



Halten **GPL-Programme** wirklich Einzug in die professionelle **Programmentwicklung**? Ja, denn der **Linux-Programmierer** kann mit guter **Unterstützung** rechnen. Sowohl KDevelop als auch Glade **zielen in Richtung** schnelle **Entwicklung** der GUI. Stichwort hier: RAD - Rapid Prototyping **Development**.



C++-Entwicklung mit Linux

Mit KDevelop und Glade arbeiten

ROMAN JORDAN

Lange Zeit galt Linux als Betriebssystem für Freaks. Wer eine zusätzliche Funktion in seinem Programm benötigte, schrieb sich diese selbst. Gleiches bei den grafischen Programmen: X-Windows stellte und stellt "nur" standardisierte API Funktionen zur Kommunikation der Anwendungsprogramme mit der X11-bzw. Hardwareumgebung zur Verfügung. Jeder Programmierer generierte sich seine grafischen Elemente selbst bzw. griff auf eine Unmenge sogenann-

ter Toolkits zurück. Programmierer, die schon länger unter Linux arbeiten, kennen vielleicht die sogenannten Athena Widgets - ein Schritt in Richtung einheitliches Aussehen.

■ Zweimal Standard

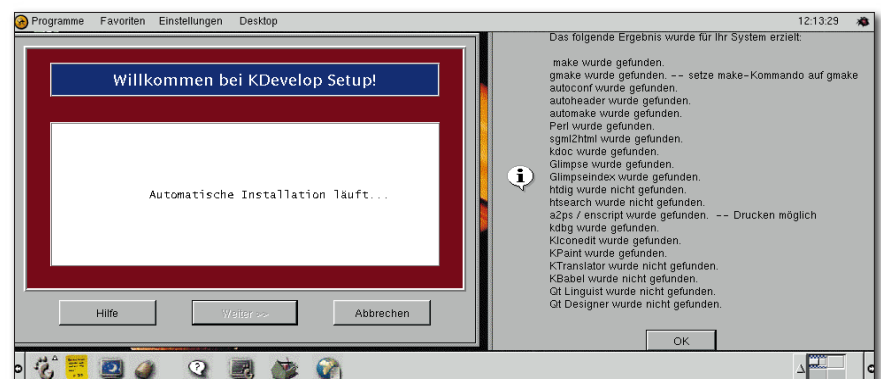
In der heutigen Zeit, in der sich Tausende mit der grafischen Linux-Programmierung beschäftigen, würde diese Vorgehensweise eine Unmenge verschiedenster Styles ergeben. An diesem Punkt setzen die Entwickler von KDE bzw. GNOME an. Sie haben es sich zum Ziel gemacht, einheitliche, lei-

stungsstarke und vor allem benutzerfreundliche Toolkits zu schaffen. Während GNOME auf das GDK/GTK zurückgreift, verwendet KDE das Qt-Toolkit der Firma Trolltech [6] (Norwegen). Bei ersterem handelt es sich um das GTK+-Toolkit, das bei der Erstellung des Programms GIMP entstand. Es ist vollständig in C geschrieben und gut strukturiert.

Andere Programmiersprachen werden über sogenannte Wrapper adaptiert. Der Part, der C++ unterstützt, ist als GTK- (oder GTKMM) verfügbar. Darüber hinaus gibt es eine Adaption auf die

QUICK INDEX

- **Zweimal Standard**
Unter Linux konkurrieren zwei GUI-Quasistandard-Bibliotheken: GDK/GTK und Qt.
- **KDevelop**
Ein mächtiges Werkzeug, das entsprechend aufwendig zu installieren ist und sehr reichhaltig ausgebaut werden kann.
- **Glade**
Mit Glade wird die Entwicklung von Programmoberflächen einfach.



DER KONFIGURATIONSSASSISTENT begleitet Sie beim ersten Start.



Plattform unabhängige WxWindows-Bibliothek. Auch für GTK+ selbst gibt es Windows-Versionen. GKT+ ist vollständig unter der LGPL erhältlich.

Qt wiederum ist von Grund auf objektorientiert und in C++ geschrieben. In die entsprechenden Bibliotheken wurden zahlreiche Funktionen integriert. Qt ist ebenfalls plattformunabhängig (und darin wohl weiter entwickelt als GDK+), jedoch behält sich Trolltech das Recht auf eine eigene Lizenzierung vor. Die kostenlose Verwendung beschränkt sich auf Open-Source-Linux-Programme. Qt ist neuerdings auch als Qt embedded (es setzt direkt auf das FrameBuffer Device auf, ohne X Windows) für verschiedenste Prozessoren erhältlich.

Der folgende Artikel zeigt, dass die grafische Programmentwicklung unter diesen beiden Oberflächen mit den richtigen Tools unter Linux deutlich an Komfort gewonnen hat. Die Entscheidung für eine Oberfläche schließt nicht die Verwendung der anderen aus, da beide parallel verwendet werden können - vorausgesetzt, die entsprechenden Libs sind verfügbar.

Schließlich werden noch zwei weitere Kandidaten zum Thema IDE vorgestellt. Diese können ebenfalls in zur C/C++-Programmerstellung eingesetzt werden.

KDevelop

Die Entwicklung von KDevelop begann 1998 mit dem Ziel, eine einfach zu bedienende IDE (Integrated Development Environment) für Unix zu erstellen. Heute steht KDevelop unter der GPL und unterstützt alles, was mit C/C++ zu tun hat (z.B. KDE/Qt-, GNOME-, C- und C++-Projekte). Sollten Sie gleich mit KDevelop loslegen wollen, so beachten Sie bitte, dass beim ersten Start die Konfiguration erfolgt. Sie wird durch einen Setup-Assistenten unterstützt. Bei Problemen klicken Sie auf den Hilfe-Button. Die Funktionalität von KDevelop kann durch einige optionale Programme erweitert werden (siehe Kasten 1). Falls diese schon auf Ihrem System installiert sind, werden sie automatisch mit eingebunden.

Ein nachträgliches Konfigurieren von KDevelop ist über "K - Entwicklung - KDevelop Setup" oder einfach "KDevelop - setup" jederzeit möglich. Wer KDevelop aus den Sourcen generieren möchte, benötigt außerdem

- Compiler ab g++/ egcs 1.0.3
- GNU make
- PERL ab Version 5.004
- autoconf 2.13/ automake 1.4
- flex 2.5.4 Qt 1.44
- KDE 1.1.x (Pakete Libs und Base)

Zur Zeit ist Version 1.3 aktuell. Sourcen und Binaries sind unter [1], [2], [3] und [4] zu finden.

KDevelop ist die wohl umfangreichste IDE, die unter Linux zur Verfügung steht. Sie baut auf das bereits erwähnte Qt-Toolkit auf. Bei der Installation wird die (im Standardfall bereits vorhandene) Hilfe zur Qt Lib mit in die IDE eingebunden.

KDevelop zeichnet sich durch Funktionalität und Übersichtlichkeit in der Bedienung aus. Durch die Integration mehrerer Hilfssysteme, ausgereiftes Dateibrowsing, automatische Generierung des Programmskeletts sowie einen integrierten Debugger bzw. eine automatische Dokumentationserstellung ist alles von einer Oberfläche aus zu erreichen. Wer noch keine Erfahrung in der C/C++-Programmierung mitbringt, kann sich (siehe Kasten "Erweiterte Hilfefunktionen") mit Hilfe des optionalen "c_cpp_reference"- Pakets einarbeiten.

Nach der Installation kann das Programm über den

Startbutton (in KDE Umgebung), Menüpunkt "Entwicklung" oder alternativ über den Kommandozeilenauftrag "KDevelop" gestartet werden.

Nach der Konfiguration wird das Programm mit dem Begrüßungsbildschirm gestartet. Nach Wegklicken desselben ist die Sicht auf die umfangreiche Online-Hilfe frei. Wer sich an dieser Stelle ein wenig mit der Bedienung der einfachen C/C++-Programmierung und Q-Lib-Funktionen beschäftigen möchte, dem sei ein Durchklicken unbedingt empfohlen.

In der Installationen Version 1.0 war bedauerlicherweise keine vollwertige deutschsprachige Hilfe verfügbar. Auch der optionale c_c++-Bestandteil [3] der Hilfe war nur in englischer Sprache verfasst. Wer damit nicht klar kommt, kann seine integrierte Hilfe

ERWEITERTE HILFEFUNKTIONEN

1. Entpacken (egal wo)
`'tar xfvz c_cpp_reference-1.0.tar.gz'`
2. Ins Verzeichnis wechseln
`cd c_cpp_reference-1.0`
3. Konfigurieren - dabei wird das Zielverzeichnis automatisch erkannt
`./configure`
4. Generieren und installieren
`make`
`make install`

EINIGE OPTIONALE ZUSATZPROGRAMME

SGML-Tools: Sammlung von Textformatierungsprogrammen, die auf SGML beruhen und eine Vielzahl von Ausgabeformaten erzeugen können.

qt: libs devel ((?)) (inklusive Beispiele und Dokumentation)

kdcc: Tool zum automatischen Generieren von Programmdokumentation. Dabei wird eine spezielle Syntax verwendet. Als Output ist Latex-, man- oder TeXInfoformat möglich.

KDevelop-c_c++_ref: C- und C++-Referenz im HTML-Format. Steht innerhalb der IDE zur Verfügung. Dazu liegt nur eine englischsprachige Dokumentation vor (siehe Hilfekasten).

A2ps 4.11: ASCII to Postscript. Programm konvertiert normale ASCII- in Postscript-Dateien

kdbg: KDE-Debugger, der sich in die IDE integrieren lässt. Sehr hilfreich !

KIconEdit: Dient zum Bearbeiten von Icons.

KDE Libs als Source: (Die Version sollte, um Missverständnisse zu vermeiden, mit den installierten Binaries übereinstimmen). Dient zum Erstellen der KDE-Bibliotheksdokumentation.

glimpse: Sehr leistungsfähiges Index- und Query-Programm. Die Funktion ähnelt der von 'grep'. Ist durch den Aufbau einer eigenen Datenbank schneller. Benötigt Package agrep.

agrep: Funktion gleich der von egrep bzw. fgrep. Jedoch fehlertoleranter und schneller. Wird für glimpse benötigt.

Cvs 1.10.6: Code Version System. Dient zum Verwalten von Programmversionen.

Qt Lib devel: Include-Dateien für die Programmentwicklung mittels Qt. Beinhalten Beispielprogramme (meist unter /usr/lib/qt/examples.a



durch symbolische Links erweitern (die aber dann nur Englisch). Dazu wechseln Sie in das KDE-HTML-Verzeichnis (entweder `"/usr/share/doc/HTML"` oder `"/opt/kde/share/doc/HTML"`), Unterverzeichnis `"de/KDevelop"` (für deutsch), und erzeugen die entsprechenden Links, wie sie im Unterverzeichnis `"en/KDevelop"` existieren. Hier ein Beispiel mit dem Unterverzeichnis Referenz:

```
cd /usr/share/doc/HTML/de/KDevelop
ln -s ../../en/KDevelop/reference
reference
```

Beachten Sie, dass Sie vor dem eventuellen Einspielen der deutschsprachigen Hilfetexte diese Links wieder löschen, sonst werden die englischsprachigen Originale überschrieben. Unterverzeichnisse, die bereits existieren, sollten Sie in der deutschen Fassung belassen.

KDevelop ist in mehrere Bereiche unterteilt. Am auffälligsten ist das Hauptfenster in der Mitte. Hier finden Sie unter verschiedenen Reitern die Editorfenster für Header-/Ressourcen-dateien, C/C++-Dateien und die Dokumentation (unterstützt HTML-Formate). Auf der linken Seite befindet sich ein Fenster mit Baumansichten. Es ist unterteilt in

- CVS, den Klassenviewer,
- LFV, die Anzeige des logischen Dateibaums (geordnet nach Funktion),
- RFV, den realen Dateibaum,
- sowie DOC, eine Gliederung der Online-Hilfe.

Während des Debuggens kommt die Variablenansicht dazu. Dieses Fenster ist mit dem Hauptfenster gekoppelt. Ein Doppelklick auf eine Datei bringt diese zum Bearbeiten bzw. zur Ansicht in das Hauptfenster. In den Ausgabefenstern kann man die jeweils aktuellen Aktivitäten verfolgen bzw. die gesetzten Breakpoints sehen. Bei Fehlern während des Kompilierens kann durch Doppelklick im "Ausgabefenster - Meldungen" an die entsprechende Stelle im Sourcecode gesprungen werden.

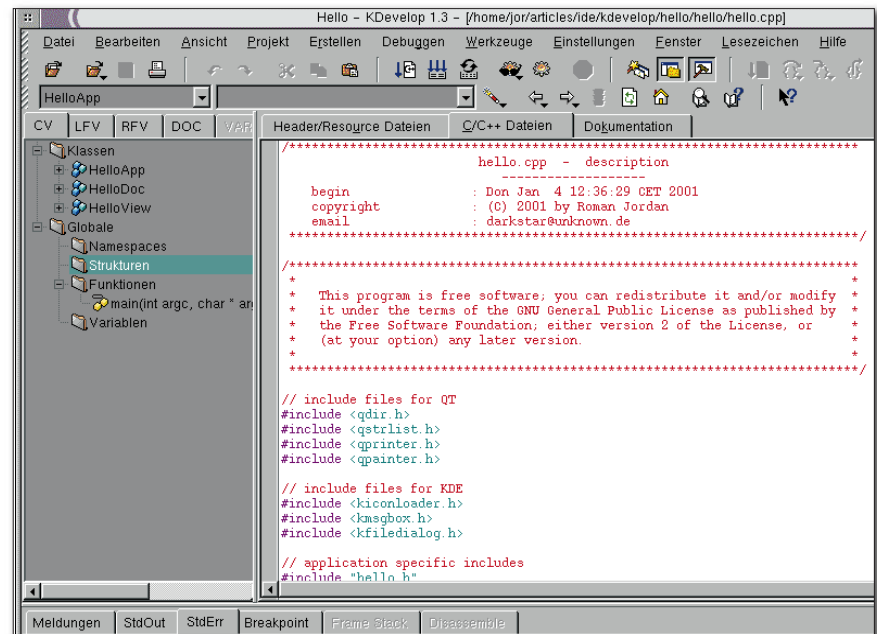
Projekte, die das Leben leichter machen

Die Verwaltung der Sourcen erfolgt unter KDevelop mittels Projekten. Um ein Projekt zu erzeugen, steht ein integrierter Wizard, KAppWizard, zur Verfügung.

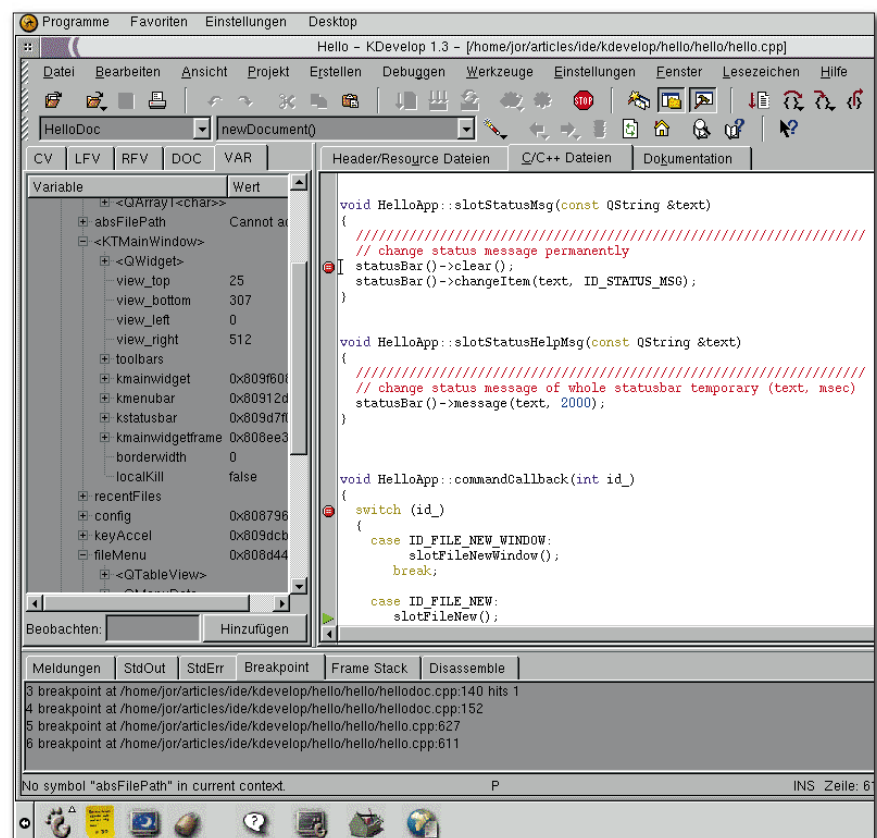
Der Anwender kann zwischen bis zu 14 (Version 1.3) verschiedenen Pro-

grammtypen wählen. So stehen z.B. KDE- (Normal/ Mini), Qt- (normal), Terminal- (C, C++) oder eigene Projekte zur Verfügung. Bei den KDE-Projekten erfolgt eine Unterscheidung zwischen Qt-2.0- und sogenannten normalen Qt-Projekten. Die Struktur

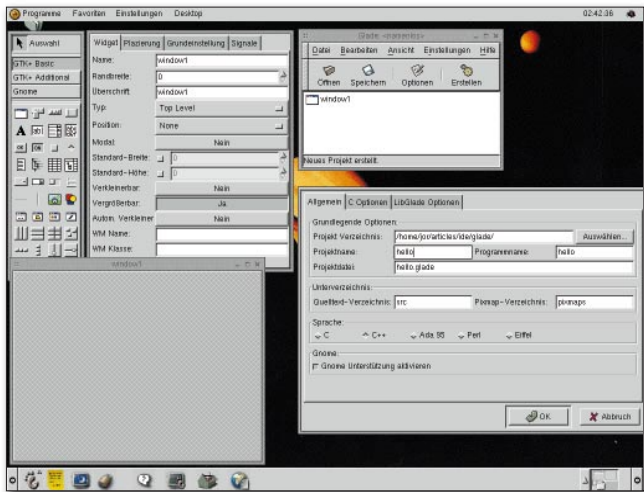
der erzeugten Dateien basiert auf dem Autoconf/Automake-Standard. Bei der Weitergabe des Programms als Paket kann der Empfänger diese mittels Entpacken und Configure-Tool generieren. Damit ist gewährleistet, dass dieses Programm auf unterschiedlichen Plattfor-



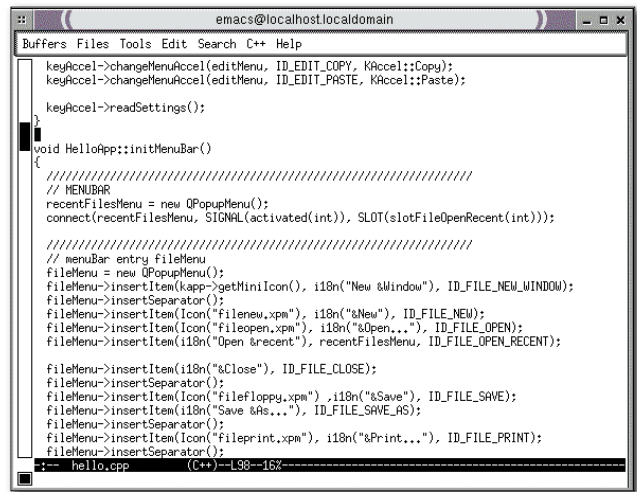
DAS ERSTE AUTOMATISCH mit KDevelop generierte Projekt.



DER INTEGRIERTE DEBUGGER bei der Arbeit, im unteren Fenster die Meldungen.



BEGINN EINES PROJEKTS im User – Interface – Builder Glade



DER GUTE, alte Emacs bei der Arbeit an einem Programm.

men lauffähig ist – vorausgesetzt, die entsprechenden Libs und Tools sind auf diesem System vorhanden: Dies wird durch "configure" überprüft.

Ein neues Projekt wird in sechs einfachen Schritten erzeugt: Wahl der Projektart, Auswahl des Icons, bei Bedarf Integration einer Versionsverwaltung, Editieren der Vorlagen für Header- und C++-Dateien. Ihr Name und eventuelle Lizenzhinweise sind so bei jeder neuen Datei des Projekts bereits vorhanden. Auch bei der Erstellung eines Projekts aus schon vorhandenen Quellen kann diese Option genutzt werden. Beim Importieren werden Ihre gewünschten Köpfe eingetragen. Im letzten Schritt wird die Dateistruktur erzeugt.

Durch Klick auf den Reiter "C/C++" gelangt man in das Editorfenster.

Der Editor unterstützt Syntax-Highlighting und erinnert von der Bedienung her stark an andere bekannte Editoren (damit ist nicht Emacs gemeint). Wer möchte, kann seine eigene Tastenbelegung erstellen. Innerhalb des Editorfensters darf auf die Online-Hilfe zugegriffen werden. Dazu den gewünschten Befehl oder das Schlüsselwort markieren und dann mittels rechter Maustaste auf Suche gehen. Sollte es sich um einen Qt- oder KDE-Befehl handeln, wird die entsprechende Hilfeeite im Dokumentationsfenster angezeigt. Über ein Vor- und ein Zurück-Icon kann innerhalb der Dokumentation geblättert werden (wie unter HTML gewohnt).

Doch was ist der beste Programmierer ohne Dokumentation? KDevelop beinhaltet fünf Online-Handbücher im HTML-Format. Wer die Qt Libs als

Devel mitinstalliert hat, erhält zusätzlich eine Qt-Referenz. Darüber hinaus können Sie jederzeit neue Dokumentensätze erstellen lassen. Auch die eigene Programmdokumentation ist kein Problem. Über den Menüpunkt "Projekt – API Dokumentation erstellen bzw. Benutzerhandbuch erstellen" sind zwei Ausgabevarianten möglich. Darin enthalten sind die Klassen und die einzelnen Funktionsaufrufe mit den im Quelltext stehenden Kommentaren. Eine Installationsanleitung wird ebenfalls automatisch erzeugt.

Alles inklusive

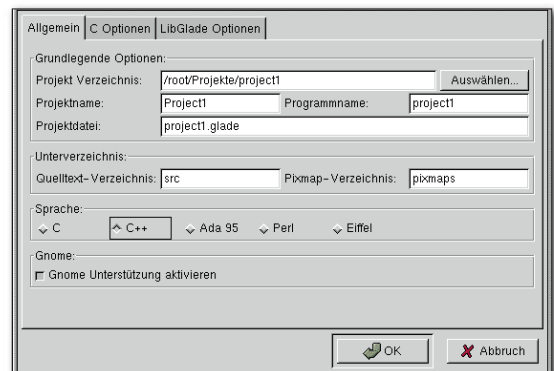
Wenn man die Applikation um ein paar selbstdefinierte Dialoge erweitern möchte, greift man am besten zum integrierten Dialogeditor. Dieser verfügt über zahlreiche bereits durch das Qt-Toolkit vordefinierte Widgets. Diese werden durch einfaches Klicken platziert.

Auf der Suche nach eventuellen Programmfehlern steht der integrierte Debugger Kdbg zur Verfügung. Man startet ihn entweder über das Menü oder über das Zahnrad-Icon mit Brille. Der Debugger ist so hervorragend integriert, dass keine Umschaltung des Fensters erfolgt. Die Quelltexte bleiben in der bekannten Form erhalten. Breakpoints werden durch Klick an die linke Seite des Editorfensters gesetzt bzw. gelöscht. Über das Debugger-Menü sind zusätzliche Ansichten wie Prozessorregister oder die bereits geladenen Bibliotheken möglich.

Glade

Im Gegensatz zu KDevelop ist Glade ein reiner User-Interface-Builders, basierend auf GTK+ und GNOME. Durch sogenannte ADD ONS (oder language bindings) [8] können C++, Ada95, Perl, Eiffel und andere Sprachen generiert werden.

Die Liste der unterstützten Programmiersprachen ist keineswegs fest definiert und wird ständig erweitert. Glade dient weniger zur Erstellung kompletter Programme als vielmehr einer schnellen und einfachen Entwicklung von Programmoberflächen, sprich Fensteranordnung, Signalverknüpfungen usw. Dabei kann das Senden von Signalen ebenso beeinflusst werden wie die Reaktionen des Widgets (Fensters) beim Signalempfang. Die einzelnen Signale werden komfortabel über Auswahlmensüs mit entsprechenden Routinen verknüpft. Der Programmierer muss sich nicht mehr um die Eingabe und Zuordnung von Signalen in seinen Quellen kümmern. Die Verwaltung der Source-Dateien erfolgt ebenfalls über Projekte. Ein Projekt besteht aus einem Main-



BEI GLADE MUSS die Option C++ erst aktiviert werden.



Window, in das je nach Bedarf weitere grafische Elemente (Dialoge, Menüs...) eingefügt werden.

Nach dem Start von Glade öffnen sich drei Fenster: das Hauptfenster mit der Projektverwaltung, ein Fenster für Eigenschaften und eines für die Palette. Das Projektfenster beinhaltet alle zum aktuellen Projekt gehörenden Windows, Popups, Dialogboxen und Menüs. Über die Menüleiste stehen Kommandos zum Speichern, Generieren von Projekten sowie zur Einstellung von Eigenschaften zur Verfügung. Die Widgets sind im Palettenfenster angezeigt. Ein neues Fenster wird durch Klicken auf Windows (oben links) im Basic Set erzeugt.

Unter GTK kann pro Fenster nur ein Widget eingefügt werden. Wer mehr benötigt, muss dazu Containerklassen (vertikal, horizontal oder Tabelle) einfügen. Diese Container sind ebenfalls im Basic Set vorhanden. Im Set "GTK+ Additional" findet man einen kompletten Kalender, Widgets zur Kurvendarstellung, die Option, selbstdefinierte Widgets einzubinden, und noch einiges mehr. Das Set "GNOME" stellt komplexere Dinge wie etwa Ziffernblatt, Ausgabefenster oder Taschenrechner zur Verfügung. Um eines dieser Widgets in seine Applikation einzubinden, genügt ein Doppelklick. Ein Vergleich mit den Widgets aus Qt zeigt enorme Ähnlichkeiten. Grafische Applikationen verlangen nun einmal eine gewisse Auswahl an Widgets. Diese müssen zur Verfügung stehen.

Das Bearbeiten des Widgets erfolgt im Fenster "Eigenschaften". Je nach Widget-Typ sind unterschiedliche Einstellungen möglich. Diese sind in sechs Bereiche (über Reiter getrennt) unterteilt und umfassen Punkte wie allgemeine Eigenschaften, Platzierung, Basic, Style, Accelerators oder Signalverknüpfung.

Einer der Hauptunterschiede zu KDevelop ist das Fehlen eines eigenen Quelltexteditors. Der Ansatz von Glade zielt in erster Linie auf die bereits erwähnte Generierung der GUI. Hier scheiden sich die Geister. Einige sind der Meinung, dass ein integrierter Editor unbedingt Bestandteil einer IDE sein sollte. Andere sprechen sich für die Nutzung bereits vorhandener Editoren aus, etwa die Unix-typischen Editoren wie Emacs oder VI. Ein durchaus stichhaltiges Argument.

Wer mit C++ arbeiten will, muss im Projektfenster die Optionen C++ aktivieren.

Voraussetzung ist, dass Glade und die entsprechenden Libs installiert sind. Wer glade – aus den Sources generiert, sollte anschließend ein "strip" ausführen, da glade – extrem viel Speicher für sich beansprucht.

Wie bei KDevelop werden mit Hilfe von Glade plattformunabhängig (via autoconf/ automake) Quellcodes gene-

auf die einfache Erweiterbarkeit der Funktionalität. Versionsverwaltung ist ebenfalls kein Thema. Diese wird schon seit langem unterstützt.

Softwareprojekte gibt es nicht direkt. Alternativ dazu können unterschiedliche Projekte in unterschiedlichen Verzeichnissen gleichzeitig über mehrere Fenster (sogenannte Buffer) bearbeitet werden. Die einzelnen Projekte können unterschiedlichster Natur sein, etwa ein Verzeichnis für ein HTML-Projekt, ein anderes enthält ein Tex-Projekt und ein drittes ein Crosscompiler-Projekt. Der Aufruf von "compile" erzeugt unter Zuhilfenahme des jeweiligen Makefiles die gewünschten Aktionen. Themen wie Syntax-Highlight (z.B. C, C++, HTML, Tex...), Sourcecode-Formatierungen (auch automatisch), integrierter Kalender- und Mailfunktionalität, Hexadezimalrechnen etc. sind alle vorhanden. Über Kommandos ist jeder Debugger adaptierbar. Ein wirklich sehr guter Kandidat ist "ddd" – ein Debugger, der an Funktionsumfang und Bedienerfreundlichkeit nur schwer zu übertreffen ist.

Einsteiger müssen sich jedoch lange einarbeiten, bis die Tastaturkürzel sitzen. Als weitere mächtige IDE sei hier noch der SourceNavigator von RedHat erwähnt. Er unterliegt der GPL und soll möglicherweise Bestandteil der nächsten RedHat-Version werden. Der SourceNavigator bietet in Kombination mit dem auf den GNU-Debugger basierenden Insight ebenfalls eine mächtige Alternative in der Programmentwicklung. Weitere Bestandteile dieser IDE sind Syntax-Highlight, Adaption anderer (z.B. Crosscompiler), sehr einfache Klassen- und Dateibrowser sowie eine Projektunterstützung. Vorhandene Softwareprojekte können ohne großen Aufwand in Projekte übernommen werden. Mehr Informationen zu diesem Thema ist unter [9] zu finden.

UR

BEGRIFFE

IDE: Integrated Development Environment. Vereint die zum Programmieren notwendigen Tools wie Compiler, Linker, Debugger und Editor unter einer Oberfläche.

GUI: Graphical User Interface. Die grafische Benutzerschnittstelle beschreibt das Aussehen und die Bedienung eines Programms. Wird z.B. die Qt Lib als Basis verwendet, ähneln sich die generierten Programme vom Aussehen her stark.

riert. Wer seine Sources per CVS verwalten möchte, hat auch hier keine Probleme. Es ist nicht ausgeschlossen, dass Glade in Zukunft als Bestandteil in die gIde (GNOME IDE) mit eingeht. Diese befindet sich jedoch erst im Beta-Stadium. Auf die Zukunft darf man jedoch sehr gespannt sein.

Emacs und RedHats SourceNavigator

Für einige Linux-Programmierer, vor allem diejenigen, die schon länger dabei sind, gibt es natürlich nur eine IDE, und die heißt Emacs.

Obwohl diese IDE schon ein wenig in die Jahre gekommen ist, besitzt sie Funktionen, die selbst die unter Windows ansässigen IDEs lange Zeit vermissen ließen und teilweise noch vermissen lassen. Das bezieht sich vor allem

INTERNETADRESSEN

- [1] <http://www.KDevelop.org>
- [2] <http://www.kde.org/current.html>
- [3] <http://www.cs.uni-potsdam.de/~smeier/KDevelop/index.html>
- [4] <http://rpmfind.net>
- [5] <http://www.kde.org>
- [6] <http://www.trolltech.com>
- [7] <http://glade.pn.org>
- [8] <http://home.wtal.de/petig/gtk>
- [9] <http://sources.redhat.com>