KDE-Applikationen von der Kommandozeile aus bedienen

K-Steuerung

KDE-Programme mit Maus und Tastatur bedienen ist in vielen Fällen sehr zeitraubend. Hier hilft DCOP, das Desktop Communication Protocol. Es steuert GUI-Programme per Kommandozeile oder Skript. Scott Wheeler

DCOP ist KDEs Skripting-Backend, über das sämtliche KDE-Applikationen miteinander kommunizieren. Es veranlasst Anwendungen dazu, ihre Daten zu aktualisieren, wenn der Anwender die Einstellungen ändert, oder startet KMail nach einem Klick auf eine E-Mail-Adresse. Doch DCOP arbeitet nicht nur im Hintergrund. Es erlaubt Anwendern ihre Arbeit mit KDE zu automatisieren, auch von einem anderen Rechner aus.

Listing 1: DCOP-Bindings in Perl

```
01 #!/usr/bin/perl
02
03 use DCOP;
04
05 while($line = <>) {
06   $contents .= $line;
07 }
08
09 $client = new DCOP();
10 $client->attach();
11
12 $klipper = $client->createObject("klipper", "klipper");
13 $klipper->setClipboardContents($contents);
```

Ein Benutzer darf allerdings nur die Programme steuern, die mit seiner eigenen User-ID laufen.

Allen KDE-Anwendungen gemein ist ein grundlegendes DCOP-Interface, das es ermöglicht, Fenster in ihrer Größe zu verändern oder sie mit Eigenschaften wie »immer im Vordergrund« zu versehen. Die meisten Programme bieten darüber hinaus eigene Methoden; KMail besitzt Funktionen, um E-Mails abzufragen oder eine neue Nachricht zu erstellen, Konqueror öffnet per DCOP ein Fenster mit einer bestimmten URL und KWord erlaubt es, die Eigenschaften von Dokumenten zu ändern.

Interfaces mit Kdcop erkunden

Weil DCOP so viele Funktionen besitzt, wirkt es auf den ersten Blick etwas unübersichtlich. Das grafische Werkzeug »kdcop« verschafft Abhilfe: Es ordnet die einzelnen Applikationen in einer Hierarchie an. Die erste Ebene zeigt alle KDE-Programme, die im Moment laufen. In der zweiten Ebene liegen die vorhandenen Interfaces einer Applikation. Jedes Interface bietet in einer weiteren Ebene eine Liste von Methoden an.

Einige Applikationen erscheinen lediglich einmal in der Interface-Liste, das sind die so genannten Unique Applications wie KMail oder KWin, von denen immer nur eine Instanz läuft. Andere Programme finden sich möglicherweise mehrmals in der Liste, sie lassen sich aber anhand der Prozess-ID unterscheiden, die Kdcop anzeigt.

Nach dem Start einer Konqueror-Instanz erscheint in der Kdcop-Hierarchie der Eintrag »konqueror-*PID*«. Er bietet neben einer Reihe von Interfaces auch »KonquerorIface (Voreinstellung)« an. Dieses Interface enthält unter anderem die Methode »createNewWindow(QString url)«. Ein Doppelklick darauf öffnet ein Fenster (siehe Abbildung 1), in dem der Bediener eine URL angibt. Nach dem Bestätigen mit »OK« sendet Kdcop eine DCOP-Nachricht an den Konqueror, der die URL in einem neuen Fenster lädt.

KDE weckt mit Musik

Die Methoden, die Kdcop anzeigt, sind überall einsetzbar: Im Folgenden soll der Multimediaplayer Juk jeden Morgen zu einer bestimmten Zeit Musik abspielen. Juk selbst hat keine solche Alarmfunktion, dafür ist die zweite Applikation KAlarm zuständig. DCOP ist dazu prädestiniert, diese beiden Programme zusammenarbeiten zu lassen.

Ein Klick auf »Neu« in KAlarm öffnet einen Dialog, dort wählt der Anwender als Aktion »Befehl« aus und zieht per Drag & Drop die Methode »juk | Player | play()« aus Kdcop ins Eingabefeld. Dort erscheint dann »dcop juk Player play« (siehe Abbildung 2). Nach dem Einstellen der gewünschten Uhrzeit ist der Wecker fertig eingerichtet.

Es bietet sich mit dieser Methode ebenfalls an, Programme per Shellskript zu steuern. Das Tool, um DCOP von der Kommandozeile zu kontrollieren, heißt »dcop«. Dieses Programm stellt die

Der Autor

Scott Wheeler ist seit mehreren Jahren aktiver KDE-Entwickler und Autor von Flashkard, KSig und Juk. Er arbeitet in Heidelberg als Software-Entwickler.

Baumstruktur, die Kdcop anzeigt, auf der Kommandozeile dar. Ein einfacher Aufruf gibt eine Liste aller laufenden KDE-Applikationen aus.

Mit einem Anwendungsnamen als Argument zeigt »dcop« die zur Verfügung stehenden Interfaces an. Wenn das Tool zusätzlich als Argument noch den Namen eines Interface erhält, gibt es eine Liste an Methoden dieses Interface aus. Der folgende Aufruf veranlasst KDE beispielsweise dazu, zum zweiten virtuellen Desktop zu wechseln:

```
dcop kwin KWinInterface setCurrentDesktop 2
```

Skript-Programmierung mit DCOP-Bindings

Doch DCOP ist nicht nur über die Kommandozeile steuerbar, das Paket »kdebindings« enthält DCOP-Bindings für eine Vielzahl von Programmiersprachen, unter anderem für Perl, Python, C, Objective-C und Java, das C++-Binding ist bereits im »kdebase«-Paket enthalten. Das kleine Perl-Skript aus Listing 1 kopiert die Eingabe von Stdin über eine DCOP-Methode in die Zwischenablage von Klipper.

Zeile 3 des Listings importiert das DCOP-Perl-Modul. Der Code in den Zeilen 5 bis 7 liest Stdin (oder eine Datei, die als Parameter übergeben wird) zeilenweise in die Variable »$contents« ein. Danach erstellen die Zeilen 9 und 10 einen neuen DCOP-Client und hängen ihn an den laufenden Server an. Diesen Code muss jedes Perl-Skript enthalten, das die DCOP-Bindings benutzt.

Der nächste Aufruf erstellt das DCOP-Interface zu KDEs Zwischenablage Klipper. Die Methode »createObject()« erwartet zwei Argumente: Den Namen der zu kontrollierenden Applikation sowie den Namen des Interface, das die entsprechenden Methoden enthält. Das Skript kann beliebig viele DCOP-Objekte für verschiedene Aufgaben erzeugen. Die letzte Zeile füllt schließlich die Zwischenablage mit dem Inhalt aus »$contents« (Abbildung 3).

Perl kennt dank seiner dynamischen Natur die Methoden, die im Objekt vorhanden sind. Es besteht ein direktes Mapping der »setClipboardContents()«-Methode in DCOP auf eine Perl-Methode. Die Perl-Bindings haben eine wichtige Eigenart: DCOP-Methoden geben immer einen Array zurück, auch wenn dieser lediglich einen Wert enthält und so als Variable zurückgegeben werden könnte. Folgender Code speichert den ersten Eintrag der Zwischenablage in einem Array:

```
@history = $klipper->getClipboardHistory
Item( 0 );
print "$history[0]\n";
```

Wer sich näher über die Perl-Bindings informieren will, sei auf die Manpage »DCOP« verwiesen. Ein DCOP-Tutorial für C++-Programmierer gibt es unter [1] und eine genaue Beschreibung des Protokolls und der zugrunde liegenden Architektur findet sich auf [2]. *(mwe)* ■

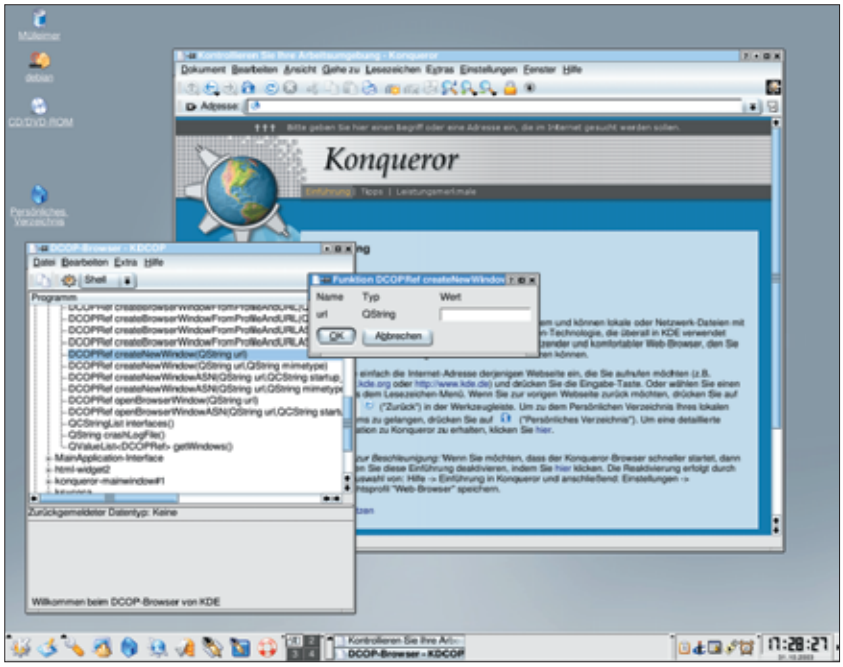

Abbildung 1: Kdcop (links) ist eine grafische Oberfläche für DCOP-Operationen. Der Benutzer hat in diesem Beispiel die Methode »createNewWindow()« (linkes Fenster) durch Doppelklick ausgewählt. Im daraufhin erscheinenden Fenster (Vordergrund Mitte) gibt er die URL ein, die Konqueror in einem neuen Fenster öffnet.

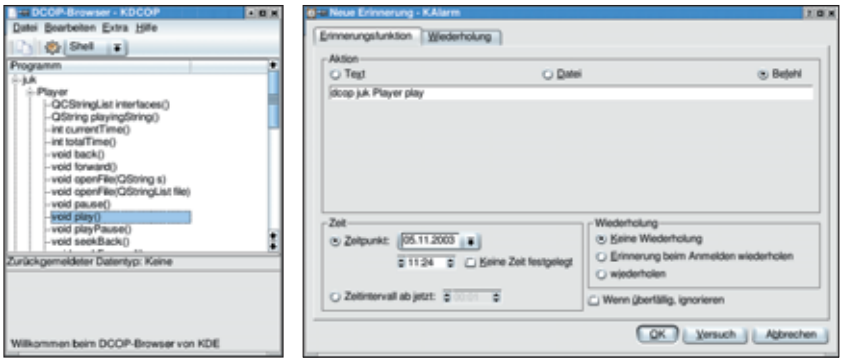

Abbildung 2: Kdcop unterstützt Drag & Drop. Die Methode »play()« wurde einfach vom Kdcop-Fenster (links) in die Eingabezeile von KAlarm gezogen (rechts). Das Tool »dcop« ruft die Methode auf.

Infos

[1] DCOP-C++-Tutorial: [http://developer.kde.org/documentation/tutorials/dot/dcopiface/dcop-interface.html]

[2] Protokoll-Dokumentation: [http://developer.kde.org/documentation/library/kdeqt/dcop.html]

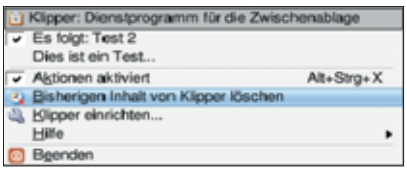

Abbildung 3: Mit den Perl-Bindings für DCOP hat ein simples Perl-Skript (siehe Listing 1) die Zwischenablage von Klipper mit Elementen gefüllt.