ПРИМЕНЕНИЕ DCC32 ДЛЯ СОЗДАНИЯ РАЗВИВАЕМЫХ ПРОГРАММНЫХ СИСТЕМ НА DELPHI

А. Г. Королев, А. И. Рязанцев (Северодонецк, Украина)

Введение

При создании сложных программных систем управления и моделирования нередко возникает дилемма – или программная система не вполне соответствует требованиям заказчика (конкретного пользователя), или она оказывается чрезмерно сложной и перенасыщенной малополезными возможностями.

Способы преодоления указанной проблемы известны. Как правило, в систему включают технологический язык, который позволяет ее развивать. Так сделано, например, в пакете Microsoft Office, где для целей развития используется язык Visual Basic for Application, или в системе 1С предприятие, где также используется свой внутренний язык. Однако создание своего внутреннего языка для каждой пользовательской системы имеет и свои недостатки. Самый очевидный из них – этот язык еще следует изучить конечному пользователю. Кроме того, он может оказаться недостаточно гибким или неудобным конечному пользователю. Разработка языка может также оказаться слишком сложной задачей, которая не соответствует сложности самой программной системы и так далее.

На наш взгляд, более перспективным было бы использование широко известных языков программирования в качестве технологических для программных систем. Например, это может быть язык Delphi, C#, Visual Basic или какой-либо другой язык класса 4GL

На этом пути, правда, возникает целый ряд проблем. И главная из них состоит в том, что конечный пользователь должен будет установить у себя в компьютере систему с соответствующим языком, а также научиться работать с его оболочкой.

Самый простой путь снятия части проблем состоит в создании специальной программы – конвертера, предназначенной для облегчения развития системы.Эта программа должна обеспечивать доступ к изменяемым частям проекта программной системы и компиляцию видоизмененной системы с созданием ее новой версии. Если использовать в качестве технологического язык Delphi (Object Pascal), то для компиляции конвертер мог бы вызывать на выполнение Run.Bat – пакет со следующим содержимым:

```
DCC32 ObjectProject.dpr >Error.txt
ObjectProject.exe
```

Здесь **DCC32 –** компилятор командной строки, имеющийся в папке BIN системы Delphi, а **ObjectProject.dpr –** главный файл проекта пользовательской системы. Возможные сообщения об ошибках поступают в файл **Error.txt.**

В самой программе-конвертере вызов пакета мог бы выглядеть следующим образом:

```
ChDir(extractFileDir(Application.ExeName));
WinExec('Run.bat',SW_HIDE).
```

Здесь первая строка делает текущей папку с приложением «конвертер», а вторая вызывает на выполнение файл Run.Bat таким образом, чтобы окно сеанса DOS было невидимым.

Для анализа возможных ошибок в приложение «конвертер» следует загрузить файл Error.Txt. Хотя компиляции подвергается и весь проект, но очевидно, что ошибки могут возникнуть только в той его части, где находятся изменения, внесенные конечным пользователем. Поэтому желательно, чтобы все изменения, вносимые конечным

пользователем в систему, выполнялись в отдельных, его собственных .PAS-файлах. Тогда, получив информацию о строках с ошибками компиляции, конечный пользователь сам сможет их найти и исправить в конвертере.

Чтобы визуализировать файл сообщений компиляции, можно воспользоваться следующим фрагментом кода:

```
var  i: Integer;
begin
  ChDir(extractFileDir(Application.ExeName));
  ErrorEdit.Lines.LoadFromFile('Error.txt');
  i := 0;
  while i<ErrorEdit.Lines.Count-1 do
  begin
    if ErrorEdit.Lines[i] = ErrorEdit.Lines[i+1] then
    ErrorEdit.Lines.Delete(i+1)
    else
    Inc(i);
  end;
end;
```

Здесь файл Error.txt загружается в компонент ErrorEdit, и в нем удаляются расположенные рядом одинаковые строки, как не несущие полезной нагрузки.

Однако такое «наивное» решение проблемы, на наш взгляд, является недостаточным. Необходимо обеспечить работу конвертера и самой программной системы без использования Delphi как такового.

Для решения этой второй задачи следует поступить таким образом.

Скопировать в папку с проектом программной системы содержимое папки BIN из системы Delphi.

Найти и переименовать файл DCC32, например, в DC32.

Удалить файл DCC32.CFG

Файл Run.Bat видоизменить, чтобы его содержимое имело вид

```
DC32 objectProject.dpr
rem >Error.txt
ObjectProject.exe
Pause
```

Затем нужно открыть папку LIB из системы Delphi, сохраняя открытой папку с нашим проектом.

Далее следует достаточно долго запускать вручную на выполнение файл Run.Bat, смотреть на экране сообщения об ошибках компиляции, а затем закрывать соответствующие окна DOS.

Сообщения будут указывать, каких именно файлов не хватает в проекте.

Соответствующие .DCU и .RES-файлы следует копировать из папки LIB в папку с проектом.

После копирования всех нужных файлов, а их число может быть более 100, проект окажется скомпилированным и запустится на выполнение.

Затем начинается вторая часть «доведения до ума» проекта.

Дело в том, что реально для компиляции системы нужны очень немногие файлы из папки BIN. Однако сказать точно и заранее, какие именно файлы понадобятся для создаваемой системы, очень непросто.

В том случае, когда авторы создавали свою систему, из папки BIN для Delphi-6 понадобились только следующие файлы:

```
lnkdfm60.dll
rlink32.dll
DC32.EXE
default.rps
```

Следует иметь в виду, что файл DC32.EXE – это переименованный DCC32.EXE.

Возможно, этих файлов будет достаточно. Общий же подход состоит в удалении файлов, позаимствованных из папки BIN, и проверки, возможна ли после этого компиляция проекта.

Если в проекте используются компоненты, принадлежащие кросс-платформе (библиотеки CLX), то тогда, возможно, понадобятся еще какие-либо .DLL-файлы.

По окончании процесса выяснения, какие именно файлы следует оставить, обнаружится, что ваша папка проекта будет иметь объем в любом случае не более 10–12 Мб. В нашем случае это было порядка 5 Мб плюс объем самого проекта.

Затем следует вернуть файл Run.Bat к следующему виду:

```
DC32 ObjectProject.dpr >Error.txt
ObjectProject.exe
```

На этом формирование папки для работы с программной системой и ее модернизации завершается. Результат следует проверить на компьютере, где заведомо нет системы Delphi или есть другая система Delphi.

Если необходимо дальнейшее развитие программной системы, то измененную систему следует перезаписывать в сформированную папку. Если при развитии не добавлялись новые типы компонентов, то, скорее всего, папка для работы с системой без Delphi остается в рабочем состоянии. Если же были добавлены новые типы компонентов, то, возможно, придется добавить из папки LIB еще несколько .DCU и .RES-файлов по сформулированному выше рецепту.

Выводы

Предложенный подход был опробован при создании системы Object GPSS, которая позволяет писать модели систем в стиле GPSS прямо на языке Object Pascal.

Эта система создает оконное приложение, являющееся моделью для конкретной задачи моделирования.

Авторы выражают уверенность, что изложенный подход окажется полезным и в других случаях использования языка Object Pascal в качестве технологического, а также в любых других случаях, когда вы захотите использовать «Delphi без Delphi».

Литература

1. **Гурин С.** Использование компилятора Delphi (DCC32.EXE) в прикладных программах. Личный сайт. 2001.
2. **Королев А. Г.** Сравнение команд и блоков GPSS World и Object GPSS. Сайт GPSS.RU раздел статьи. 2003.