Kylix: The waiting is over

Michael Van Canneyt

March 16, 2014

1 Introduction

Halfway 1999 Borland (then called Inprise) announced its plans to port their RAD tool Delphi from Windows to Linux under the new name of Kylix. After one year and a half, the result of their efforts will is available for sale from several websites, and shipping is expected to start in March of this year. For many people this will be the end of a long period of waiting, since Borland has actively tried to keep the developers' interest in Kylix high. Now, finally the time has come. What follows is an overview of what can be expected, based on experiences with some Beta releases (also called Field Tests, in Borland Jargon).

2 Is it really Delphi for Linux?

Yes, it is 100platform there will be no real new things in the IDE: Kylix is Delphi, running on a Linux platform. That this feat was performed using a small trick (the IDE runs on top of WINE, a Windows emulator that runs on top of Linux and X-Windows) is something that only the die-hard Linux-purists will protest about, because the final result definitely more than makes up for this. All features that one so easily gets accustomed to, such as code templates, code and class completion, complete integration of the visual designer with the powerful source editor, are there. The integrated debugger is also present, complete with the object inspector and CPU window.

For people that have previously developed under Linux, but do not know Delphi, developing will get a new dimension. Despite recent improvements in open source projects such as KDevelop or Glade, GUI or Web development for Unix will seem primitive compared to what can be achieved with Delphi; whole applications can be coded without typing a single line of code; (The database application shown in the figure is made without a single line of code, and allows to view and alter the data kept in a MySQL database) simple drag and drop and filling in of some properties will create a working database application in no time. The Visual designer and the more than 150 classes that ship standard with Delphi supply a developer with everything that is needed to create a working application without having to worry about the gory details of what needs to be done: The only coding that is left to do is the logic of your application's user interface. But the command-line people are not forgotten: the compiler can be called from the command-line and is lightning fast, so there also there is to be gained from Kylix.

The debugger of Kylix will probably leave little to be desired, even for GDB or DDD users; it features all that can be expected from a debugger; the debug inspector allows easy navigation through all classes and instances of classes. Although it is said to emit stabs debugging info (the native format of GDB), on the author's system, GDB reported '(no debugging symbols found)'.

One detail to be kept in mind for Linux developers is that Kylix uses Pascal (more precisely, Object Pascal, an Object Oriented version of Pascal, extended by Borland) as its programming language; For a system where the dominant programming language is C, this may prove a hurdle which some people will hesitantly take. For these people, the wait will continue a little longer, till C++ builder, which uses the same techologies and Libraries as Delphi, will be released on Linux as well.

As Delphi for Windows, Kylix will ship in 3 flavours, which don't quite correspond to their Windows counterparts, however. The 'Professional' version seems to have made place for the 'Desktop Developer' edition, and the 'Enterprise' edition survived the transition to Kylix under the name 'Server Developer'. New will be the release of an open-source version (scheduled to be released later this year).

3 What can it do?

Kylix is designed to make GUI and Web applications, usually with a database behind it, in a RAD way. People looking for a tool to make kernel modules, or command-line programs will find themself left in the cold, because Kernel modules cannot be built with Kylix and command-line programs will mainly have to be coded as before.

To this end, Kylix ships with more than 150 components to make GUI, Web and Database applications. They can be divided in three major parts:

CLX is a set of components (Unix users will probably be more familiar with the term 'Widgets') for the creation of GUI applications. Most of the Components that were present in the Windows version are present in the Linux version as well. Only some very specific Windows components are no longer present. Needless to say, the reference to windows in these components has disappeared. The 'Win32' page in the component palette is now called 'Common controls'.

The CLX components are built on top of the well-known Qt library from Troll Tech. Borland has used the Qt components to replace all Win32 specific code that was underlying the Class Library. Previous code that didn't use any Win32 specific code should therefore port easily to Linux, since almost all of the properties that were present under Windows, are present under Linux.

People without previous experience in Delphi will find themselves with a set of about 50 components (or widgets) that can be dropped on a form (window), for which a series of properties (color, dimensions, scaling behaviour etc.) can be set, and to which an event handler can be attached with the click of a mouse.

DataCLX Is a set of components (mostly non-visual) that can be dropped on a form and which represent data from a database; Built on top of Borland's Midas (known from Windows) and DBExpress (newly developed) technology, this provides access to data in several formats. Applications that access MySQL, Interbase, Oracle and DB2 databases can be coded or ported with little effort. As a byproduct of the use of Midas techology, applications can be coded that store data in a flat file, in XML format. (Borland has dubbed this functionality 'MyBase' in their 'Product sheet'). Many CLX classes therefore have a DB-Aware counterpart, as under Delphi for Windows. The use of Midas also means that making a three-tier applications will prove not much more difficult than a local database application.

People that coded GUI DB applications under Linux before will appreciate the ease of use with with a database application can be coded; for simple applica-

tions, a single SQL statement to retrieve the data to be edited can be the only 'coding' that should be done. More complex database applications will require more coding, but in general the coding is restricted to the business logic and user-friendlyness of the application. All the low-level code of handling screen and widget updates is handled by the database-aware components of Kylix.

NetCLX is the term used for all internet related components on the component palette of Kylix. There are components to make CGI programs as well as Apache modules; Components are provided to make client and server applications for many network protocols (The Internet Direct or Indy components, also known as Winshoes, equally available for Windows). The NetCLX cooperate with the DataCLX components so coding a web-site with a Database behind it should not prove more difficult than a GUI application using the same database.

Linux programmers that programmed internet applications will find the event-driven approach of the internet components quite intiutive and easy to use. The gory details of making connections and sending data are completely hidden by the components provided by Kylix. Programming Web applications should be reduced to handling the useful content of your pages.

Doubtlessly, many third party vendors of Delphi components will soon also release versions of their components for Kylix. People who developed their own components should find it not too difficult to migrate to Kylix, provided they didn't use specific Win32 code, but used Borland's wrappers around the Win32 API calls.

4 Porting issues

Windows developers will of course be interested in knowing whether their existing code will be reusable in Kylix. By and large, the answer to this question is 'Yes'. Of course, there are some things to take into account:

- There is no BDE for Linux. All applications that made use of the BDE will need to switch to Midas; the use of Midas introduces some additional complexity in an application, since now there is an extra 'layer' between the GUI part of the application and the actual Database. The BDE has been replaced by DBExpress. This is a thin layer around the underlying database engine, and is designed for SQL server applications. Currently there are drivers for Interbase, MySQL, Oracle and DB2 (the latter 2 only in the Server developer edition).

  There are of course third party (some even open source) alternatives for the BDE which may also get ported to Kylix.

- Similarly, the ADO database support is not available.

- Variant support. Variants are a 'feature' of the Windows operating system; as such, they have been deeply integrated in Delphi, even to the point of making it a special language construct. On Linux, there is no variant support by the OS. This is reflected in the Run-Time library, because all variant support has been removed from the System unit.

- There is no COM, or DCOM or CORBA. Since COM and DCOM are deeply rooted in Windows, it is not surprising that the parts of Delphi concerned with COM did not survive the step to Linux. The Interfaces language construct still exists, as the Interfaces implementation of Delphi is compatible with COM, but was implemented independent of COM. CORBA support may be made

available at a later time, but to the authors' knowledge, Borland hasn't made any statements about that.

- Any application that uses Win32 API calls will need to be re-coded. Not all Windows functionality can be emulated on Linux; often alternatives exist, but this may not always be the case. This is also the case for dynamic libraries; DLL's that are present on a Windows system (TAPI, MAPI and the like) are not present on Linux.

- The Netscape or IIS web-server components are also not ported to Linux; the latter for obvious reasons.

- Linux' file system is case dependent and uses the '/' character as a directory separator, not a backslash. All 'Uses' clauses in units must be checked and made case sensitive. A funny decision of Borland, since Pascal is a case-insensitive language, and the uses clause is part of the language specification. Linux also doesn't support drives, nor UNC filenames.

Finally, many units have been prefixed with the letter 'Q'; instead of the unit 'forms', one should now use the unit 'QForms' and so on; this is nothing major, but does require the programmer to make use of the IFDEF directive in his program code.

5 What about low-level Linux programming ?

The complete class library for Linux is built on top of the C library of the Linux system. The basic units of the windows system (mainly the SysUtils and System unit) work as expected, some of the specific windows calls have been emulated to work on Linux.

Since the C library headers have been translated to Pascal and are collected in the 'Libc' unit (partly equivalent to the 'Windows' unit on Windows), it is possible to make use of all functionality on a Linux system and create low-level (often command-line) Linux applications.

Other libraries that have been translated are X and Xpm libs. If more libraries are needed (many graphics libraries jump to mind) these will need to be translated by the programmer himself.

Kylix makes use of the Qt library, which is the basic building block for the KDE desktop system. The integration with the KDE desktop ends there: there is no KDE specific part in Kylix, meaning that, from the positive side, a Kylix application will run on a GNOME system as well, provided it has Qt installed. From the negative side, there are some KDE services which would greatly benefit a Linux programmer.

There is nothing which prohibits the use of the GTK toolkit (used in GNOME applications) in Kylix applications except that the translation of the GTK headers, needed to access the GTK library, is necessary[1]. The RAD aspect of Kylix will be lost then, since the components delivered with Kylix cannot choose the underlying toolkit.

So low-level programming with Kylix is definitely possible, but then the RAD aspect of development will be lost; even so, the good IDE source editor makes the use of Kylix a pleasure.

[1] a task completed by the Free Pascal team

6 Is this heaven?

Till now, this has seemingly been the 'good news show'. There are, however, also some bad points to be mentioned.

- The price tag. 999 USD for the desktop developer version and 1999 USD for the server developer version is a lot of money on a platform dominated by open source (and hence free) software. The promised open source version will certainly fill this gap, but it remains an open question what functionality will be included, and how license problems will be tackled.
- Kylix does not run on small systems, something Linux is well suited for. Applications generated by Kylix also use much memory; it is doubtful that a Kylix generated application will run smoothly on older Linux systems. (a 486 with 16 Mb ram, perfectly capable of running Apache with PHP will NOT do very well.). The same can be said of Kylix generated applications.
- It supports only on RedHat 6.2 or higher, Mandrake 7.2 or SuSE 7.0; even then these systems must have some basic components patched (C library, and the dynamic loader). Kernel version must be high enough. (2.2.16).
- A Kylix application is large and uses a lot of dynamic libraries. a simple 'Hello, World!' program takes 445kb and uses 12 shared libaries; when running, it used about 6Mb of memory resources. At the rate with which libraries change in the Linux world, this is not really a plus, since the possibility that a client runs a system with a different set of libraries is not imaginary; having to update this system may not be desirable since the updating of the system may well leave it in an unusably state.
- There is no installation program support; In view of the large number of libraries needed, something as 'Installshield' or 'Wise' for Linux is desirable. What is more, this should be a system that is compatible with the install formats RPM or Debian's .deb. Distributing programs may prove not an easy task; for someone wishing to install programs and provide support, this may prove a serious problem.
- Many libraries (and a linux system contains a lot of them) are not supported; if the use of these libraries is desirable, the C headers will have to be translated. On a system where shared libraries are a major component of the system, the lack of support is definitely a drawback. It will probably be simply a matter of time before most major libraries will be supported.
- Finally, the IDE itself is far from bugfree; in the course of a working day, it was not uncommon to have to kill Kylix at least 20 times; half of the times work is lost. The 'qps' or 'kpm' process managers were never far off. This was to be expected, since this must be considered the initial release of a new product. To the credit of Borland it must be said that this improved over the various beta tests (the opposite would have been surprising) and the class library itself behaved very stable (insofar it was tested.)

In spite of all these points, heaven is close by for the Windows Delphi programmer. Kylix installed flawlessly and after 15 minutes the first database application was running. this shows it is definitely possible to develop and deploy quality applications on Linux. The flaws and drawbacks still present will, no doubt, be fixed in the future. This prospect, and the prospect of being able to develop in a familiar environment on a OS which is evolving at great pace, should attract many Delphi

programmers to Linux. For Linux programmers, the IDE by itself is ample justification for giving Kylix a try - one need but to wait for the open source version. For the C programmers, the RAD aspect will hopefully more than make up for the nuisance of learning Object Pascal.