анализа и управления проектами (Project Expert, Sure Track, MS Project, P3 и др.), всем процессом движения материалов от поставщиков