Development environments on test

THE RIGHT FOUNDATIONS

Development environments have the function of simplifying the programming and development of software. Tim Schuermann presents an overview of the most interesting products under Linux

Linux has been something of a paradise for programmers right from its modest beginnings, in fact there's hardly a programming language around that can't be found under this Free operating system. However, the problem with many languages is their somewhat cryptic operation. For example, Visual Basic programmers who are used to a supportive graphic interface often recoil at command line compilers. Thankfully, an ever-increasing number of graphic environments make the development of software child's play. In this article, we will present a small selection of these integrated development environments (IDEs).

Invasion

The line-up for our IDE test is as follows: Anjuta, in the current beta version 0.1.8, KDevelop 2.0.2, KDE Studio Gold 3.0, as well as Kylix 2. In contrast to Anjuta and KDevelop, the latter two environments are commercial software. KDE Studio from "theKompany" is available in a no-frills, Free, Open Source version, whereas Borland's Kylix may only be used free of charge for non-commercial projects.

The operation and user-interface of all the programs presented here relies rather strongly on the relevant windows applications. The concept of the project serves as the working basis for all the candidates. Usually, this is nothing other than a collection of all the files that are needed for the new program. This applies not only to the source code, but also to the used libraries and the documentation. Some IDEs even permit the administration of several programs and libraries within a project.

With the exception of Kylix, the environments do not have their own compiler (nor the accompanying help programs). Instead, they all (without exception) access the appropriate GNU command line tools. They could therefore be defined as more of a central, graphic attachment than as a standalone product.

Those who have migrated from the development environments of Microsoft and Borland under the Windows operating system will miss the integrated dialog editors. These provide assistance in the creation of windows, dialog boxes and the contents thereof – similarly to in a painting program. If offered at all, the current IDEs access external software for this.

If this is not installed on your computer, the necessary source code must be manually entered by the programmer – costing valuable time. It is worth pointing out that all the development environments, with the exception of Kylix, enable the direct creation of distributable pages, for example in the rpm format, which is certainly something worthy of praise.

Despite the features our test candidates have in common, the four applications differ widely from each other. Where these differences lie will be revealed in the following sections. At the same time we will present the individual IDEs in detail.

Anjuta 0.1.8

Anjuta comes under the GPL development environment. Its current version is in the beta phase and is therefore not yet completely refined. This makes it the youngest of the projects presented here, but nevertheless it already gives a very promising impression.

Anjuta, in its current version, aims at the development of **Gtk**, i.e. GNOME-based applications. It therefore comes as no surprise that it was created on this base. Anjuta does not directly support applications that use **Qt** or **KDE.** All assistants and assistance entries are completely aligned to Gtk and GNOME, however if you conscientiously ignore these, you will have no problems creating Qt-based programs. This is, by and large, a rather awkward way of doing things, and it is simpler for Qt or KDE developers to use KDevelop or KDE Studio.

Anjuta understands the programming languages C and C++. The compatibility with further languages, such as Java for example, is in the planning stage at present, and some are already partly supported.

In the creation of a new project, an assistant helps the user, leading him to the desired target in few steps. In our test, it took a mere six mouse clicks to create a skeleton of a new application. Anjuta automatically stores all projects in the *Projects* directory, though this path specification can be changed in the settings.

Gtk, Qt Two libraries, which programmers can use in their own applications. They supply graphic objects e.g. menus or dialog windows. In this way, working with windows is made easier for programmers. Gtk comes under GPL, and Qt is likewise available under this license.

Hello World program
A classical programming example, which crops up in many books introducing a programming language. It is an application whose sole purpose is to output the text "Hello World".

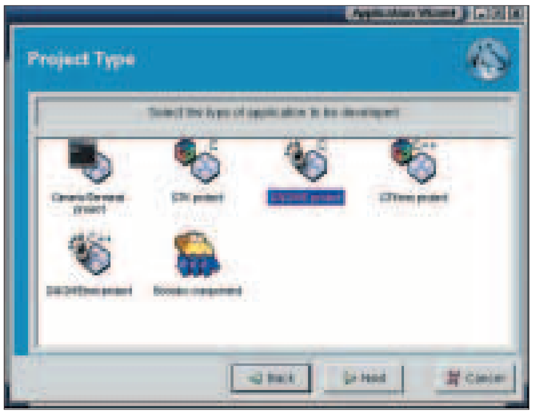
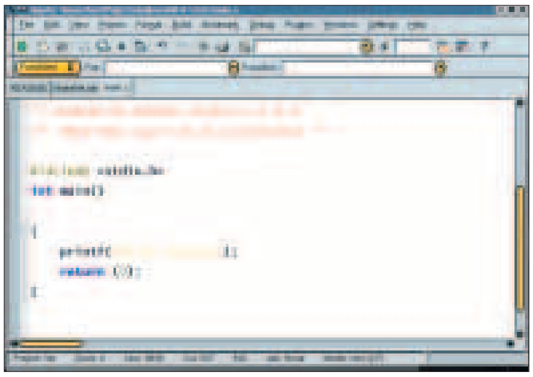


Figure 1: Anjuta's Application Wizard helps the user when creating a new project

The number of created files cannot be moaned about: A simple **"Hello World" program** in C needs a minimum of one, but usually two files (a "make" file for the compiling process and a second file with the actual source code). This is naturally without the otherwise usual documentation in form of the mandatory README and INSTALL files.

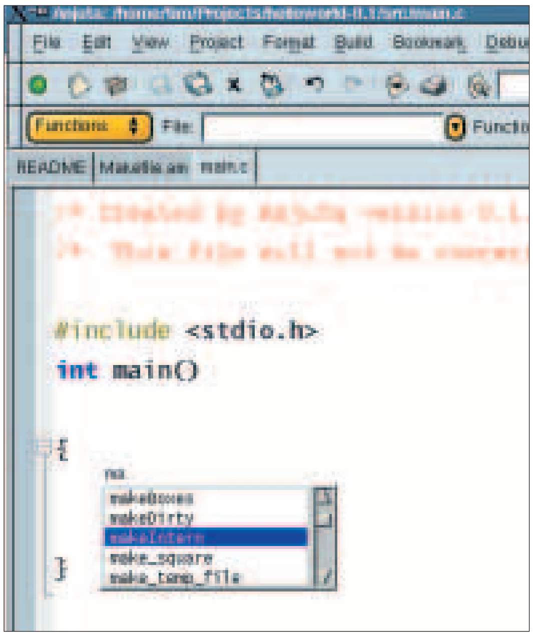
Under the standard settings, Anjuta creates three directories with 64 files altogether. Beyond that, we recommend that Anjuta's main window is not enlarged to cover the entire display. The reason for this is that the software likes to hide software messages and additional important dialog windows in the background.



```
#include <stdio.h>
int main()
{
    printf(                    );
    return (0);
}
```

Figure 2: Anjuta's main window. A "Hello World" program is given as an example in the editor

The editor is very user-friendly for processing source code. By pressing Alt+Enter, even an auto-completion can be activated. This pops up a list of all functions that are available at the current cursor position. As well as this, Anjuta loads and processes other text files of the most diverse formats, such as Java source code or LaTeX files. For a multitude of title formats, the development environment constructs appropriate templates for coloured emphases. We were also very impressed with the ability to quickly fade functions and classes in and out with the plus and minus symbols to the left of the input code. This promotes clarity in projects of all sizes.



```
/* Created by Anjuta version 0.1
/* This file will not be overwr

#include <stdio.h>
int main()

{
     ma
     makeBoxes
     makeDirty
     makeIntern
}    make_square
     make_temp_file
```

Figure 3: The automatic completion in Anjuta

The available functions offered by Anjuta come to an end here; it unfortunately does not support group work or data exchange, as yet. There is however easy access to the debugger Gdb, a standard program for the detection of errors. The graphic creation of an application surface, based on Gtk, takes place through the external program Glade.

The offered assistance is merely sufficient and is essentially limited to short descriptions of the most important menu options. Anjuta accesses external sources for the documentation of the library functions.

KDevelop 2.0.2

In the attempt to copy KDevelop's rpm archive to our test computer with SuSE-Linux, we were more than exasperated: our package manager pointed out a total of 20 uninstalled packages. Most files thereby referenced the Docbook documentation system. This provides KDevelop assistance in executing all documentation functions.

If the packages refuse to be installed, then the development environment automatically used Yast2 on its first starting and therefore undertook the installation of the missing packages itself. Beware those who are logged in as root though: no direct warning or confirmation through the user exists. With an inserted installation CD, it only takes a few seconds and the hard drive is a few megabytes fuller.

The first impression is confirmed directly after opening: KDevelop unites a large number of external programs under one surface. Many familiar external programs repeatedly appear in the menus. Thus, for the graphic design of dialog windows, Qt-Designer will be started. This software is a creation of Trolltech, the manufacturer of Qt.

The surface, in comparison with Anjuta, leans more strongly towards the Windows models.

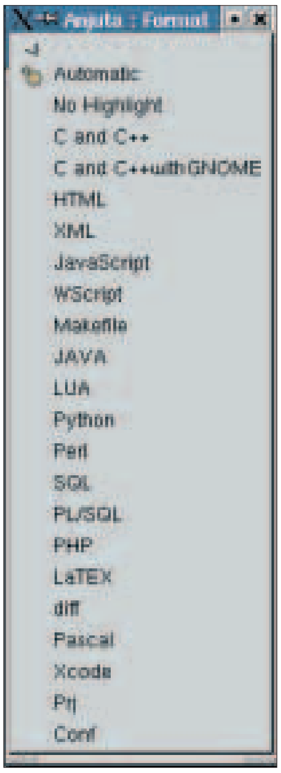


Figure 4: The Anjuta editor currently supports the all these formats

However, it seems overloaded and acts somewhat chaotically. The standard three-compartment main window contains a view in the left border, which, depending on register page activated there, permits different views of the current project. One can also access the very detailed assistance, which even offers a complete language reference for C++.

External sources are however accessed for the documentation of the Qt and KDE classes. An editor window is located on the right-hand side, which apart from editing the source code, also undertakes the display of other documents, such as assistance files. Like so many other elements of the surface, the behaviour of this window can be freely configured. At the bottom border of display is a status window, by which different message types over different register pages can be surveyed and assessed.

Altogether, KDevelop emphasises integration very strongly. This should come as no surprise, after all the development environment comes from the KDE project, whose KDE desktop pursues the same goal.

It is also noticeable here that the roots of KDevelop are found in the KDE project. Beside the many KDE and Qt program variations, the development environment unfortunately offers only one type of application, based on the competing GNOME libraries.

In contrast to Anjuta, KDevelop is thus an ideal candidate when it comes to the creation of KDE or Qt-based C or C++ programs. KDevelop does not speak any other languages or dialects. Fortunately, the assistant offers many possibilities during the set up such as the configuration of the version management tool CVS.

This enables several people to simultaneously work on the same project. KDevelop and Kylix are the only programs presented here, which allow a team to work on one project in this way. Apart from the pure source files, the assistant can even create the documentation belonging to the program. This can be done for function and class documentations (using its own source code) and for user manuals. For the latter however, some knowledge of the documentation system (used by KDevelop especially for this purpose) is necessary.

Once all development packages are installed, the entire software project can thus be created and administered within one surface. Just like Anjuta, KDevelop does not scrimp on the number of newly created files. This not withstanding, the overview remains intact due to the different tree views of the project in the left window section.

The integrated class browser is likewise a success. This allows all the classes used in the project (complete with their attributes and methods) to be easily seen and manipulated. An assistant even takes over the creation of new classes by generating the code frames and the associated files.

Methods and attributes can then be added or deleted using the appropriate dialog window. In doing this, the right mouse button proves to be a real magic wand. Methods or attributes, entered into the code by hand, are automatically transferred into the addressed views after going through a translation procedure. The graphic class opinion is likewise very helpful. This is a window, in which the class hierarchies are clearly represented in form of a diagram. The depiction would however be desirable in the popular UML notation. All in all, KDevelop contains many useful functions, which are unfortunately hidden to the user at first glance.

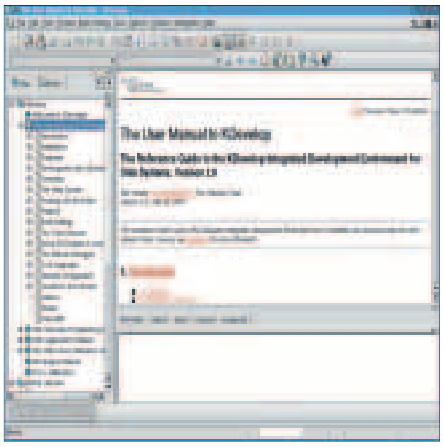


Figure 5: The main window of KDevelop. A help document has been selected for display

The bypass of the console messages into the different status window registers takes time getting used to. The product of our test program was thus rerouted into several registers. This procedure has however the advantage that the messages are outputted clearly and sorted according to type. The text editor does not approach the ease of Anjuta, problem-less operation is however ensured.

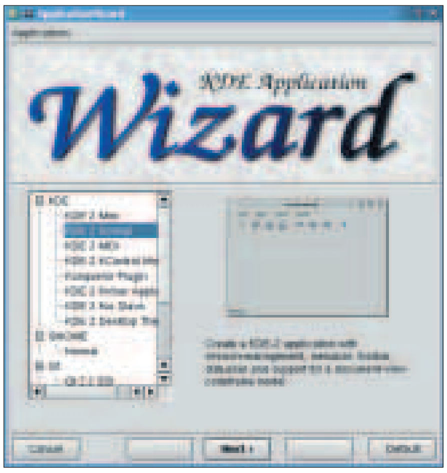


Figure 6: KDevelop's application Wizard

KDE Studio Gold Version 3.0

KDE Studio was developed by "theKompany" and is available in two versions. KDE Studio (published under the GPL) is the Free, Open Source version of the commercial KDE Studio Gold. The difference between the versions is in the offered range of functions.

Beyond that, the commercial version is the only version that will be developed and supported by theKompany from now on. We will be taking a look at the test version of KDE Studio Gold, which can be downloaded free of charge from the manufacturer's homepage. It offers the full function range and is restricted only to a limited duration of use of 15 minutes. The full version is available for approximately US$25.

Directly after the start users are welcomed by an assistant, which, among other things, enables the creation of a new workspace. KDE Studio differs from the other programs in this test by referring to a project as a workspace (working environment). KDE Studio uses the name project to define a subgroup of a workspace. This means a project can be a library or a new program.

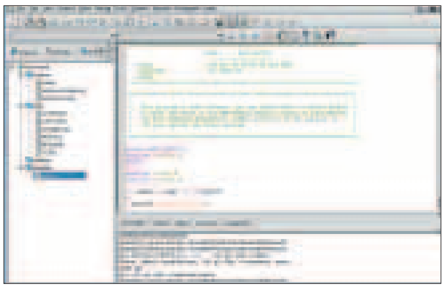


Figure 7: KDevelop displays an open project in this tree view

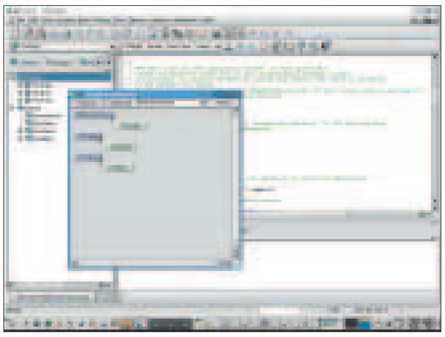


Figure 8: The different class views in KDevelop

When creating a new project, it quickly becomes clear that KDE Studio also puts its accent on C and C++ development using a Qt/KDE base. This IDE offers no option of creating GNOME applications. What remains however is the possibility of creating a custom project by hand.

In the generation of a small terminal program, KDE Studio, in contrast with the competition, only creates one copy of the most necessary scripts and files. Somewhat too economical perhaps: the source coding file with the ever-needed main() function must be manually created. The application only offers templates for a KDE or Qt program.

The surface of the main window resembles that of KDevelop, the difference being that it seems less cluttered. KDE Studio offers a tree representation of the project in its upper left-hand window, though assistance files cannot be displayed there. A status window is located in the lower section and the upper right-hand part is where the source files are worked on. One can switch backwards and forwards between different, opened files with the help of the register. The editor offers the standard fare and, like Anjuta, offers the possibility of selectively fading functions and classes in and out.

KDE Studio does not achieve the function range of KDevelop. It does, on the one hand, have a class browser (called class explorer here) and a graphic view, but these need to be explicitly called up from the menu. On top of this, they look a little showy and unclear.

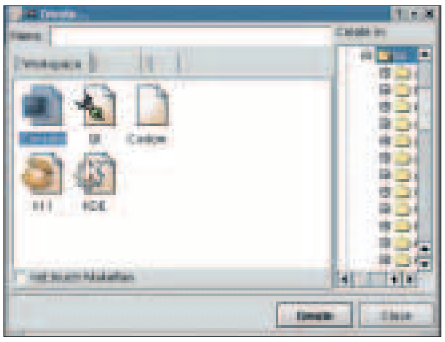


Figure 9: KDE Studio Gold's assistant

There are absolutely no help mechanisms for the creation of classes, like those integrated into KDevelop. The complete documentation was likewise conspicuous in our demo version by its absence. This seems strange considering theKompany advertises the purchase of KDE Studio with the promise of complete documentation.

Few other interesting functions were to be found in KDE Studio Gold 3.0, apart from the mandatory debug and compiler options.

Kylix 2 Open Edition

Kylix 2 breaks the mould in several ways. On the one hand it comes from the highly regarded compiler manufacturer Borland, who already has many years of experience with different development environments on the Windows OS. On the other hand, this is a commercial product.

Contrary to KDE Studio Gold, Borland publishes a downsized version as a Free, Open edition. As the name suggests, only Free, Open Source projects under the GNU license may be created with it.

Those who want a greater function range or commercial software, are given two rather costly alternatives. The first of these is the professional version, at a price of 325 euros from Borland's Web site. The expensive company version is the second possibility, and yours for 2585 euro. The free Open Edition is usually sufficient for private users. Heed must be paid, that programs compiled with this version have an appropriate note inserted at their start. The exact differences between the individual Kylix versions can be found on Borland's Web site.

Kylix brings with it a conversion of Delphi, the popular Windows development environment. Similarly to Microsoft's VisualBasic, this is a complete self-development from Borland. Programs written with Delphi, as well as the Kylix environment use both their own language (named Object Pascal) and the program libraries created by Borland.

Domestic applications can be created relatively quickly in this way, but this is dependent on the manufacturer and the libraries thereof. With Kylix, Borland is pursuing the target of being able to make an application that was developed under Delphi, available under Linux simply by recompiling it. The reverse of this is naturally also possible.

For this purpose, Borland not only transferred the CLX class library to Linux, but also made the compiler and the IDE available under Linux. A disadvantage in comparison with the other development environments is the closed source strategy, meaning only the source code of the CLX libraries is laid bare. After the start, the Open Edition requires the entry of a registration code. This is mandatory, but can be taken from *http://register.borland.com* for free. If the entry of the two codes is correct, a familiar picture awaits the experienced Delphi user. It is no coincidence that the surface resembles that of its big brother from the Windows world.

As in the first Kylix version, the surface from Delphi has been ported over and brought to life under Linux with the Wine emulator. This unfortunately has the

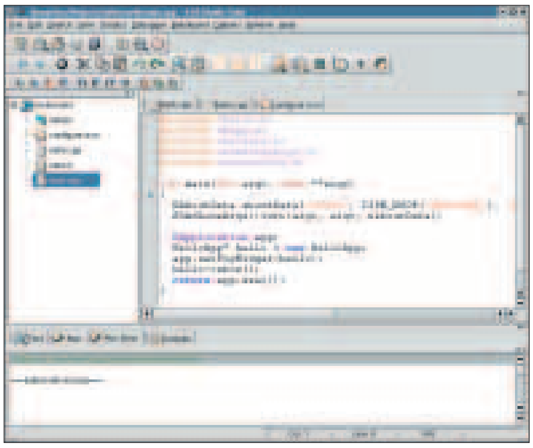


Figure 10: KDE Studio Gold's windows in action

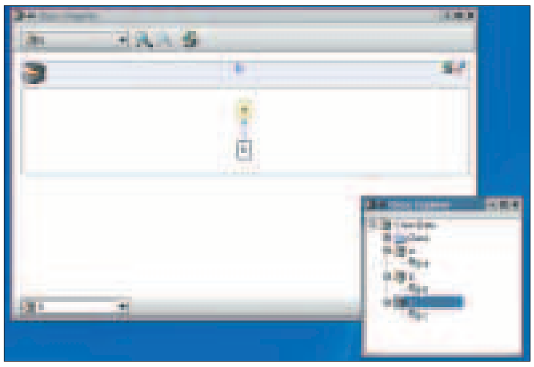


Figure 11: KDE Studio Gold class views. In the picture, the classes b and c were derived from class a

unpleasant side effect that the entire environment reacts somewhat sedately – lets just say it's not one of the fastest in its class. The programs created through Kylix are fortunately native Linux programs and not dependent on Wine.

Delphi users are however not the only ones who will feel right at home. It is really noticeable that the manufacturer of this product has had many years of experience under his belt. Directly after the start, an empty project, consisting of an empty dialog window, is opened beside the main window. This space can be used to create and arrange appropriate items, like switches or lists, just like in a painting application.

A bar in the top margin presents all the available elements, appropriate at this juncture. Adjustments of the respective characteristics take place via the ever-present object inspector. This window is commonly located on the left-hand side of the display.

The user creates graphic elements in windows defined as forms, and while this is happening, Kylix automatically generates the suitable source code in the background. The user must then merely fill this out in another editing window. A useful overview over all the classes and components used in the project, like the diagram offered in KDevelop, is sadly missing here.

After one has acquainted oneself with all the elements of the window and is familiar with the language Object Pascal, then the creation of a complete application with Kylix occurs very quickly and effectively. This is, in no small part, greatly aided by the user-friendly window. This window can be adapted to one's own needs in almost every fathomable area. All settings can be saved and called up again at the push of a button. Different adjustments for different projects are thus no longer a problem.

Kylix is the only program presented here that supports teamwork. In contrast to the competitor, data is not exchanged by means of the Free document management system CVS, but through its own repository.

The assistance provided was the only one in our test of any real quality. Detailed and complete passages left hardly any questions unanswered. The assistance system used is again unfortunately a Borland internal development and has a striking resemblance to its Windows counterpart. The assistance documents are supplemented by attached demo-programs, which was unique in our test.

Result

A comparison is rarely clear cut, as each presented development environment has its own target group. C or C++ programmers, who want to predominantly write Gtk-based programs intended for GNOME, should cast an eye on Anjuta. KDevelop should however be the first stop for KDE and Qt programmers. KDE Studio unfortunately disqualifies itself, as its function range lags far behind KDevelop and the asking price is disproportionate. Programmers coming across from Delphi, should take a closer look at Kylix.

Programmers, who have collected their first programming experience under Windows with VisualBasic, still need to be forced to transfer to something else. Those who don't want to descend into the crypt depths of C and C++ just yet, should instead try out Kylix for size. For beginners, this is the pick of all solutions presented here.

If you only write small programs, you should really consider whether the use of a large development environment is worthwhile at all. Small projects often don't need the packages required by the environments. In such cases, it can be like shooting a canon at a sparrow – overkill to say the least.

Finally, it must be said that apart from Kylix, none of the development environments achieve the programming comfort that Windows users are familiar with. Difficulties in converting from Windows should therefore be taken into account.

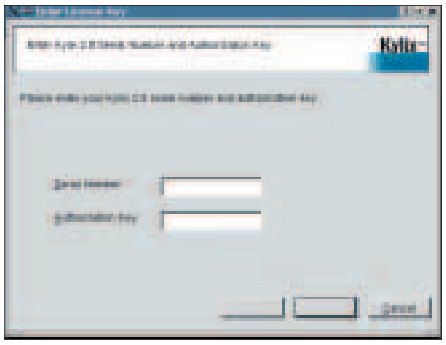


Figure 12: The registration dialog of Kylix 2

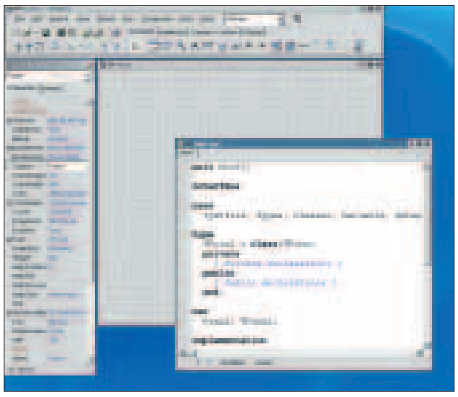


Figure 13: The start window of Kylix 2: The source code of the window in the background is directly transferred into the appropriate source code, seen in the editor window

Info

Homepage of Anjuta *http://anjuta.sourceforge.net*
Homepage of theKompany, as well as KDE Studio Gold *http://www.thekompany.com*
Homepage of the KDevelop project *http://www.kdevelop.org*
Homepage of the company Borland *http://www.borland.de*
Central location, where the codes for the Open Editon of Kylix 2 can be found *http://register.borland.com*