

Portierung auf plattformübergreifenden Software - Überblick und Fallbeispiel mit Qt

7. Juni 2004

Rechtlicher Hinweis

Dieser Beitrag ist lizenziert unter der UVM Lizenz für die freie Nutzung unveränderter Inhalte.

Zusammenfassung

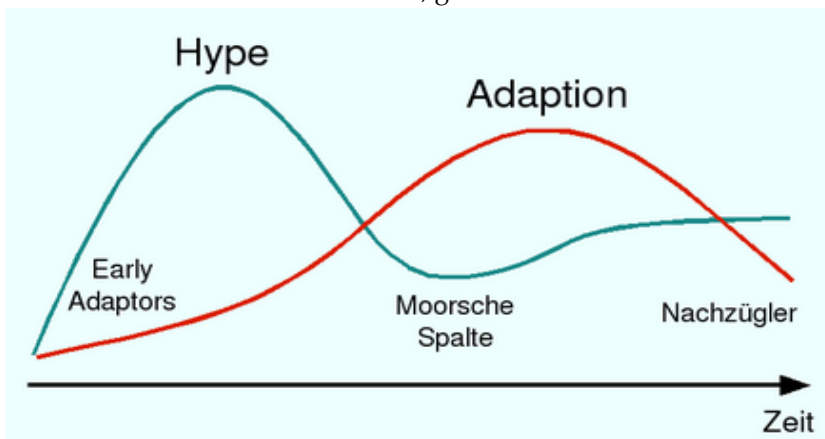
In der aktuellen Situation, in der immer mehr öffentliche Einrichtungen und Firmen ihre Server und Desktops auf Linux umstellen, hat sich ein stetig wachsender Markt für Software entwickelt, die auf dem Linux-Desktop läuft. Hinzu kommen Heimanwender, die als Privatkunden ebenfalls einen Markt für Linux-Software darstellen. Schon heute gibt es einen ansehnlichen Kundenkreis.

Die oft gewünschte oder sogar zwingend notwendige Portierung von vorhandener und über einen langen Zeitraum gewachsener Software stellt sich jedoch zumeist als nicht trivial dar. Gefordert ist, dass die zu realisierenden Linux-Lösungen eine zu den von der bisherigen Windows-Umgebung bekannten Applikationen möglichst äquivalente Funktionalität haben. Die entstehenden Applikationen sollen sich dann nahtlos in die Zieldesktops – sowohl unter Linux als auch unter Windows – integrieren lassen.

Der Vortrag beleuchtet ausgehend von einer eingehenden Diskussion der Anforderungen verschiedene Möglichkeiten der Neuentwicklung von Software, aber insbesondere auch der Portierung existierender Programme. Hier bieten plattformübergreifende Toolkits und OpenSource-Software die Möglichkeit zukunfts-sichere Lösungen zu schaffen. Sie ermöglichen es mit nur einer Code-Basis, mehrere Plattformen mit voller Desktop-Integration zu bedienen. Zur Sprache kommen auch Bezugsmöglichkeiten für OpenSource-Implementierungen, zu beachtende Lizenzfragen und zu erwartende Vor- und Nachteile. In einem praktischen Beispiel wird eine Windows-Anwendung (MFC) unter Anwendung der C++-Bibliothek Qt schrittweise auf den Linux-Desktop portiert.

1 Portierung auf plattform-übergreifende Software

Linux ist schon seit langem eine beliebte Wahl der IT-Experten, wenn es um Serversysteme geht. Das hat dazu geführt, dass Linux aus dem heutigen Servermarkt nicht mehr wegzudenken ist. Wenn es auf Stabilität, Skalierbarkeit und Offenheit ankommt, gibt es keine Alternative.



Für den Linux-Desktop steht diese Entwicklung noch am Anfang. Der Übergang in die Hype-Phase zeichnet sich aber bereits ab, wie die zunehmende Präsenz des Themas „Linux-Desktop“ in den Medien und eine

Vielfalt an aktuellen Studien zum Einsatz in Unternehmen zeigen. Zu erwarten ist ein typischer Verlauf der Hype-Kurve bei gleichzeitigem Anstieg der installierten Systeme wie oben gezeigt. Dass der Umstieg auf den Linux-Desktop für Unternehmen und Verwaltungen schon jetzt einen Return on Invest bringen kann, legen neuere Studien von Gartner [1] oder die Client-Studie von Unilog anlässlich der Linux-Migration der Stadt München [2] nahe. Abhängig ist der Erfolg von einer Reihe von Faktoren, welche ausführlich in den Studien betrachtet werden.

Am einfachsten stellt sich die Migration zum Linux-Desktop für Institutionen dar, die bisher proprietäre UNIX-Systeme oder Mainframe-basierte Software eingesetzt haben. Die Einführung des KDE-Desktops wird von den Mitarbeitern als Schritt hin zu einem modernen Desktop sehr gut angenommen.

Als Beispiel ist hier der in der Planung befindliche Linuxumstieg der Oberfinanzdirektion Niedersachsen mit 11.000 Computer-Arbeitsplätzen genannt. Im Einsatz kommt in den Finanzämtern bisher Solaris x86 mit einem nur rudimentär genutzten CDE. Mitarbeiter forderten zunehmend einen Windows-artigen Desktop. Mit KDE können die IT-Verantwortlichen diese Forderung entgegenkommen und haben gleichzeitig ein optimal administrierbares System zur Verfügung. Die Unterschiede zwischen Solaris und Linux bzgl. der Administration sind im Vergleich zu anderen Betriebssystemen eher gering.

Der Umstieg von Windows auf Linux ist auch immer dann gut möglich, wenn nur eine begrenzte Auswahl von Anwendungen unter Windows eingesetzt wird. Sind diese portabel, kann leicht migriert werden. Problematisch ist in der Regel, wenn Software sehr stark auf Microsoft-Produkte aufsetzt. Software, für die noch keine Linux-Version existiert, wird daher derzeit in der Regel über Terminalserver auf dem Linux-Desktop verfügbar gemacht. Von Behörden wie etwa der Stadtverwaltung Schwäbisch Hall wird dies aber durchaus nur als Übergangslösung betrachtet. Ein Konkurrenz-Produkt unter Linux kann hier durch seine Verfügbarkeit auf beiden Plattformen eine reine Windows-Software verdrängen.

Mit der wachsenden Verfügbarkeit von Desktop-Software unter Linux wird die Migration für immer mehr Institutionen und Firmen möglich. Mit jedem Programm das für den Linux-Desktop zur Verfügung steht, erweitert sich der Kreis der potentiellen Linux-Migrierer. Ist die gleiche Software, die unter Windows genutzt wird, auch unter Linux verfügbar, sinken die Migrationskosten. Die Softwareanbieter, die von Beginn an mit Linux-Versionen dabei sind, werden langfristig hierbei die Nase vorne haben.

Eine plattformunabhängige Software kann von Softwareunternehmen in der Regel nicht von heute auf morgen angeboten werden. Es ist eine langfristige Planung für einen möglichst sanften Umstieg nötig. Um so mehr ist zu empfehlen, dass Softwareunternehmen sich bereits heute mit den Plattformen von morgen auseinandersetzen.

Der Windows-Markt wird deshalb noch lange nicht verschwinden. Und auch der Markt der Embedded-Geräte ist mit seinem derzeitigen Wachstum nicht zu vernachlässigen. Eine plattformunabhängige Software muss daher folgende in Zukunft relevante Plattformen unterstützen:

- KDE/Linux
- Windows
- Mac OS X
- PocketPC und Linux/Embedded

Ein großer Teil der Windows-Software ist derzeit in *Visual C++* realisiert. Bei der Implementierung kamen die *Microsoft Foundation Classes* (MFC) der Windows-Plattform massiv zur Anwendung. Diese sind jedoch nur dort verfügbar, so dass MFC die Softwareanbieter auf der Microsoft-Plattform gefangen hält. Gleiches ist auch bei einem Umstieg auf C# und .NET zu erwarten, insbesondere aufgrund der auf .NET-Technologie registrierten Patente.

Wer sich aus dieser Abhängigkeit befreien möchte, um sich für andere Plattformen zu öffnen, muss sich mit mindestens einem anderen Toolkit vertraut machen. Mit einer plattformunabhängigen Lösung können über Windows und Linux hinaus noch weitere Plattformen bedient werden. Wer würde nicht gerne auf diese Weise 2, 3 oder gar 4 Fliegen mit einer Klappe schlagen?

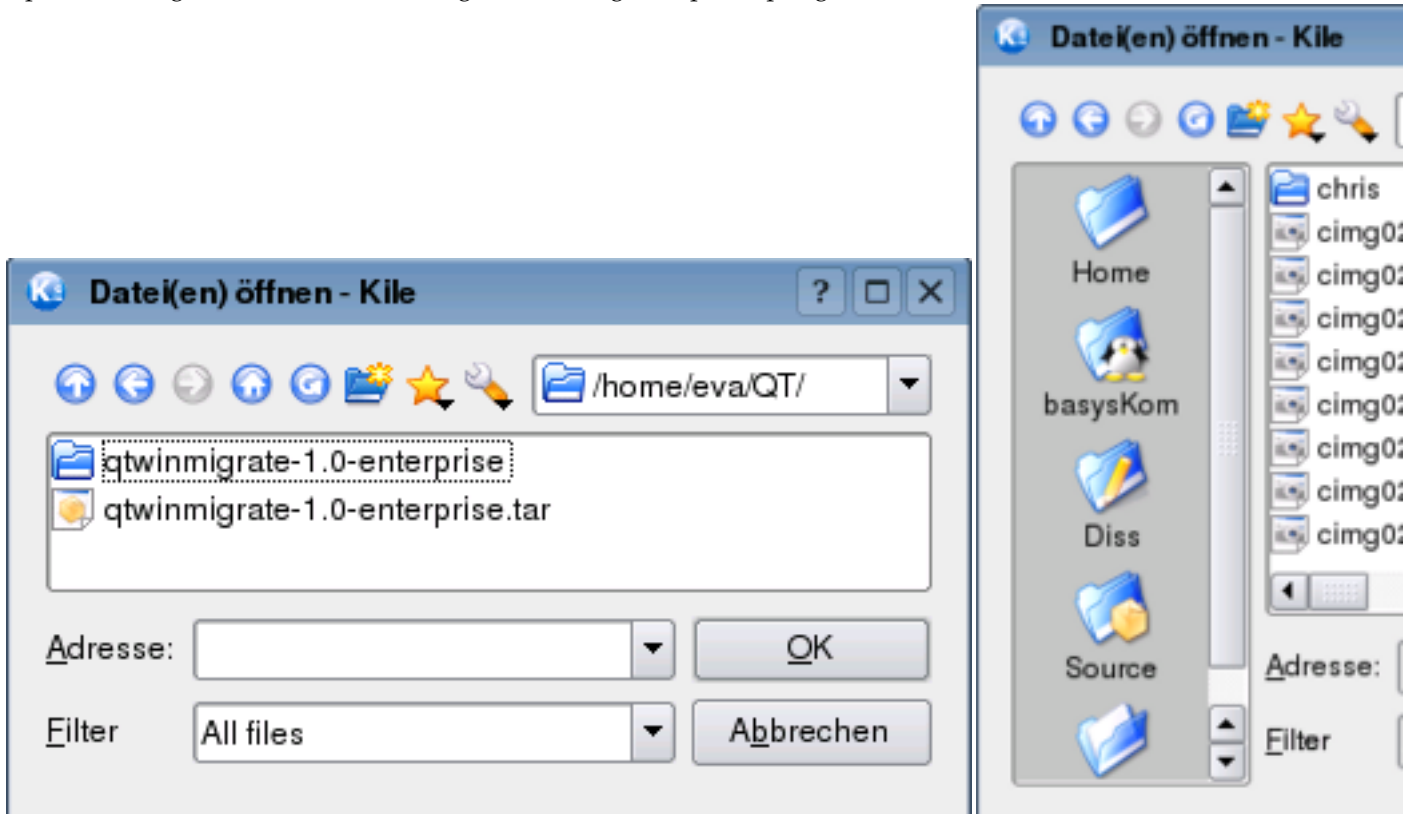
2 Desktop-Plattform der Zukunft: KDE

Das K Desktop Environment (KDE) [3] ist eine moderne Desktopumgebung für UNIX Workstations. Seine größte Verbreitung findet KDE auf Linux-Installationen, weshalb im Folgenden diese Plattform vorgestellt

werden soll.

Bei Start des KDE-Projekts 1996 war die Zielsetzung, einen Desktop für Linux zu schaffen, der einfach anzuwendend ist - ebenso wie es für die Oberflächen von MacOS oder Microsoft Windows der Fall ist. Das Fehlen einer zeitgemäßen Desktopumgebung hat lange Zeit verhindert, daß Linux den Weg auf die Desktops der typischen Computeranwender in Büros und im Heimbereich gefunden hat.

Dieses Ziel hat KDE heute erreicht. Als anwenderfreundlicher Desktop läßt es sich an die Bedürfnisse des Benutzers anpassen, wie der im Screenshot dargestellte Dateidialog in zwei Versionen beispielhaft zeigt. Insbesondere auch für Firmenumfelder ist KDE beispielsweise durch Lockdown-Mechanismen und Helpdesk-Funktionalität in Form von Remote-Zugriff über VNC oder RDP bestens gerüstet. Weitere Vorteile liegen im transparenten Zugriff aufs Netzwerk, integriertem Drag&Drop, Skripting-Schnittstellen und vielem mehr.



Im Dialog zum Öffnen oder Schließen einer Datei läßt sich eine Navigationsleiste und eine Vorschau konfigurieren (links). Die in der Navigationsleiste gezeigten Verzeichnisse können im Firmenumfeld global oder je Anwendung vorgegeben werden. Rechts ist ein einfacher Dialog als Alternative gezeigt. Für den Entwickler ist insbesondere die Verfügbarkeit von hochwertigen Entwicklungstools wie der IDE *KDevelop* oder des Anwendungs-Analysetools *KCacheGrind* von Interesse. Das zugrunde liegende Toolkit *Qt* gibt es zudem auch für andere Plattformen. Doch KDE hat noch mehr zu bieten.

Als Desktop-Umgebung steht eine ganze Infrastruktur zur Einbettung und Anbindung der eigenen Anwendungen an das Desktop-Framework zur Verfügung. Die für den Desktop entwickelten Technologien können ohne Weiteres in eigenen Software-Produkten genutzt werden. Dazu gehören vor allem Standardanwendungen, die als Komponenten umgesetzt wurden und auch in proprietären Anwendungen verwendet werden können. Diese Komponenten sind u.a.

- die Webdarstellung mit *khtml/kjs* (wird beispielsweise von dem Browser *Safari* für Mac OS X verwendet),
- der Editor *Kate*,
- die Email-Komponente *KMail*,
- das Adressbuch *KAddressbook*,
- der Terminkalender *KOrganizer* und

- Office-Elemente (Teil des KDE-Projektes ist das Office-Paket *KOffice* . An einer Integration der OpenOffice.org-Programme in das KDE-Framework wird derzeit entwickelt).

KDE selber verwendet diese Komponenten in eigenen Programmen wie der PIM-Suite *Kontact* (siehe Screenshot). In *Kontact* sind verschiedene Komponenten wie das Email-Programm, Addressbuch und der Terminkalender eingebettet. Der Zugriff auf die Komponenten über Skripting-Schnittstellen ermöglicht zudem einen flexiblen Einsatz, auch wenn sie nicht eingebettet werden.

Eine weitere interessante Technologie zum Einsatz in eigenen Programmen findet sich mit den transparenten Netzwerk-Protokollen. Diese ermöglichen den einfachen Zugriff auf Netzressourcen, ohne dass die Programme die Protokolle selber implementieren müssen. Derzeit stehen über 20 Protokolle zur Verfügung, darunter HTTP, FTP, SMB (für den Zugriff auf Windows- und Samba-Shares), sftp (Verschlüsseltes FTP), LDAP und IMAP.

3 Plattformübergreifend ja - aber wie?

Die eigene Software für alle relevanten Plattformen anbieten zu können, ist für Softwareunternehmen von großem Interesse. Doch wie kann eine solche zukunftssichere Lösung erreicht werden, ohne dass hohe Kosten entstehen und der Aufwand unüberschaubar wird?

Die Zielsetzung einer plattformunabhängigen Software ist je nach Ausgangspunkt nicht ohne zeitlichen und finanziellen Aufwand zu erreichen. Die besten Möglichkeiten sind sicherlich bei einer Neuimplementierung gegeben. Bei dieser kann die Forderung nach Plattformunabhängigkeit von Anfang an in der Softwarekonzipierung berücksichtigt werden.

Bei der Portierung bereits bestehender Softwarelösungen, aber auch bei der Konzeption neuer Software, ist letztlich das Ziel, eine zukunftssichere Lösung zu schaffen bei gleichzeitig optimalem Aufwand. Optimal heißt nicht zwangsläufig minimal, denn eine initial höhere Investition bei der Portierung kann sich später beispielsweise durch niedrigere Wartungskosten auszahlen. Als Kostenfaktoren der Portierung sind vor allem folgende Punkte zu beachten:

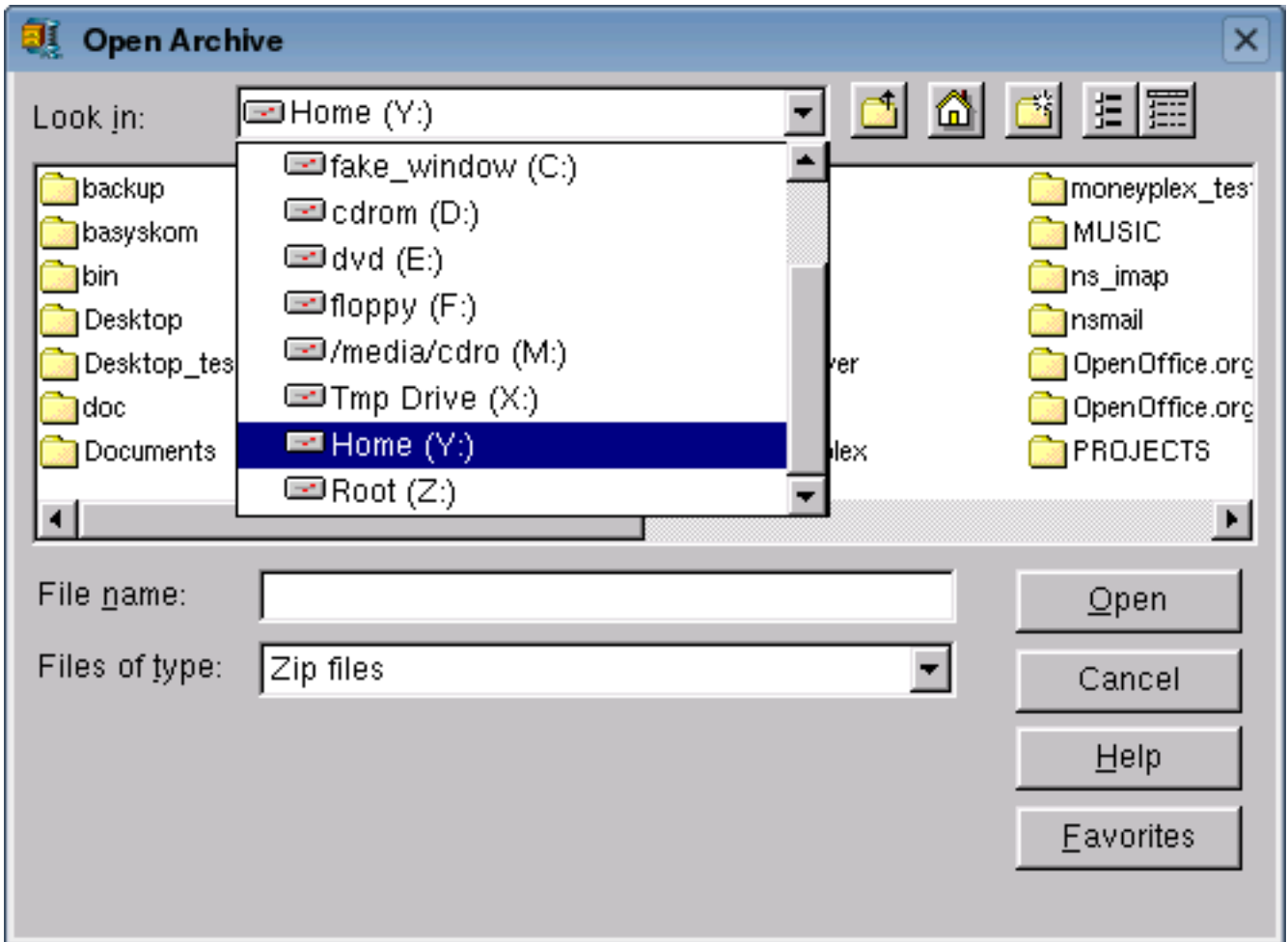
- Lizenzkosten bzw. Lizenzbeschränkungen
- Schulungen der Entwickler
- Personalkosten für die Portierung durch eigene Mitarbeiter
- Beratung und Anpassungen durch externe Dienstleister
- Anpassung der Dokumentation

Um die neue Software möglichst einfach migrieren zu können und gleichzeitig für die nächsten Lebenszyklen der Software zu rüsten, sind zudem folgende Randbedingungen von Bedeutung:

- die Verwendung bereits bestehender Implementierungen ist einer Neuimplementierung vorzuziehen
- vollständige Integration in die Arbeitsumgebung des Anwenders
- Verwendung zeitgemäßer Technologien
- wartbarer Sourcecode

Es gibt verschiedene Wege, um diese Forderungen und Randbedingungen möglichst gut zu erfüllen. Im Folgenden wird ein Überblick über die Möglichkeiten zur Portierung gegeben.

3.1 Direkte Ausführung der Windows-Software



Filedialog eines mit Wine emulierten Programmes (hier WinZip). Die Verzeichnisse werden Linux-Verzeichnissen zugeordnet und der Anwender muss die Verknüpfungen kennen. Eine verführerische Lösung ist es, die Software überhaupt nicht zu portieren und stattdessen auf die verschiedenen Lösungen zu vertrauen, mit denen bereits jetzt Windows-Software unter Linux verfügbar gemacht wird. Zu unterscheiden ist hier zwischen zwei Ansätzen: der Emulation und der Ausführung auf einem Remote Rechner. Zur Emulation kann die Software *Wine* eingesetzt werden, welche auf PC-Hardware die Windows-Plattform nachbildet. Die Windows-Schnittstellen wurden dazu aufgrund fehlernder Dokumentation re-engineert und sind daher nur unvollständig implementiert. Hinzu kommt, dass diese Unterstützung nur für Win32-Programme und auch nur im Windows 95 oder 98 Look&Feel funktioniert - das .NET-Framework wird durch Wine nicht unterstützt und ist aufgrund des großen Umfangs auch nicht geplant.

Produkte wie *Win4Lin* oder das im Firmenumfeld beliebte *VMware* gehen einen etwas anderen Weg, denn sie emulieren einen Rechner im Rechner. Windows (mit einer gültigen Lizenz) muss hier komplett auf dem emulierten Rechner installiert werden. Darauf können dann alle Programme wie üblich ausgeführt werden. Mit dem Betrieb eines Rechners im Rechner geht auch ein entsprechend erhöhter Ressourcenbedarf in Form von Hauptspeicher, Prozessorkapazität und Festplattenplatz einher. Für Entwickler von Software für mehrere Plattformen ist diese Lösung aber optimal, denn sie ermöglicht einen einfachen Wechsel zwischen den Systemen.

Mit der Ausführung auf einem anderen Rechner und dem Remote-Zugriff darauf ist wie bei *VMware* eine komplette Windows-Umgebung gegeben. Die zwei verfügbaren Lösungen finden sich in Form des *Windows Terminal Servers* für den Windows Server 2003 und des nicht ganz preisgünstigen *Citrix*.

Problematisch ist (wie auch bei *Vmware*) der Austausch der Daten. Letztlich handelt es sich um die Benutzung zweier unterschiedlicher Rechner mit unterschiedlichen Dateisystemen. Der gemeinsame Zugriff auf

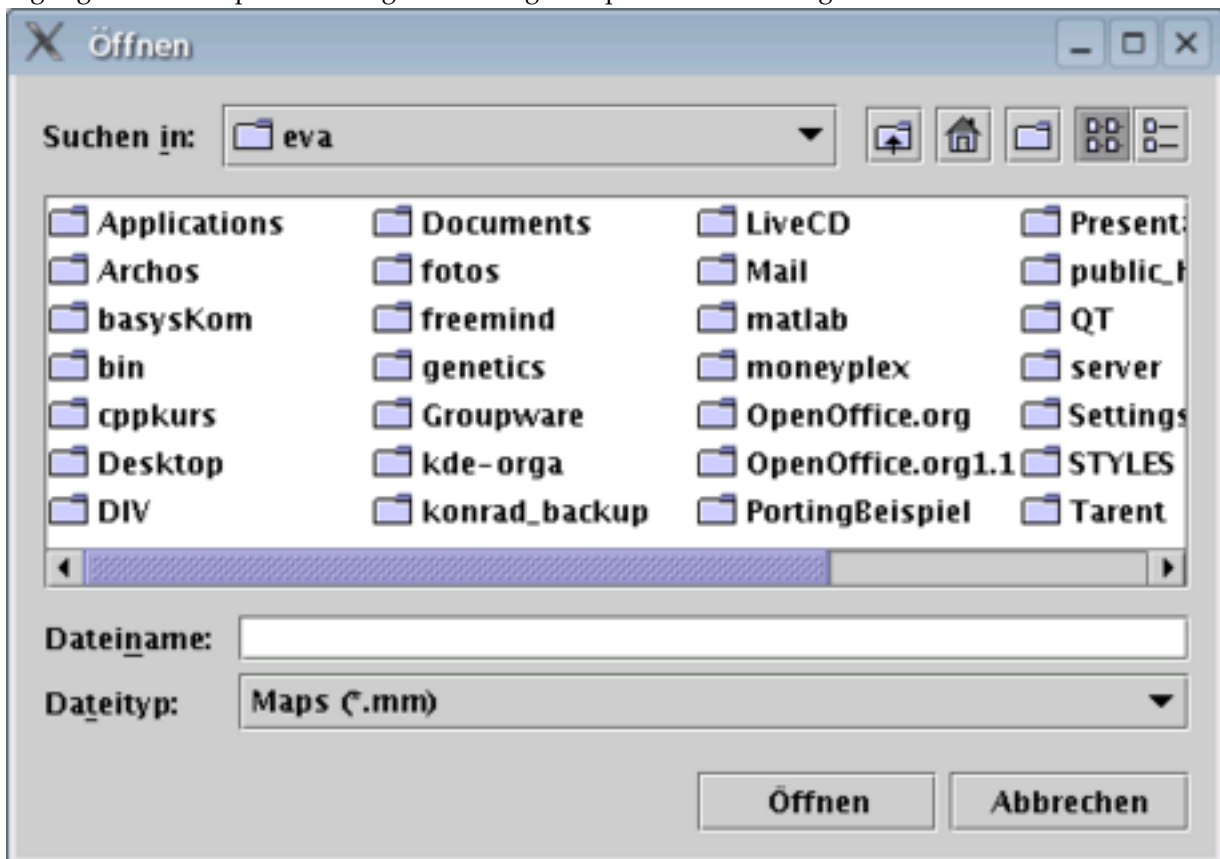
Daten kann über einen Fileserver erfolgen. Neben *Wine* ermöglicht es *Citrix* zudem als einzige Lösung, nur eine einzige Applikation auf dem Anwenderdesktop anzuzeigen. *VMware*, *Win4Lin* und der *Windows Terminal Server* zeigen immer einen kompletten Windows-Desktop, auf dem dann wiederum die Zielanwendung ausgeführt wird. Desktop-Integration in Form einer gemeinsamen Zwischenablage und Drag&Drop sind aber bei allen diesen Systemen problematisch.

Die hier vorgestellten Lösungen werden sich sicherlich weiterentwickeln und ihre Möglichkeiten verbessern. Will man aber wirklich mehrere Plattformen unterstützen, ohne Kunden zusätzliche Kosten und dem Anwender Unannehmlichkeiten in der Bedienung aufzubürden, bleibt nur die Möglichkeit einer plattformunabhängigen Softwareimplementierung.

4 Überblick über plattformunabhängige Softwarelösungen

In den letzten Jahren wurden für die Erstellung von plattformunabhängigen Programmen vor allem Web- und Java-Anwendungen propagiert. Doch ist dies für Desktop-Anwendungen wirklich die richtige Wahl?

Webanwendungen integrieren sich kaum in den Anwender-Desktop. Nur wenige GUI-Elemente stehen zur Verfügung und Desktop-Anbindungen wie Drag&Drop können nur mit großem Aufwand realisiert werden.



Zeich-

nung Filedialog eines Java-Programms Auch Java-Anwendungen integrieren sich aufgrund ihres Look&Feels und häufig schlechter Performance nur wenig in den Desktop. Dem Anwender stellen sich die Programme oft als Fremdkörper dar.

Jeder Desktop hat ein natives Toolkit. Das ist jenes, in dem der Desktop selber programmiert ist. Natürlicherweise integrieren sich solche Anwendungen am besten, die das selbe Toolkit verwenden. Dies schließt aber andere Toolkits nicht unbedingt aus, denn diese können als Zwischenlayer zum Desktop-Toolkit dienen. Man spricht dann von *Bindings*. Toolkits mit Bindings zu den nativen Toolkits mehrerer Plattformen ermöglichen es, plattformunabhängige Software zu schreiben. Der gleiche Source-Code kann auf mehreren Plattformen kompiliert und eingesetzt werden.

Betrachtet man die Toolkits der Standard-Desktops, so fällt auf, dass die natürliche Programmiersprache der Desktops C++ ist. Dies gilt für Windows mit MFC genauso, wie für MacOS mit Quartz und auch für KDE

mit Qt. Die Ausnahme bilden CDE mit Motif und GNOME mit Gtk, die beide in C realisiert sind. Als objektorientierte Programmiersprache verbindet C++ die Forderung nach einer Programmiersprache, welche die Methoden des Softwareengineerings unterstützt, mit der optimalen Anbindung an die verschiedenen Desktop-Frameworks. Für Windows ist das native Toolkit derzeit noch MFC und wird gerade von .NET abgelöst.

Für Programme ohne GUI liegt schon seit längerem die Lösung für Plattformneutralität darin, möglichst standardkonform zu programmieren. Für C++ bietet sich die Verwendung der *Standard Template Library* (STL) als Teil des ANSI C++-Standards an. Als Erweiterung kommt die Bibliothek *boost* [4] in Frage.

Um für GUI-Anwendungen plattformunabhängige Anwendungen zu programmieren, benötigt man ein Toolkit, welches nach Möglichkeit eine gute Schnittstelle bietet, um schnell und einfach die eigene Anwendung realisieren zu können. Die Programmiersprache des Toolkits muss auf allen Zielsystemen verfügbar sein. Damit eine gute Einbettung in die jeweiligen Desktops gegeben ist, ist zudem eine Anbindung an den nativen Desktop nötig.

Das Toolkit Qt stellt hierbei einen Sonderfall dar. Als natives Toolkit des KDE-Desktops bietet es gleichzeitig unter Windows und MacOSX schlanke und performante Layer zu deren nativen Toolkits. Zusammen mit einer hohen Qualität ist Qt die optimale Lösung, um mehrere Plattformen zu unterstützen.

Dies gilt insbesondere für MFC-Programme. Zum einen bleibt mit C++ die Programmiersprache gleich und alte Codeteile können unter Umständen direkt weiter verwendet werden. Zum anderen verfügt Qt über ein Migrations-Framework, das es ermöglicht, eine MFC-Anwendung schrittweise zu migrieren.

Tabelle 1:

Ausgangssituation	Portierungsansatz
Web: PHP, Perl, JavaScript, Java	Ist plattformunabhängig; es wird das Toolkit des Browsers verwendet.
Skriptsprachen: Python, Perl, Tcl/Tk	Verwendung von plattformunabhängigen Bindings, z.B. für Python.
Visual Basic, VBA	Neuimplementierung, z.B. in Java oder C++.
C/C++ ohne GUI	Portierung auf standardkonforme Toolkits (STL, boost) oder auf das native Toolkit.
C/C++ mit MFC	Sanfte Migration auf Qt, wxWindows.
C#	Mit Mono soll eine plattformunabhängige Lösung erreicht werden. Es gibt aber noch keine stabile Version.
Delphi	Derzeit noch plattformunabhängig (mit Kylix). Da das Produkt von Borland ausläuft, ist die Migration auf Qt in Planung.
Java	Ist bereits plattformunabhängig, evtl. Portierung auf Toolkit.

Portable Toolkits für Windows

Etwas anders stellt sich die Situation dar, wenn nicht MFC der Ausgangspunkt ist, sondern eine andere Programmiersprache. Tabelle 1 listet einige typische Ausgangspunkte und mögliche Lösungen dafür.

Besondere Berücksichtigung bei allen Lösungen für plattformunabhängigen Code benötigt eine mögliche OLE- oder COM-Anbindung an Microsoft-Programme.

5 Beispiele verfügbarer Software

Den Schritt hin zu einer plattformneutralen Codebasis sind bereits einige Firmen gegangen. Die folgende Tabelle listet einige freie Projekte und eine ganze Reihe von Firmen mit ihren Produkten, welche für mehrere Plattformen verfügbar sind.

Table Produkte für mehrere Plattformen

5.1 Von MFC zu Qt

Das plattformunabhängige Toolkit Qt der norwegischen Firma Trolltech [5] richtet sich an Entwickler, welche nur einen einzelnen Source-Tree für mehrere Plattformen erstellen und warten möchten. Soll eine neue Plattform unterstützt werden, so muss mit der entsprechenden Qt-Version nur neu kompiliert werden. Die in C++ geschriebene Klassenbibliothek ist besonders durch ihre Verwendung im KDE-Projekt populär geworden.

Qt besticht vor allem durch seine hohe Qualität und die gute Dokumentation. Da die Bibliothek im Sourcecode verfügbar ist, kann mit dem Toolkit flexibel gearbeitet werden. Es stehen Methoden für grundlegen-

Tabelle 2:

Software	Typ	Toolkit	Haupt-Plattformen
OpenOffice (frei)	Office	eigenes Toolkit, Java	Windows, UNIX, Linux, MacOSX (mit mit X)
Gimp (frei)	Grafik	GTK	Windows, UNIX, Linux, MacOSX (nur mit X)
Scribus (frei)	DTP	Qt	Linux
Matlab (Mathworks)	Mathematik/ Simulation	Kern: C, GUI: Java	Windows, UNIX, Linux, MacOSX
Eagle (CadSoft)	Platinenlayout	Qt	Windows, Linux
Parity (ParitySoftware)	ERP	Qt	Windows, Linux
Kylix (Borland)	IDE	Delphi	Linux
Tax 2004 (Buhl)	Steuerprogramm	Emulation mit Wine	Windows, Linux
Map&Route (Telekom)	Telefonbuch	GTK	Windows, Linux
Opera (Opera)	Webbrowser	Qt	Windows, Linux, MacOSX, Linux/Embe
Moneyplex (Matrica)	Homebanking	Qt	Windows, Linux, OS/2
Brockhaus/Duden	Lexikon	Qt	Windows, Linux, MacOSX

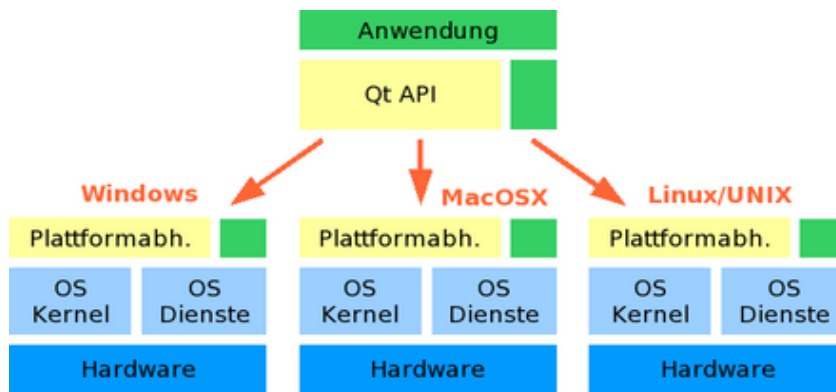
de Datenverarbeitung und eine breite Palette an GUI-Elementen zur Verfügung. Statt Callbacks zur Verbindung der GUI-Elemente mit den dahinterliegenden Funktionen setzt Qt auf einen innovativen Signal-Slot-Mechanismus. Die GUI wird mit einem Designer-Programm inklusive eines flexiblen Layouts erstellt. Ein starres Design mit fest vorgegebenen Größen der GUI-Elemente wird so ohne zusätzlichen Programmieraufwand vermieden. Über diese Basisfunktionalität hinaus kann auf weitere Module zurückgegriffen werden. Der Programmierer kann Qt-Klassen für Netzwerk-Funktionalität, Datenbank-Anbindungen, XML-Verarbeitung und vieles mehr einsetzen.

Qt steht in der frei verfügbaren Version unter der Freien Lizenz *GNU General Public License (GPL)*. Trotzdem können mit Qt auch proprietäre und kommerzielle Programme entwickelt werden, denn Qt steht unter einer Dual-Lizenz zur Verfügung:

- für GPL-Programme wird Qt unter der GPL-Lizenz kostenlos (zumindest auf Linux, Unix und Mac OS X) genutzt
- für proprietäre Software kann Qt pro Entwickler und Plattform lizenziert werden. Ein entsprechender Produkt-Support kann dann in Anspruch genommen werden.

Die proprietäre Lizenz stellt neben der Möglichkeit, Support in Anspruch zu nehmen, vor allem sicher, dass Qt aktiv gewartet und weiterentwickelt wird. Dies ist die Voraussetzung dafür, dass Qt als langfristige Grundlage eines Softwareprodukts in Frage kommt. Zudem gibt es einen Vertrag zwischen dem KDE e.V. und Trolltech. Dieser beinhaltet, dass Qt unter einer freien Lizenz weiterentwickelt werden kann, sollte Trolltech das Produkt einstellen oder verkaufen.

Qt steht derzeit für Unix/X11 und auch als Qt/Win, Qt/Mac und Qt/Embedded zur Verfügung. Ein Qt/WinCE-Version wird derzeit von Trolltech nicht ausgeschlossen. Die Windows-Version von Qt verwendet das native Windows-Toolkit. Verschiedene Style-Engines inklusive der XP-Engine können verwendet werden, so dass sich dem Benutzer das vertraute Look&Feel von XP bietet. Anstelle der eigenen Qt-Dialoge für Dateien, Drucken, etc. verwendet Qt/Win die Dialoge des aktuellen Systems. Ähnlich greift die Qt/Mac-Version auf Quartz zurück, um GUI-Elemente und Dialoge darzustellen. Unter UNIX und Linux verfügt Qt über eine eigene Style-Engine, welche die verschiedensten Styles verwenden kann, wie die Konfigurationsmöglichkeiten des KDE-Desktops zeigen.



Das plattformunabhängige Toolkit Qt bietet neben der breiten Palette an GUI-Elementen und Werkzeugen auch ein Migrationsframework speziell für MFC-Anwendungen. Eine Portierung auf Qt bedeutet jedoch nicht, dass man nun komplett auf die Microsoft-Welt verzichten müsste. Qt verfügt über Anbindungen zu .NET und auch ActiveQt als Schnittstelle zu ActiveX steht zur Verfügung. Diese Elemente sind jedoch nicht portabel und bilden einen systemspezifischen Teil.

Ist ein Programm über proprietäre Protokolle und Schnittstellen an Microsoft-Produkte angebunden, so können diese nicht portiert werden. Dies ist beispielsweise für OLE und COM der Fall. Für diese muss unter Linux ein passender Ersatz gefunden werden - identisch wird dieser aber nie sein. Dies erfordert in der Regel auch im Sourcecode der zu portierenden Programme einen plattformabhängigen Teil.

Die besonderen Vorteile von Qt für den Einsatz als Basis der eigenen Anwendung liegen zusammengefasst in folgenden Punkten:

- Erweiterte Märkte durch Plattformunabhängigkeit
- Flache Lernkurven durch einfache APIs
- Niedriger Wartungsaufwand des eigenen Sourcecodes
- Migrations-Frameworks für MFC und Motif

5.2 Ablauf einer Portierung

Die Portierung einer Anwendung von MFC nach Qt mit dem MigrationsFramework von Trolltech wird schrittweise durchgeführt. GUI-Elemente (Controls) und Dialoge können parallel in MFC und Qt implementiert sein und funktionieren im gleichen Programm. Zur Portierung werden zuerst die Dialoge auf die neue Bibliothek umgestellt. Der Qt Designer, ein im Leistungsumfang von Qt enthaltener Interface-Designer, hilft dabei. Er kann vorhandene Ressource-Dateien importieren und somit automatisiert in Dialog-Vorlagen für Qt umwandeln. Sind die Dialoge portiert, können einzelne GUI-Elemente Schritt für Schritt umgesetzt werden. Die dazugehörigen Algorithmen und Funktionen ohne GUI werden gleichzeitig auf das neue Toolkit portiert. Gegebenenfalls kann dieser Schritt auch in Form einer Refaktorisierung vor der GUI-Portierung durchgeführt werden. Dann ist der Zeitpunkt gekommen, den eigentlichen Kern der Anwendung mit seinen Menüs, Toolbars und Statusleisten in eine Qt-Anwendung zu überführen. Es können aber immer noch MFC-Controls und Dialoge ausgeführt werden. Sind auch diese migriert, ist die Anwendung bereit, um auf anderen Plattformen übersetzt zu werden.

Plattformabhängige Funktionselemente wie COM-Schnittstellen und OLE-Anbindung werden sinnvollerweise in eigene Module gekapselt. Hier kann beispielsweise ein Plugin-Konzept eingesetzt werden. Weitere Plugins für KDE und andere Plattformen realisieren dann die Kommunikation mit den jeweiligen Desktop-Komponenten. So kann jede Plattform optimal angebunden werden.

5.3 Zusammenfassung und Bewertung

Ein wachsender Linux-Markt und der weiterhin bestehende Windows-Markt machen plattformunabhängigen Code notwendig. Die Portierung ist eine Investition in die Zukunft.

Da viele Desktop-Anwendungen mit MFC programmiert wurden und C++ die native Programmiersprache der Desktops ist, bietet sich das plattformunabhängige Toolkit Qt als strategische Plattform der Zukunft an. Der Aufwand für die Portierung wird durch ein Migrations-Framework gemildert. Eine schrittweise Portierung bei durchgehend funktionstüchtigem Code ist damit möglich.

5.4 Über die Autorin

Dip.-Ing. Eva Brucherseifer ist Geschäftsführerin der basysKom GbR, die Beratung und Auftragsentwicklungen für plattformunabhängige Business-Anwendungen sowie Dienstleistungen für den Linux-Desktop KDE anbietet. Nach ihrem Studium der Automatisierungstechnik an der Technischen Universität Darmstadt war sie als wissenschaftliche Mitarbeiterin mit dem Schwerpunktthema Evolutionäre Algorithmen beschäftigt. Seit 2000 trägt sie aktiv zum KDE-Projekt bei und ist seit August 2002 stellvertretende Vorsitzende des KDE e.V.

5.5 Literatur

- [1] http://www4.gartner.com/DisplayDocument?id=406459&ref=g_search <http://www4.gartner.com/DisplayDocument?id=406459&ref=g_search>
- [2] http://www.muenchen.de/Rathaus/lhm_alt/mde/aktuell/39405/ms_linux.html <http://www.muenchen.de/Rathaus/lhm_alt/mde/aktuell/39405/ms_linux.html>
- [3] <http://www.kde.org> <<http://www.kde.org/>> K Desktop Environment
- [4] <http://www.boost.org> <<http://www.boost.org/>> Boost Bibliothek
- [5] <http://www.trolltech.com> <<http://www.trolltech.com/>> Qt Bibliothek