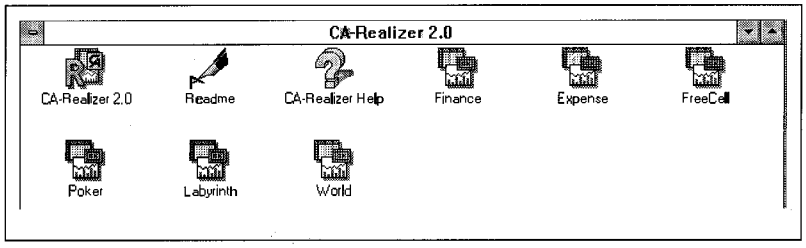
Programmieren, ohne zu programmieren? Arbeiten mit dem CA-Realizer

Michael Haas



Wird das Basic-Entwicklungssystem CA-Realizer dem selbstgesetzten Anspruch gerecht, Programmentwicklung zu ermöglichen, ohne daß man eine Zeile Programmcode schreiben muß?

Visuelles Programmieren

Seit der zunehmenden Verbreitung grafischer Benutzeroberflächen wie Windows etc. erscheinen immer mehr Systeme zur sog. visuellen Programmentwicklung auf dem Markt. Dahinter steht die ebenso einfache wie richtige Beobachtung, daß bei der Programmentwicklung die meiste Energie auf die Gestaltung einer bedienerfreundlichen Programmoberfläche verwendet wird. Idee der visuellen Umgebungen ist es, zunächst die Programmoberfläche auf einer Art Zeichenbrett zu gestalten und erst dann die Funktionalität hinzuzugeben. Dies sieht dann z.B. so aus, daß der Programmcode, der nach dem Druck auf einen Knopf ausgeführt werden soll, in einem Fenster eingegeben wird, das dem Knopf zugeordnet ist.

Das Programm *CA-Realizer* geht noch einen Schritt weiter, indem es dem Entwickler eine große Anzahl von "Programmable Application Tools" zur Verfügung stellt. Dabei handelt es sich z.B. um einen kompletten Texteditor oder Funktionen zur Erstellung von Diagrammen. Daher findet man auf der Verpakkung des Realizer die Behauptung: *"Generate applications without writing a single line of code."*

Wird mit dem Realizer das Programmieren wirklich zum Kinderspiel? Welche Möglichkeiten bieten sich dem Juristen mit dem Realizer? Diese Fragen sollen im Mittelpunkt der folgenden Besprechung stehen. Dabei haben die grundsätzlichen Erwägungen zu visuellem Programmieren durchaus über den Realizer hinaus Geltung für andere Umgebungen.

Beispielprogramm

Um die Besprechung realitätsnah zu gestalten, wird anhand des Programms *MdB-Brief*, das auf der Diskettenbeilage zu finden ist, der typische Gang einer Programmentwicklung mit dem Realizer dargestellt.

Dieses Programm ermöglicht es, Briefe an einen bestimmten Bundestagsabgeordneten oder eine Gruppe von Bundestagsabgeordneten zu schreiben. Wer das Programm schon jetzt betrachten möchte, braucht nur die Diskette einzulegen und (unter Windows) das darauf befindliche Programm *Install* zu starten. Ein kleiner Vorgriff: Der Realizer erstellte dieses Programm ebenso automatisch wie die ganze Diskette.

Das Realizer-Konzept

Bei CA-Realizer handelt es sich um einen Basic-Compiler, der keinen reinen Maschinencode erzeugt, wie das z.B. C-Compiler tun. Aus diesem Grund wird zum Starten einer Realizer-Anwendung ein Laufzeitsystem benötigt. Dieses darf mit den Anwendungen frei verteilt werden.

Nach dem Start des Realizer findet man sich in einer gewohnten Programmierumgebung, die auf der direkten Eingabe von Codezeilen beruht *(vgl. Abb. 1)*.

Die visuelle Entwicklungsumgebung verbirgt sich hinter dem Hilfsprogramm *Form Developer*, kurz *FormDev*. In FormDev kann man die Benutzeroberfläche entwerfen und den einzelnen Objekten Programmcode zuordnen. Ist man damit fertig, erzeugt

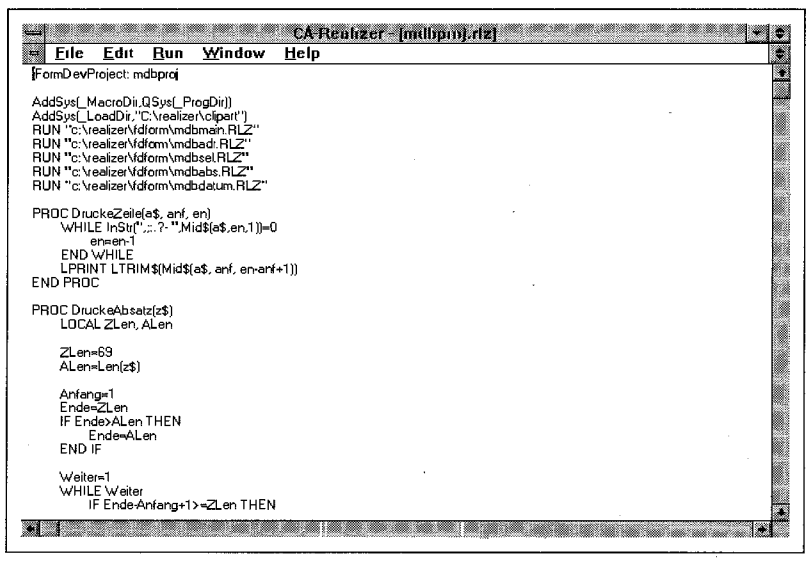
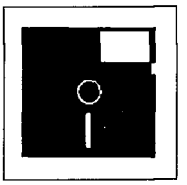
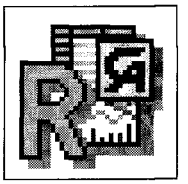


Abb. 1: Realizer-Quellcode

Michael Haas ist Mitarbeiter am Lehrstuhl für Rechtsinformatik der Universität des Saarlandes.





FormDev den übersetzungsfähigen Code für den Realizer.
Bei Realizer ist man also nicht auf die visuelle Programmierung angewiesen. Man kann auch codeorientiert arbeiten, wie z.B. unter MS-QuickBasic. Das kann Vorteile bei der Portierung vorhandener Programme bringen.

XBase-Unterstützung

Grundlage des Beispielprogramms soll eine XBase Datenbank bilden, die die Daten der Abgeordneten enthält. Zunächst soll es möglich sein, die einzelnen Datensätze anzuschauen, zu suchen und zu ändern.
Der Schlüsselbegriff bei der Programmierung mit dem Realizer ist der des Formulars. Dabei handelt es sich um ein Fenster, das Objekte zur Interaktion (Knöpfe, Textboxen u.s.w.) enthält, und damit die Schnittstelle zum Benutzer darstellt. Formulare werden mit FormDev visuell entworfen *(Abb. 2)*.
Da das XBase-Format besonders unterstützt wird, ist es möglich, ein entsprechendes Formular weitgehend automatisch erstellen zu lassen. Startet man das erstellte Formular, so wird man enttäuscht. Zwar findet man alle Felder der Datenbank am Bildschirm, und auch Eingaben sind möglich, aber der Inhalt der Datenbank wird nicht angezeigt und Blättern o.ä. ist ebenfalls nicht möglich. Diese Funktionen müssen von Hand hinzugefügt werden. Das heißt im Klartext, es muß programmiert werden.
Schon hier zeigt sich also, daß das Versprechen nicht gehalten wird, man könne Programme erstellen, ohne eine Zeile Code zu schreiben. Daß dies grundsätzlich machbar ist, zeigen moderne Datenbanken, die entsprechende Masken automatisch erstellen.
Mit einem geringen Grundverständnis von Programmierung und nach einem kurzen Blick in die umfangreichen Handbücher, stellt es kein Problem dar, einzelne Datensätze aus der XBase-Datei zu lesen und in der Maske des Formulars auszugeben.
Allerdings wird man vergeblich nach einer Möglichkeit suchen, einen bestimmten Datensatz zum Beispiel über den Namen des Abgeordneten zu finden. Das liegt daran, daß der Realizer zwar XBase-Datendateien unterstützt, nicht jedoch die zur schnellen Suche notwendigen Indexdateien. Um dennoch eine Suchmöglichkeit anbieten zu können, ist Programmierwissen erforderlich, das über die Grundlagen hinaus Kenntnis besonderer Algorithmen umfaßt.
Mit diesem Wissen wäre es auch möglich, die Änderung einzelner Datensätze zu erlauben. Mit der Exportfunktion des Realizer ist es nur möglich, eine komplette XBase-Datei zu speichern. Bei einer Datenbank, die wie die hier verwendete über 600 Datensätze enthält, ist eine solche Bearbeitung im Arbeitsspeicher kaum sinnvoll möglich.
Das Entwerfen der Such- und Änderungsfunktionen käme dem Entwurf einer kompletten XBase-Schnittstelle gleich, was mit jeder Programmiersprache möglich ist. Da XBase-orientierte Datenbanken gerade im juristischen Bereich eine besondere Bedeutung zukommt, stellt die sehr eingeschränkte XBase-Unterstützung des Realizer für diesen Anwenderkreis ein erhebliches Manko dar.
Für das Programm MdB-Brief bedeutet dies, daß Änderungen an der Datenbank nicht möglich sind. Die Suche ist nur nach dem Namen eines Abgeordneten möglich. Es wird ein Algorithmus zum binären Suchen verwendet. Dazu muß die Datenbank in aufsteigender Reihenfolge nach Namen sortiert sein. Die Suche wird in der Mitte der Datei begonnen. Liegt der gesuchte Name nach dem in der Mitte gefundenen, wird in der oberen Hälfte weitergesucht, sonst in der unteren. In der entsprechenden Hälfte wird wieder die Mitte genommen u.s.w. Das Verfahren ähnelt also der Suche im Telefonbuch. (Eine genaue Beschreibung dieses und vieler weiterer Algorithmen findet sich bei *Wirth, Algorithmen und Datenstrukturen in Modula-2.*)

Grenzen von FormDev

Da das Beispielprogramm neben dem gerade besprochenen Formular noch weitere zur Eingabe des Textes und zur Auswahl von Empfängern benötigt, soll als nächstes ein Formular erstellt werden, daß es wie eine Toolbar erlaubt, zwischen den einzelnen Funktionen hin und her zu schalten.
Die Knöpfe können dabei mit einzelnen Grafiken realisiert werden. Eine optisch ansprechende Gestaltung eigener Programme ist dadurch ohne weiteres möglich.

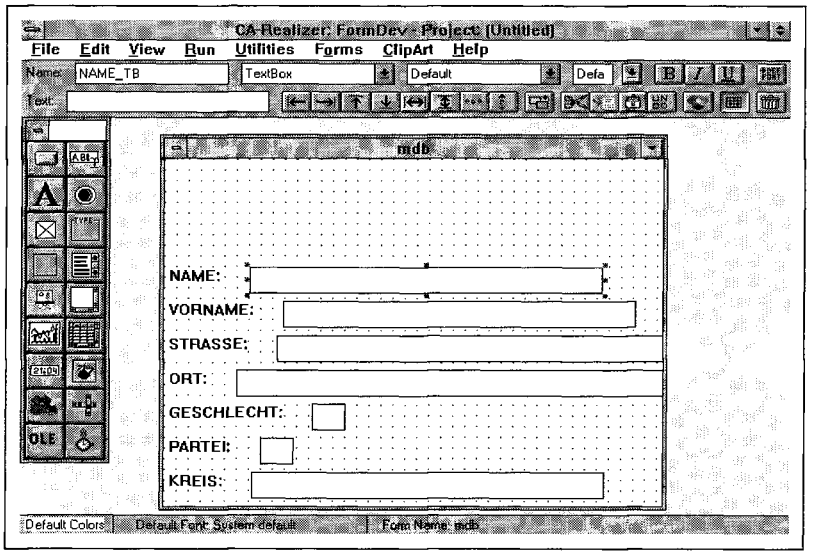
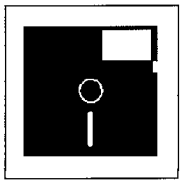
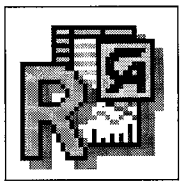


Abb. 2: Visueller Formularentwurf





Das Tool-Formular soll immer das oberste Fenster sein, d.h. kein anderes Fenster soll es überdekken können. Dazu muß eine bestimmte Eigenschaft des Formulars gesetzt werden. Die richtige Einstellung findet man im Handbuch, aber nicht in der Auswahl für Formulareigenschaften. Das heißt, an dieser Stelle muß man die visuelle Seite des Realizer verlassen und direkt im Programmcode arbeiten, den FormDev erzeugt. Diese Zweiteilung zwischen visueller und "normaler" Programmierung erweist sich als eine echte Schwachstelle des Systems.

Wie gesehen, schöpft die visuelle Seite nicht alle Möglichkeiten der Programmierung aus. Auf der anderen Seite wird im Handbuch davor gewarnt, Änderungen an dem Code vorzunehmen, den FormDev erzeugt hat. Mit Recht, denn das kann der point of no return zur visuellen Umgebung sein. Teilweise werden Änderungen, die man im Code vorgenommen hat, in FormDev nicht erkannt und gehen so verloren. Teilweise führen solche Änderungen sogar dazu, daß FormDev mit dem Code gar nichts mehr anfangen kann. Das wird auch der Grund dafür sein, daß es nicht möglich ist, eines der zahlreichen mitgelieferten Beispielprogramme in FormDev zu laden.

In der Testphase des Beispiels wurde eine Beschreibungsdatei zerstört (nicht durch die Entwicklungsumgebung), die FormDev zum Laden des Projekts benötigt. Danach war ein Laden des Projekts, das alle Formulare einer Anwendung zusammenfaßt und unter anderem die Menüstruktur für diese enthält, nicht mehr möglich. Zum Glück war der Programmcode unbeschädigt, so daß die Entwicklung im Realizer weitergehen konnte.

Die Frage stellt sich, ob eine Zweiteilung der Entwicklungsumgebung überhaupt sinnvoll ist, oder ob man besser nur eine visuelle Umgebung anbietet, wie dies z.B. bei MS-VisualBasic der Fall ist. (Zwar arbeitet auch VisualBasic intern mit einem fortlaufenden Programmcode, dieser bleibt dem Entwickler aber weitgehend verborgen.)

Oft wird als Argument gegen rein visuelle Programmierung angeführt, daß man bei großen Projekten keinen Überblick über den gesamten Programmcode hat, da dieser sich zerstückelt hinter den einzelnen Objekten verbirgt. Dies ist häufig richtig, jedoch nicht zwingend. Insbesondere ist der Umkehrschluß nicht gültig, daß der Code in normalen Umgebungen übersichtlicher ist. Sieht man sich den von FormDev erzeugten Code an, so kann von Übersichtlichkeit keine Rede sein.

Visuelle Programmiersysteme sollten daher Werkzeuge bereithalten, die es möglich machen, auch bei großen Projekten die Übersicht zu behalten. Zu nennen wären z.B. grafische Browser für die Darstellung von Objekthierarchien und Querverbindungen zwischen den Objekten.

Programmtest

Ein wichtiger Punkt bei der Programmentwicklung ist die Testphase. Hier benötigt der Programmierer umfangreiche Unterstützung durch die Entwicklungsumgebung. Realizer bietet dazu mehrere Möglichkeiten.

Zunächst kann man einzelne Formulare in FormDev starten und ihre Funktion überprüfen. Leider ist bei Projekten, die mehrere Formulare umfassen oder für mehrere Formulare Prozeduren zentral bereitstellen, so auch beim Beispielprogramm, ein Test des gesamten Projekts nur im Realizer direkt möglich. Das heißt, daß ein häufiges Wechseln zwischen Realizer und FormDev nötig ist. Ärgerlich ist dabei, daß man jedesmal das Projekt neu laden muß.

Zur Ablaufkontrolle bietet Realizer die üblichen Hilfsmittel wie Unterbrechungen, Einzelschritt und Variablenkontrolle.

In der Testphase erwies sich das System als etwas instabil, so daß es öfters von Windows gestoppt wurde. Die genauen Zusammenhänge konnten nicht ermittelt werden. So kann auch nicht ausgeschlossen werden, daß die Probleme letztlich bei Windows zu suchen sind. Dann wäre ein Umstieg auf OS/2 zu überlegen, das vom Realizer unterstützt wird und dem allgemein eine größere Absturzsicherheit zugesprochen wird. Unter Windows kann man dem Entwickler nur raten, seine Programme auch in Extremsituationen wie Speicherplatzmangel usw. zu testen.

Der Project-Builder

Nachdem alle Fehler beseitigt sind, muß aus den Quelltexten noch eine lauffähige Pro-

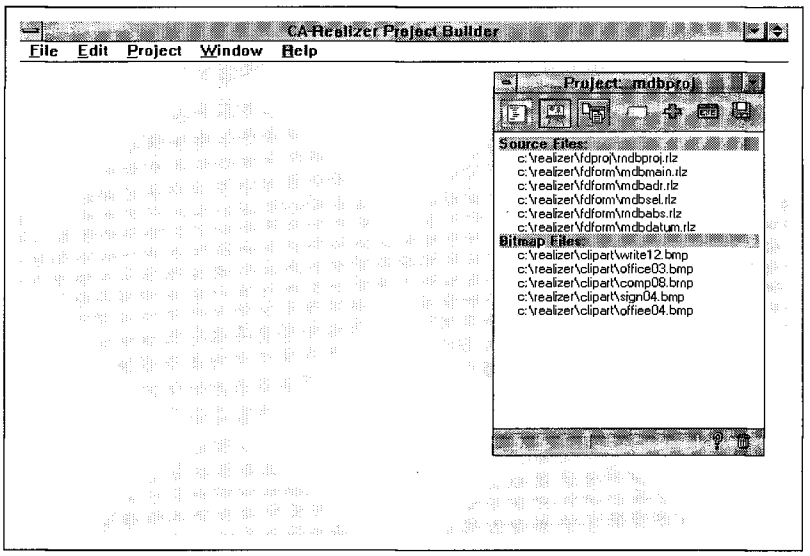
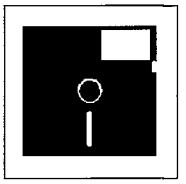
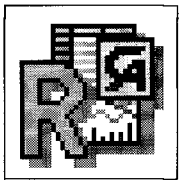


Abb. 3: Der Project-Builder bei der Arbeit





grammdatei erzeugt werden. Dies erledigt beim Realizer der *Project-Builder*. Dieses Programm erstellt eine Liste aller in einem Projekt verwendeten Dateien *(vgl. Abb. 3, auf der Vorseite)*. Erscheinen Dateien in roter Schrift, so muß zusätzlich noch das Verzeichnis, indem sie sich befinden, eingetragen werden. Wird dies nicht getan, warnt der Project-Builder, daß er einige Dateien nicht finden kann. Nach einem Klick auf den EXE-Knopf wird die ausführbare Datei erzeugt. Wie oben bereits erwähnt, benötigt diese Datei das Laufzeitsystem des Realizer. Will man seine Entwicklung verbreiten, muß es mit verteilt werden. Damit auch alle nötigen Dateien vorhanden sind, bietet der Project- Builder die Möglichkeit, eine Installationsdiskette zu erstellen. Diese enthält das Laufzeitsystem, alle zum Programm gehörenden Dateien und ein Installprogramm, das alle Komponenten auf einem anderen System installieren kann.
Das Installationsprogramm läßt sich vielfältig konfigurieren. Damit hat man eine ausgezeichnete Möglichkeit, seine Entwicklungen in professioneller Form zu verteilen.

Application Tools

Eine Besonderheit gegenüber anderen Entwicklungssystemen stellen die *"Programmable Application Tools"* des Realizers dar. Man könnte sie als Ausschnitte aus entsprechenden Anwendungsprogrammen bezeichnen. Durch "Zeichnen" in FormDev oder einige Programmzeilen im Realizer kann man z.B. ein Spreadsheet erstellen. Dahinter verbirgt sich der darstellende Teil einer Tabellenkalkulation. Berechnungen kann man damit allerdings erst durchführen, wenn man die entsprechenden Funktionen programmiert hat.
Im Beispielprogramm wird ein TextLog verwendet. Das ist ein Fenster, daß die Funktionen eines kompletten Texteditors erfüllt. Auf den Inhalt des Fensters kann durch Basic-Funktionen zugegriffen werden, so daß es möglich ist, die Funktionalität zu erweitern. Im Beipiel wird der Text ausgelesen, formatiert, um die Absender- und Empfängerangabe erweitert und schließlich ausgedruckt.
Dieses Konzept findet sich bei allen Application Tools wieder. Dem Entwickler werden die grundlegenden Routineaufgaben abgenommen, die eigentlichen Funktionen muß er dann selbst schaffen. Die Application Tools sind also nicht nur programmierbar sondern auch programmierbedürftig.
Alle *Application Tools* vorzustellen, würde zu weit führen. Hingewiesen sei nur noch auf die Diagrammfunktion, die alle wichtigen Diagrammtypen auf gelungene Weise darstellen kann *(vgl Abb. 4)*.
Durch die *Application Tools* kommt dem Realizer eine Stellung zwischen Programmiersystem und programmierbarem Anwendungsprogramm zu. Ein denkbarer Anwender ist deshalb der Entwickler eines Excel-Makros zur Berechnung von Anwaltskosten, der seine Arbeit vermarkten will, ohne sich der mühsamen Aufgabe der Programmierung einer adäquaten Oberfläche unterziehen zu müssen.
Kein Application Tool, sondern eine eigenständige Anwendung ist der Reportgenerator *CA-RET*. Auf der Basis von XBase-Datenbanken bietet er alle Funktionen zur Erstellung von Berichten, die man auch von einer modernen Datenbank erwarten würde. Im Gegensatz zum Realizer unterstützt CA-RET auch Indexdateien und Relationen.
Ein mit CA-RET erstellter Report ist direkt ausdruckbar. Ausserdem kann der Realizer das Programm über DDE fernsteuern, entsprechende Funktionen sind vorhanden.

Fazit

Ohne Kenntnisse der komplizierten Programmierung von Windows kann man mit Hilfe des FormDev innerhalb kurzer Zeit Programme erstellen, deren Oberfläche den typischen "Windowslook" aufweist – vorausgesetzt natürlich, man hat Grundkenntnisse im Programmieren mit Basic oder Pascal. Dann ist es nach etwa ein bis zwei Tagen Einarbeitungszeit möglich, den ersten Entwurf einer kleinen Anwendung, wie z. B. den des Beispielprogrammes, zu erstellen.
Realizer bietet sich also dazu an, den Umstieg von der DOS- zur Windowsprogrammierung zu wagen. Durch die umfangreichen Application Tools können auch komplexere Darstellungen schnell verwirklicht werden.

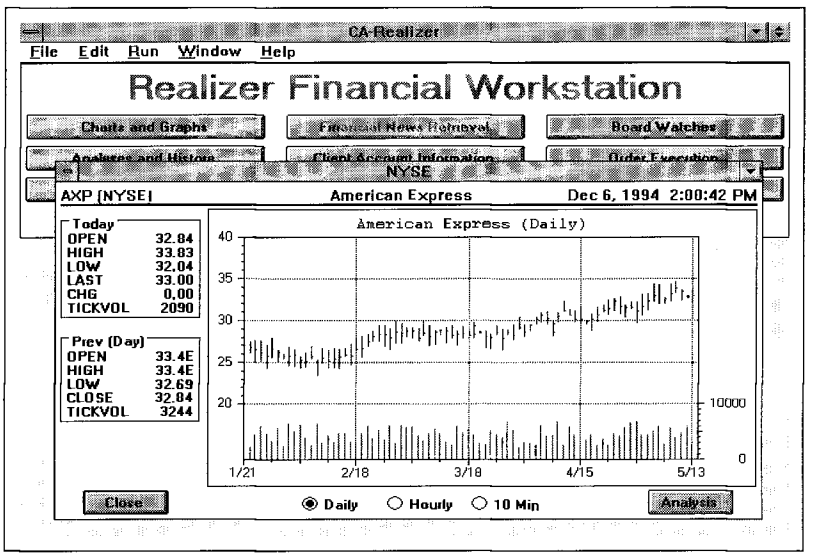
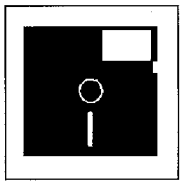
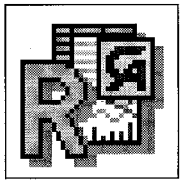


Abb. 4: Die Diagrammfunktion





Die Darstellungen mit Daten zu füllen, bleibt Aufgabe des Entwicklers. Realizer ist eben keine fertige Anwendung, die man programmieren kann, sondern eine Programmierumgebung, die man programmieren muß.
Dem schnellen Zugang zur Windowsprogrammierung stehen die nur eingeschränkten Möglichkeiten des FormDev gegenüber. So schnell man in den FormDev hineinwächst, so schnell wächst man auch wieder heraus. An sich wäre das noch zu verschmerzen, da man die nötigen Verfeinerungen direkt im Realizer vornehmen kann. Wegen des möglichen Verlusts von Änderungen und durch das unkomfortable Hin- und Herschalten zwischen Realizer und FormDev ist dieser Weg aber nur mit Einschränkungen gangbar. In einer neuen Version wäre diesbezüglich dringend Abhilfe nötig und zwar entweder durch Verbesserung der Integration von Realizer und FormDev oder durch Erweiterung des FormDev. Beides zugleich wäre die beste Lösung.
Der Jurist wünscht sich des weiteren auf Grund der zahlreichen XBase-Dateien in seinem Arbeitsfeld eine Verbesserung der XBase-Unterstützung. Es ist unverständlich, daß der Reportgenerator Indexdateien und Relationen unterstützt, die Programmierumgebung aber nicht.
Programmieren, ohne zu programmieren? Nein, das kann man sicher nicht sagen. Aber Windows programmieren, ohne Windows zu programmieren, wäre eine in einem Satz konzentrierte Beschreibung des Realizer.

Zur Diskettenbeilage: Abgeordneten-Korrespondenz

Nach dem Vorbild amerikanischer Programme aus der Public Domain-Landschaft

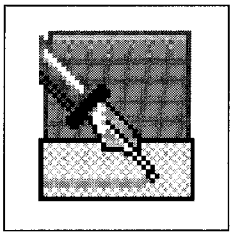
Das Programm MDB-Brief auf der Diskettenbeilage ist eine mit dem Programm CA-Realizer (*vom Autor des Beitrags in diesem Heft, vgl. S. 2904–2908*) erstellte kleine Beispielsanwendung. Die gewählte Aufgabenstellung orientiert sich an zahlreichen Programmen aus der amerikanischen Public Domain-Landschaft, die die Korrespondenz mit den Abgeordneten (sei es als Einzel-, sei es als Serienbrief) zum Gegenstand haben. Das Korrespondenzprogramm benützt die Datei MDB.DBF (dBASE-Format), die selbstverständlich auch selbständig in Verbindung mit anderen Serienbrief-Generatoren genutzt werden kann.

Installation

Aufruf von INSTALL.EXE unter Windows

Installiert wird das Programm unter Windows durch Aufrufen von INSTALL.EXE. Der Installationsvorgang erklärt sich dann selbst. Das Installationsprogramm kopiert das Programm in das angegebene Verzeichnis und die benötigten Laufzeitdateien in das unterhalb des Windows-Verzeichnisses angelegte Verzeichnis RLZRUN20 (= Realizer-Runtime). Um das Programm transparent zu machen, werden die Quelldateien (*.RLZ) mit installiert. Das ist bei Realizer-Programmen nicht notwendig.

Programmfunktionen



1. Text schreiben
Über das Schreibfeder-Icon in der Auswahlleiste am rechten oberen Bildschirmrand startet man einen Editor, der das Erstellen des Brieftextes erlaubt. Der Editor verfügt aber auch über die Funktion "Text einlesen", so daß man vorbereiteten Text übernehmen kann. (Dem Schreibfeder-Icon gleichberechtigt ist das Symbol "Text" am unteren Bildschirmrand.)