Professionell programmieren mit MS Access

Karel Štípek

Einleitung

Das Programmieren ist einfacher geworden. Vor allem Microsoft-Office Produkte, wie Excel und Access bestätigen diesen Eindruck. Verschiedene Assistenten, Beispielprogramme, Auto-Complete Funktionen u.a. nehmen Sie unter die Arme. Einfache Applikationen sind mit ein paar Mausklicken fertig. Es gibt aber auch komplexere Aufgaben und dann können Sie an der einfachen Technik scheitern.

Einige schnellgemachte Lösungen bekomme ich manchmal zur Wartung, z.B. weil der Autor die Firma verlassen hat (möglicherweise war das gerade der Kündigungsgrund, sich von nicht sehr gelungenen Produkten zu lösen). Oder werde ich von jemandem um Hilfe gebeten, wenn das selbstgebastelte Programm schon so komplex ist, dass es mit den einfachen Mitteln nicht mehr weiter geht. Solche Werke kann man meistens nur neu machen – bevor Sie sich ohne jede Dokumentation in die fremde Denkweise hineinarbeiten, sind Sie mit der neuen Lösung halbwegs fertig. Die vorher investierte Zeit geht verloren, weil jemand glaubte, dass das Programmieren so einfach ist.

Ich möchte einige Aspekte und Ideen für die erfolgreiche Software-Entwicklung von komplexeren Aufgaben darstellen ohne Sie dabei mit umfangreichen Analysen- und Datenbanktheorien zu entmutigen. Es werden überwiegend eigene Erfahrungen aus der langjährigen Tätigkeit in der Branche zusammengefasst. Ich will das Programmieren nicht schwieriger machen. Ganz im Gegenteil – es kann noch leichter werden.

Es geht eigentlich nicht nur um das Programmieren selbst. Ich möchte auch einige Probleme und Situationen vorstellen, mit denen man in der Schule selten konfrontiert ist – das wahre Leben ist doch ein wenig anders. Vor allem die Kommunikation mit vielen unterschiedlichen Menschen, die Notwendigkeit auf die sich ständig ändernden Aufgaben zu reagieren, u.v.a. stellen eine ganz andere Art Herausforderung dar, als man in der Schule gewohnt ist.

Mein Artikel hat zwei Teile. Im Ersten werde ich allgemeine Grundsätze für die Problemanalyse präsentieren, im zweiten dann konkrete Tipps für die Entwicklung mit Microsoft-Access anbringen. Umfangreichere Beschreibungen einiger typischer programmtechnischer Aufgaben werde ich in den nächsten Artikeln vorstellen.

Die Problemanalyse

Mit der Programmierung beginnen Sie am Besten damit, dass Sie den Computer AUSschalten. Nein, es ist kein Tippfehler. Und ich werde sogar behaupten, dass Sie desto schneller fertig werden, je länger Sie das liebe Gerät ausgeschaltet lassen Denn zuerst sollten Sie jedes Problem ausreichend analysieren, bevor Sie sich in die Arbeit stürzen.

Die Kunst des Programmierens besteht nicht darin, eine bestimmte Programmiersprache auswendig zu kennen. Wenn Sie wissen, welche Operation oder Datenmanipulation Sie gerade brauchen und können auch richtig begründen, warum gerade diese, ist es keine Schande in der Hilfe oder im Handbuch nachzuschauen. Es geht auch nicht anders – fast jedes Jahr kommt etwas Neues – neue Entwicklungsumgebungen, neue Versionen der alten, dass kann man nicht alles im Kopf behalten. Beachten Sie nur das Microsoft Access; nur in den letzten fünf Jahren habe ich schon mit den Versionen 1, 2, 95, 97 und 2000 gearbeitet.

Das einzige, was „ewig" bleibt, ist das richtige analytische Denken. Ich werde mich freuen, Ihnen einiges davon beibringen zu dürfen.

Welches Programm ist gut?

Viele Wege können zum scheinbar gleichen Ziel führen. Wann ist aber eine Lösung besser als die andere? Warum laufen einige Programme jahrelang erfolgreich und einige kommen nie ordentlich zum Einsatz, weil sie nie fertig werden? Was entscheidet über den Erfolg oder Misserfolg der Arbeit eines Software-Entwicklers?

Funktionsfähigkeit

Die erste Anforderung ist natürlich, dass das Programm genau das tut, was der Anwender will und alle seine Wünsche lückenlos abdeckt. Es ist nicht immer einfach. Später wird noch mehrmals darüber gesprochen, dass der Anwender manchmal (und nicht nur am Anfang) ganz wenig weiß, was er eigentlich will. Er kann sich nämlich nur schwer vorstellen, was alles die moderne Technik (und natürlich auch Sie) für ihn machen können.

Zuverlässigkeit

Das fertige Produkt muss „bombenfest und idiotensicher" sein. Das bedeutet, dass es erstens richtig reagiert , wenn es korrekt bedient wird und zweitens, dass es mit geduldigen und klaren Anweisungen antwortet, wenn es „misshandelt" wird. Kein Bedienungs- oder Datenfehler darf zu einem Absturz führen. Fehlerhafte Eingangswerte müssen schon bei der Eingabe ausgeschlossen werden. Theoretisch sollte es für den Benutzer immer möglich sein, die schon eingegebenen Daten zu korrigieren, bzw. zu löschen. Eine nachträgliche Korrektur kann aber in komplexen Systemen eine ziemlich komplizierte Aufgabe sein.

Bedienungsanleitung

Ziel der EDV ist es, die Arbeit zu erleichtern. Ihr Produkt wird positiv angenommen, wenn die Bedienung möglichst einfach ist und wenn man sich auch ohne aufwendiges Lernen im Programm schnell auskennt.

Die meisten Menschen sind nicht bereit, ein dickes Anwenderhandbuch zu lesen, bevor sie mit der Arbeit beginnen. Genau nach der bekannten Murphy's Regel: *„Erst wenn alle Versuche scheitern, ist es an der Zeit, sich die Bedienungsanleitung anzuschauen."*

Es ist heute auch nicht notwendig ein solches Handbuch zu erfassen, wenn es nicht ausdrücklich verlangt wird. Viel besser sind die EDV-eigenen Lösungen wie Statuszeilentexte, Quick-Infos und vor allem die kontextsensitive Hilfe. Man bekommt mit der Taste in jedem Stand des Programmablaufs eine Information angezeigt, die sich auf die jeweilige Situation bezieht. Die Seite kann ausgedruckt werden, das ist aber meistens nicht notwendig.

Besonders wichtig sind die ausführlichen und exakten Erklärungen aller unrichtigen oder fehlenden Eingaben und aller möglichen Ausnahmezustände des Programms mit dem entsprechenden Lösungsvorschlag.

Ergonomie

Es ist günstig, wenn die Oberfläche, die Menüstruktur und die Tastenkombinationen (Hot-Keys) möglichst ähnlich den Programmen sind, mit denen der Anwender bereits schon arbeitet. Es wird zwar manchmal darüber diskutiert, ob Microsoft-Office gerade das beste Beispiel für eine Benutzeroberfläche ist, eines ist aber sicher. Wer jemand gewohnt ist, mit den Menüs im Word und Excel zu arbeiten, wird er logisch von Ihrem Programm die gleichen Menüpunkte für die gleichen Tätigkeiten (Datei öffnen, Programm beenden, usw.) auch verlangen.

Die oben genannten Punkte reichen eigentlich. Sie bekommen einen Auftrag, erstellen ein Produkt, testen es, übergeben es an den Kunden und damit ist es fertig. Das wirkliche Leben ist aber mannigfaltiger und es kommen weitere Kriterien, die die wahre Qualität des Programms bestimmen. Und die sind für Ihren Erfolg genauso wichtig, wie die oberen, auch wenn sie der Anwender nicht unmittelbar beurteilen kann, weil sie für ihn mehr weniger unsichtbar bleiben.

Anpassungsfähigkeit

Sie sollen das Programm so gestalten, dass es jederzeit und ohne besonderen Aufwand veränderbar ist. Warum? Der Kunde sagt

doch, was er will und Ihre Aufgabe ist es, die beste technische Lösung zu realisieren.

Es ist nur rein theoretisch so. Erst nachdem Sie das Programm teilweise oder sogar fertig haben, kommt der Kunde darauf, was für ihn noch besser wäre. Nachdem er sieht, wie gut Ihr Programm ist, wird er sich Erweiterungen wünschen, an die er in der Anfangsphase überhaupt nicht denkt.

Außerdem ändern sich laufend die Rahmenbedingungen – Gesetze, Firmenstruktur, Produktsortiment, usw. Die meisten Programmänderungen werden von Ihnen beim Jahreswechsel verlangt (und nicht nur bei dem letzten vom 1999 auf 2000). Und es kann noch schlimmer werden. Einige Jahresabschlusstätigkeiten werden nicht im Jänner, sondern auch einige Monate später durchgeführt. Ihr Programm muss dann einige Zeit z.B. zwischen den Berechnungsalgorithmen des Vorjahres und des neuen Jahres umschalten können.

Langlebigkeit

Sie werden sich bestimmt wünschen, dass Ihr Produkt lange im Einsatz ist und das es Ihnen auch nach Jahren durch die Erweiterungs-, bzw. Wartungsaufträge Geld bringt. Dafür muss es erstens anpassungsfähig (siehe oben) und zweitens gut dokumentiert sein. Es ist nicht nur schwierig, ein Programm weiter zu entwickeln, wenn der Autor nicht mehr verfügbar ist. Es ist auch verdammt schwierig, sich nach einiger Zeit in eigenem Werk schnell zu orientieren. Sie müssen sich zwingen, während der Arbeit auch die Teile zu dokumentieren, die Ihnen zu dem Zeitpunkt der Programmerstellung absolut klar und unkompliziert erscheinen.

Analyse - Von der Idee zu der EDV-Lösung

Die Arbeit an einem EDV-Projekt fängt damit an, dass sich zwei Gruppen von Menschen in Besprechungen treffen: die zukünftigen Anwender und Sie (eventuell mit Ihren Kollegen) als Software-Entwickler. Beide Gruppen sprechen vom Anfang an ganz unterschiedliche Sprachen. Die ersten haben die reale Welt, z.b. konkrete Produkte ihrer Firma im Kopf, die anderen wieder die EDV-Implementierung – Tabellen, Masken, Berichte, usw. Welche Grundsätze können die Verständigung verbessern und dadurch die Effizienz der Analysephase erhöhen?

Nichts ist unmöglich

Die wichtigsten Ansprechpartner, die man für die Problemanalyse braucht, sind Menschen mit langjährigen Erfahrungen in ihrem Bereich, oft aber nicht in der EDV. Sie haben erstens weniger Vorstellungen, was der Computer überhaupt alles kann, zweitens bestimmtes Misstrauen und die Älteren manchmal teilweise Angst, die neue Technik zu bedienen. Sie stellen oft vorsichtige Fragen, ob das oder das im Programm möglich wäre. Die einzige richtige Antwort ist: *„Ja, alles ist möglich“*. Nur so können Sie wirklich erfahren, was der Anwender von der zukünftigen Lösung erwartet. Auch wenn Sie den Eindruck haben sollten, dass die Aufgabe unlösbar oder sehr kompliziert ist, ignorieren Sie das in der Analysephase. Sie werden sehen, Sie werden es schon schaffen. Manches Problem wird einfacher, wenn es in die streng logische Sprache eines Programmanalytikers übersetzt wird.

Top-Down Technik

Genauso wie eine Firma eine hierarchische Struktur hat, hat auch eine EDV-Lösung eine bestimmte Hierarchie. Die Analyse eines Problems muss von der groben Teilen stufenweise bis in kleinste Detail durchgeführt werden. Hier ist es unter anderem Ihre Aufgabe, die Diskussion zu moderieren, bzw. etwas abzubremsen. Wenn Sie es zulassen, dass sofort am Anfang die Fachleute ins tiefste Detail gehen, verlieren Sie und auch Ihre Partner den gesamten Überblick. Es ist enorm mühsam, die grundsätzlichen groben Zusammenhänge, die man am Anfang nicht richtig erkannt hat, später ins Programm zu implementieren.

Graphische Darstellung

Ein Bild sagt mehr als tausend Worte. Zeichnen Sie viele Diagramme, in denen Sie den Aufbau des Programms, das Datenmodell, u.v.a. darstellen. Sie werden Ihnen auch bei der Anwendung der Top-Down Technik helfen – zuerst werden die Module der oberen Schicht als eine Einheit analysiert, dann geht man erst tiefer ins Detail. Die moderne Technik bietet verschiedene Mittel, mit denen Sie sich Zeit sparen können. So werden bei unserer Firma z.B. die beschriebenen Blätter (Flip-Charts) mit einer Digitalkamera fotografiert und dann per Mail an alle Besprechungsteilnehmer als Beilage des Protokolls verteilt.

Fachliche Ausbildung

Je mehr Sie über die fachlichen Probleme wissen, desto besser werden Sie Ihre Kunden verstehen. Sie können natürlich nicht in einigen Stunden das Wissensniveau erreichen, was die anderen jahrelang gesammelt haben. Es ist aber zumindest wichtig, dass Sie sich in den Grundbegriffen gut auskennen. Das Pflichtenheft sollte deswegen als eines der ersten Kapitel eine Begriffserklärung enthalten. Sie ersparen später viel Zeit für das Umprogrammieren von falsch verstandenen Aufgaben und Zusammenhängen.

Die Begriffserklärung ist nicht nur für Sie als Entwickler wichtig. Bei einer großen Firma werden die EDV-Tätigkeiten nicht immer optimal koordiniert. Im Laufe der Jahre entstehen in mehreren Abteilungen Programme für den unmittelbaren Bedarf, die als „Insellösungen“ bezeichnet werden. Wenn Sie ein Problem analysieren, sollten Sie sich bemühen, diese Teillösungen durch die neue Entwicklung „unter einen Hut“ zu bringen. Dabei ist es auch wichtig, die Bedeutung der Fachbegriffe auch unter den Fachleuten selbst zu vereinheitlichen. Vor allem geht es um verschieden Kodierungssysteme, Definitionen von Sammelbegriffen, Gruppierungen, Auffinden von eindeutigen Schlüsselfeldern für die Identifizierung der Datensätze, usw.

Keine Frage ist blöd

Lassen Sie kein Problem offen. Wiederholen Sie die Aussagen Ihrer Partner mit Ihren eigenen Worten und fragen Sie, ob Sie alles richtig verstanden haben. Jede Kleinigkeit, die bei der händischen Datenführung problemlos z.B. mit einer Notiz auf der Registrierkarte gelöst wird, muss in Ihrem Programm berücksichtigt und ausprogrammiert werden. Ihre gezielten Fragen werden auch den Fachleuten zugute kommen, sie werden dadurch bestimmt auch einige weniger sichtbare Zusammenhänge in ihrem Bereich entdecken.

Lassen Sie sich Zeit

Wenn möglich, lassen Sie sich Zeit genug für die Analyse. Die Softwareentwicklung ist in jedem Fall ein zyklisches Prozess, bei welchem die Anforderungen der Anwender mit Ihren Realisierungsvorstellungen abgeglichen werden. Schreiben Sie nach jeder Besprechung (möglichst unmittelbar danach, wenn Sie noch alles frisch im Kopf haben) ein Protokoll, fassen Sie die besprochenen Themen zusammen und bitten Sie Ihre Partner um die Stellungnahme. Dabei fallen Ihnen auch Fragen auf, die Sie als Themen für die nächste Besprechung vorschlagen können.

Prototyp erstellen

Sobald es möglich wird, die Anwenderwünsche teilweise zu visualisieren, machen Sie ein Programmprototyp. Ein paar leere Masken nur mit Feldern, Aufschriften und Schaltflächen ohne jede Funktionalität werden Sie nicht viel Zeit kosten und reichen vollkommen. Der Kunde wird schneller erkennen, was Sie ihm als Lösung anbieten können, und es fallen ihm neue Ideen ein. Einen Prototyp können Sie bei Bedarf einfacher ändern als später komplett fertige, funktionsfähige Programmteile.

Etwas Psychologie

Sie werden auf Ihrem Weg auf mit Problemen konfrontiert, die mit der Programmierung selbst nichts zu tun haben. Man darf nicht vergessen, dass nicht die Computer sondern die Menschen im Spiel sind. Und die haben auch Eigenschaften, die man berücksichtigen muss, wenn man wirklich etwas Gutes schaffen will.

- Verlieren Sie bei den Besprechungen auch ein paar nicht sachliche Sätze. Lernen Sie Ihre Partner auch ein wenig von ihrer privaten Seite kennen. Zu einer wirklich guten Zusammenarbeit können Sie niemanden zwingen. Wenn sich jeder Teilnehmer wohl fühlt und die Atmosphäre entspannt ist, kommen Sie auch in rein sachlichen Fragen schneller zum Ziel.

- Seien Sie vorsichtig mit der Kritik, auch wenn sie berechtigt ist. Es ist zu erwarten, dass Sie besonders was die EDV-Kenntnisse angeht, den älteren Kollegen weit überlegen sind. Diese haben dagegen viel mehr Erfahrungen aus der Praxis, ohne die Sie schwierig weiter kommen. Haben Sie ein gemeinsames Ziel, nicht das persönliche Prestige vor den Augen.

- E-Mail ist heute neben dem Gespräch die optimale Form der Kommunikation, wirkt aber doch ein bisschen zu unpersönlich. Wenn es Ihnen zeitlich möglich ist, begleiten Sie das Verschicken einer Mail auch mit einem kurzen Telefongespräch. So übergeben Sie die genaue technische Information schriftlich und pflegen dabei auch die menschlichen Beziehungen.
- Wenn Ihr Programm ein älteres auflösen soll, achten Sie darauf, dass nichts von dem alten, was gut war, im neuen fehlt. Bemühen Sie sich, dass die Berichte vom Layout her, möglichst ähnlich bleiben. Die Notwendigkeit einer Umstellung der gewohnten Arbeitsweise wird immer negativ angenommen und manchmal beurteilen die Anwender Ihr großes Werk nach unglaublichen Kleinigkeiten.

Pflichtenheft

Die Ergebnisse der Analysephase sollten Sie in einem Pflichtenheft festlegen. Das stellt dann die Grundlage für den Projektauftrag und den Kostenvoranschlag dar. Wenn Sie ohne ein Pflichtenheft einfach alle Wünsche des Anwenders laufend erfüllen wollen, kann es passieren, dass Sie nie fertig werden. Wenn später eine größere Erweiterung Ihres Produkts erwünscht wird, die über den Rahmen des Pflichtenheftes hinausgeht, wird der erste Auftrag zuerst abgeschlossen und dann ein neuer formuliert.

Realisierung mit Microsoft-Access

Wenn in dem zweiten Teil des Artikels über Funktionen gesprochen wird, werden eigentlich Prozeduren oder Funktionen gemeint. Die Funktionen sind im Einsatz den Prozeduren etwas überlegen, der Unterschied wird später erklärt.

Allgemeine Tipps

Sicherheit

Wie der Klassiker Murphy sagt: *„Wenn etwas schief gehen kann, dann geht's auch einmal schief."*

Es kann immer etwas passieren. Nicht nur der Festplatten-Crash ist das Gespenst. Die meisten haben es schon erlebt, dass sich aus unbekannten Gründen plötzlich eine Datenbank nicht öffnen lässt. Wenn Sie eine Fehlermeldung bekommen, dass die Datei nicht in Ordnung ist, lässt sie sich noch manchmal mit Access reparieren. Wenn Access beim Öffnungsversuch sofort mit einer Speicherschutzverletzung endet, ist meistens keine Abhilfe möglich. Und Sie können nur zu einer Sicherungskopie greifen, aber nur dann, wenn Sie eine haben.

Das Schlimmste ist nicht nur das, dass Sie etwas neu eintippen müssen. Es geht viel mehr darum, dass Sie wahrscheinlich nicht alle kleinsten Schritte durchgehend protokollieren und wissen eigentlich nicht genau, was alles neu zu machen ist. Natürlich (wieder Murphy) kommt es nicht zu Problemen, wenn Sie in der Entwicklungsphase sind und Sie noch relativ viel Zeit haben. Die Datenbank wird unbrauchbar, wenn in einer halben Stunde die Präsentation Ihrer zweimonatiger Arbeit stattfinden soll.

Ein paar Tipps von einem, der schon Einiges verloren hat:

- Machen Sie sich spätestens jede halbe Stunde eine lokale Kopie der Datenbank. Das geht sehr schnell über die Ablage – Sie markieren die Datei, drücken Strg C und Strg V und schon sehen Sie eine neue durchgehend nummerierte Datei **„Kopie (...) von ..."** im gleichen Verzeichnis. Wenn Sie die Option ***„Standard Öffnungsmodus"*** auf ***„Gemeinsame Nutzung"*** eingestellt haben, brauchen Sie nicht einmal die Datenbank schließen.
- Komprimieren Sie am Ende jedes Arbeitstages Ihre Datenbank in eine **ZIP**-Datei, schreiben Sie auch das Datum im Format **JJMMTT** in den Namen und legen Sie sie in einem extra Verzeichnis **„Save"** ab. Damit Sie auch vor dem Festplatten-Crash sicher sind, kopieren Sie die Datei auch auf ein regelmäßig gesichertes Laufwerk in Ihrem Firmennetzwerk oder auf Diskette, wenn Sie zu Hause sind. Danach löschen Sie alle im vorigen Absatz beschriebenen Zwischenkopien, am besten unter der Aktivierung des Papierkorbs, damit sie doch lieber nicht sofort endgültig entfernt werden. Den Papierkorb leeren Sie eventuell erst am nächsten Tag vor der Tagessicherung.
- Es ist nicht nur sinnvoll, dass Sie den letzten Stand der Entwicklung sichern. Es kommt auch einmal der Auftraggeber und sagt: *„Seien Sie uns, bitte, nicht böse, aber die Maske, die Sie uns vorige Woche gezeigt haben, die hat uns doch besser gefallen, wir sollten die nicht ändern lassen. Ist es für Sie sehr umständlich den alten Stand wieder herzustellen?"* Dann werden Sie Ihr Sicherungskonzept und die laufende Dokumentierung (darüber wird noch gesprochen) sehr hoch schätzen.
- Wie lange Sie die Tagessicherungen aufbewahren, hängt davon ab, wie groß Ihre Festplatte ist, bzw. wie laut Ihr Server-Administrator die Richtlinien über Ausnutzung des Speicherplatzes am Server wiederholt. Speichern Sie aber auf jeden Fall für immer alle Versionen des Programms, die Sie in die Produktion abgegeben haben. Glauben Sie Ihren Anwendern nicht, dass sie die drei Jahre alte Programmversion sicher nie mehr brauchen. Es kann sein, dass sich im Laufe der Jahre auch die Tabellenstrukturen ändern. Das neue Programm kann dann logisch nicht mit den alten Daten arbeiten, weil z.B. die neueren Felder damals noch nicht existierten. Und was machen Sie dann, wenn jemand die alte Statistik noch einmal ausdrucken will?

Namenskonventionen

Sie arbeiten mit vielen unterschiedlichen Datenbank-Objekten, die die gleiche logische Bedeutung haben, also gleich benannt werden könnten. Wie unterscheiden Sie aber, ob der Name „`Kunden`" eine Tabelle, ein Formular oder einen Bericht bezeichnet?

Eine bewährte Lösung ist es, für die Namen der Objekte je nach dem Typ ein bestimmtes Präfix zu verwenden. So kann z. B. „`tblKunden`" eine Tabelle und „`rptKunden`" einen Bericht bezeichnen. Genauso wie für die Datenbank-Objekte sind die Präfixe auch für die Benennung von Tabellenfeldern, Steuerelementen und Variablen je nach dem Typ nützlich. Jedes Access-Buch enthält sinnvolle Vorschläge zu dem Thema. Einige verfeinerte Präfixe aus meiner Praxis werden später in den objektspezifischen Kapiteln dargestellt.

Allerdings muss man nichts übertreiben. Die Verwendung der Präfixe bei allen Namen im Programm ist nicht notwendig. Wenn Sie wissen, dass Sie auf einige Steuerelemente nicht im Kode verweisen brauchen, lassen Sie ganz ruhig die vom Access automatisch vergebenen Namen. Sie sollten sie aber sofort „richtig" benennen, wenn Sie sie doch einmal im Kode ansprechen wollen. Es gibt nichts Schlimmeres für die Wartung, als z.B. Anweisungen wie „`Text94.Visible = True`". Dann müssen Sie immer zuerst in das Formular schauen, welches Feld `Text94` eigentlich ist, bevor Sie etwas im Kode ändern können.

Auch bei den Namen von lokalen Variablen in Funktionen können Sie auf die Konventionen verzichten, weil Sie sowieso in der Deklaration den Typ problemlos sehen können. Außerdem können Sie im Rahmen einer Funktion den Überblick behalten auch wenn die Variablennamen ganz kurz und dadurch nicht besonders sprechend sind.

Ein weiteres Thema ist die Reihenfolge der Verben und Hauptwörtern z.B. in Funktionsnamen.

Soll eine Funktion „`BearbeiteKunden`" oder „`KundenBearbeiten`" heißen? Es ist eigentlich egal!? Gut, wenn das gleiche System im ganzen Programm angewendet wird. Ich verwende die in der objektorientierten Programmierung übliche Syntax `<Objekt>.<Methode>`, es kommt also zuerst das Hauptwort und dann das Verb.

Es ist bei der Entwicklung unvermeidlich, verschiedene Testobjekte, ältere Versionen, u.a. in der Datenbank temporär abzulegen. Sie werden sich Zeit und Speicherplatz sparen können, wenn Sie alle Objekte, die nicht endgültig für die produktive Version des Programms geplant sind, gesondert benennen. Vorschlag: Präfix „`a_`". Wenn Sie vor der Auslieferung des Programms die Liste der Objekte im Datenbankfenster alphabetisch sortieren, stehen diese temporären Sachen am Anfang und Sie können sie löschen und danach die Datenbank komprimieren.

Die Namen in Access dürfen zwar Leerzeichen enthalten, es ist aber nicht empfehlenswert. Erstens kann es zur Verwechslung kommen (ein oder zwei Leerzeichen sind schwierig zu erkennen), zweitens müssen Sie solche Namen bei der Verwendung z.B. in SQL-Ausdrücken immer in eckige Klammern einschließen. Bei

Namen, die aus mehreren Wörtern bestehen ist es besser, entweder durch die Groß/Kleinschreibung oder durch Underscore („_“) die Wörter optisch zu trennen.

Laufende Dokumentation
In diesem Kapitel geht es um die Dokumentation für die Entwickler. Die Anwenderdokumentation wurde im Kapitel „Welches Programm ist gut?“ behandelt.

Ein Access-Programm kann recht umfangreich sein. Mehrere Hundert Objekte in einer Datenbank sind keine Ausnahme. Wie können Sie dabei den Überblick nicht verlieren? Eine bestimmte Disziplin, wie z.B. die o.g. Namenskonventionen und laufende Dokumentation sind unentbehrlich.

Das Wort „laufend“ sei betont. Der normale Vorgang ist der, dass Sie zuerst das Programm möglichst schnell fertig haben wollen und dann einmal, irgendwann, schreiben Sie die Dokumentation. Die ersparte Zeit beim Entwickeln wird viel mehr Aufwand kosten, wenn Sie Ihr Produkt nachträglich dokumentieren wollen.

Die einfachste Form der Dokumentation sind die Kommentare im Programmkode. Lieber mehr als weniger. Jede Funktion sollte eine Standardbeschreibung gleich nach der Definitionszeile enthalten. Der Name des Autors, Datum der Erstellung, bzw. der Änderung sind nicht so wichtig, wenn das Unterprogramm nur im Rahmen Ihres Programms eingesetzt wird. Folgende Informationen sind aber unverzichtbar:

- Beschreibung der Funktion und die Bedeutung des Rückgabewertes.
- Erklärung aller Aufrufparameter
- Angabe aller Programmteile, wo die Funktion verwendet wird. Das ist besonders wichtig. Wenn Sie die Funktion verändern, müssen Sie natürlich alle betroffenen Programmteile testen. Verlassen Sie sich nicht darauf, dass Sie mit der Suche eines Namens in der gesamten Datenbank alle Aufrufe automatisch finden. Sie können Ihre Funktionen auf im SQL-Ausdruck einer Abfrage oder direkt im Ereignisfeld eines Formulars oder Steuerelements eintragen und solche Stellen finden Sie mit der Suche nicht.

  Außer der Kommentare im Programmkode ist es empfehlenswert, weitere begleitende Dokumente währen der Programmentwicklung zu führen, wie z.B:

- Graphische Darstellung der Programmstruktur. Sie enthält die Information über die Aktionen, die durch die Aktivierung der Menüpunkte oder Schaltflächen ausgelöst werden, bzw. über die Hierarchie der verschachtelten Funktionsaufrufe.
- Graphische Darstellung des Datenflusses. Die Veranschaulichung des Weges von den Tabellen über Abfragen bis zur Darstellung im Formular oder Bericht ist für die schnelle Fehlersuche besonders hilfreich. Der Anwender meldet Ihnen einen Fehler in einer Spalte eines Berichts. Sie müssen dann schnell entscheiden: liegt es an den eingegebenen Daten oder haben Sie eine falsche Formel eingebaut? Entweder können Sie mehrere Objekte nacheinander öffnen und die Datenquellen suchen oder schauen Sie auf ein übersichtliches Blatt Ihrer Dokumentation.
- Ein Systemhandbuch. Hier sollen in einer für den Entwickler (nicht unbedingt für den Anwender) optimalen Form die wichtigen internen Algorithmen und Abläufe beschrieben werden.

  Noch schwieriger als die Dokumentation das erste Mal zu erstellen ist es, sie auch bei den späteren Programmänderungen immer aktuell zu halten, sonst wird sie bald unbrauchbar.

  Informationen in elektronischer Form können auch verschiedene handschriftliche Notizen ersetzen. Sie sind besser lesbar, Sie können sie strukturieren, durchsuchen, an Ihre Mitarbeiter weitergeben. Zwei solche Notizensammlungen möchte ich Ihnen empfehlen.

- Alle noch nicht erledigten Teilaufgaben schreiben Sie in eine „ToDo“ Datei. Sie wird in zwei Hauptkapiteln aufgeteilt. Das erste enthält alle Sachen, die sofort zu erledigen sind (z.B. gefundene Fehler), das zweite Änderungswünsche und Verbesserungsvorschläge mit etwas niedrigerer Priorität. Was fertig ist, wird aus dieser Datei gelöscht.
- Evidenz des Entwicklungsfortschritts, bzw. der Programmänderungen können Sie in einer „LOG“ Datei festhalten. Mit Hilfe dieser Datei können Sie im Notfall Ihre Arbeit wiederholen, wenn Sie nach einem Systemabsturz auf eine Sicherungskopie älteres Datums zugreifen müssen. Weiter können Sie genau verfolgen, wann Sie eine Änderung gemacht haben. Das ist bei der Fehlersuche wichtig, wenn Sie eine ältere (noch fehlerfreie) Programmversion mit der aktuellen vergleichen brauchen.

  Speichern Sie über längeren Zeitraum (in einem extra Ordner) alle Mails, die Sie im Zusammenhang mit dem jeweiligen Projekt empfangen oder verschickt haben. Es kann in einem Team auch zu Streitigkeiten kommen, wenn der Abgabetermin naht und die Arbeit noch nicht fertig ist. Für den Notfall haben Sie klare Beweise in der Hand, z.B. wann Sie jemanden um Informationen gebeten haben und die Antwort nicht bekommen haben, usw. Ja, natürlich ist es für die Team-Zusammenarbeit viel besser, wenn man so was nie braucht. Es ist doch wichtiger, Erfolg zu haben, als eigene Unschuld am Misserfolg beweisen zu können.

Entwicklung und Produktion
Schaffen Sie sich die Möglichkeit beim Ablauf des Programms zwischen Ihnen als Entwickler und dem „normalen“ Anwender zu unterscheiden. Eine Lösung dafür ist z.B. ein spezieller Username (mit Passwort), das eine globale Variable setzt, die dann im Kode abgefragt werden kann. So können Sie dann, wenn Sie das Programm testen, das Verhalten Ihren Bedürfnissen anpassen, wie z.B:

- Fehlerbehandlungsroutine abschalten, damit Sie im Falle eines Laufzeitfehlers auf der richtigen Kode-Zeile stehen bleiben.
- Zusätzliche Informationen, z.B. Meldungen über den Programmablauf, Aufrufparameter der Funktionen, diverse Zwischenergebnisse ausgeben.
- Sanduhr (Hourglass) nicht aktivieren, Veränderungen am Bildschirm zulassen (Echo)
- Sicherheitsabfragen (z. B. „Wollen Sie die Veränderungen speichern“) deaktivieren

  Diese Maßnahmen sparen Ihnen besonders bei der wiederholten Fehlersuche und Testen viel Zeit.

Never change a winning team
Die Arbeit muss Spaß machen. Ein guter Software-Entwickler betrachtet seine Tätigkeit nicht nur als eine mechanische Erfüllung der vorgelegten Anwenderwünsche, sondern ist es für ihn auch eine Art Herausforderung und Leidenschaft, gute, übersichtliche und effiziente Lösungen zu realisieren.

Auch wenn das Programm gut läuft und alle damit zufrieden sind, werden Sie manchmal sehen, dass Sie etwas besser machen konnten. Und Sie ändern dann im Programm nur eine unbedeutende Kleinigkeit, die Ihr Werk durch vergessene weitere Konsequenzen total vernichtet. Bitte, machen Sie unnötige Änderungen nur dann, wenn Sie genug Zeit für das ausführliche Testen des gesamten Programms (nicht nur des veränderten Teils) haben. Und vergewissern Sie sich vorher, dass Sie eine zuverlässige, getestete Version gespeichert haben, damit Sie im Notfall Ihre Versuche problemlos rückgängig machen können.

Die Funktionsfähigkeit und die Abgabetermine müssen (leider) doch vor der Programmiererkunst Vorrang haben. Nein, ich bin selbst nicht anders. Über unnötige Probleme und Stress, die ich mir schon dadurch angetan habe und noch immer antue, könnte ich ganz dicke Bücher schreiben ...

Tipps zu einzelnen Objekttypen

Tabellen

Präfix-Vorschläge

`tbl`	Basistabelle mit Daten, die vom Anwender eingegeben, bzw. importiert wurden
`stbl`	Projektdefinitionen (Systemtabelle), die vom Entwickler einmalig befüllt wurde
`tbs`	Hilfstabelle für die Entwicklung, die beim Programmablauf nicht verwendet wird

`qry` berechnete Tabelle (gespeicherte Abfrage – deswegen Abfragen-Präfix)

Trennen Sie Daten vom Programm

Speichern Sie alle Anwender-Tabellen (`tbl`...) in einer anderen MDB-Datei als die restlichen Datenbankobjekte und verknüpfen Sie sie mit der Programm-MDB. Sie können dann eine neue Programmversion liefern und die Anwender-Daten bleiben dabei unverändert. Außerdem können Sie mit dieser Technik:

- mehrere Daten-MDBs betreiben und die Verknüpfung mit dem Programm ändern. Sie beliefern z.B. die Anwender in ganz Österreich mit Daten, wobei für jeden nur die Angaben seines Bundeslandes relevant sind. Dann bekommt jeder die gleiche Programm-MDB und eine der neun vorbereiteten Daten-MDBs. Bei der Anmeldung wird aus einer User-Tabelle das jeweilige Bundesland ausgelesen und die Verknüpfung zu der richtigen Daten-MDB automatisch hergestellt.
- die Programm-MDB auf einem lokalen und die Daten-MDB auf einem regelmäßig gesicherten Netzlaufwerk speichern. So gewinnt der Anwender viel an Sicherheit. Ihr Programm können Sie jederzeit neu liefern. Die eingegebenen Daten sind dagegen von dem Arbeitsaufwand her viel wichtiger und müssen gesichert werden.

Entwurf des Datenmodells

Vermeiden Sie die Speicherung von redundanten (unnötigen) Daten, dadurch dass Sie die 1: N und M:N Beziehungen erkennen und das Datenmodell (Tabellenstrukturen, Schlüsselfelder und Beziehungen) richtig entwerfen. Überprüfen Sie aufgrund von geplanten Maskeninhalten und Berichten, ob Sie mit Abfragen die erforderlichen Daten aus den Tabellen einfach ableiten können.

Definieren Sie eindeutige Schlüssel und die Regeln für die referentielle Integrität, um Dateninkonsistenzen zu vermeiden. So wird es z. B. nicht möglich sein, den Datensatz in einer Master-Tabelle zu löschen, wenn es noch in der sekundären Tabelle Datensätze mit dem gleichen Schlüsselwert gibt.

Datensätze richtig identifizieren

Verwenden Sie die automatisch generierten eindeutigen primären Schlüssel anstatt von explizit vergebenen Nummern, bzw. Nummernkreisen für die Identifizierung der Datensätze. Die sog. sprechenden Schlüssel werden manchmal damit begründet, dass man aus dem Feld direkt weitere Attribute sichtbar werden. So kann z.B. die erste Stelle einer Personalnummer die Bezeichnung der Abteilung enthalten. Wozu ist aber die Information, nachdem die Mitarbeiter die Abteilung wechseln oder die Firma umstrukturiert ist? Wollen Sie in allen Tabellen, wo die Nummer auftritt, sie wieder ändern? Zusätzliche Angaben können problemlos in anderen Feldern, bzw. anderen Tabellen gespeichert sein und trotzdem in jeder Maske oder Bericht angezeigt werden.

Eingegebene (importierte) und berechnete Datenfelder

Unterscheiden Sie zwischen den Datenfeldern in den Basistabellen, die direkt eingegeben oder importiert werden und den berechneten Feldern. Die Ersteren können z.B. den Präfix je nach ihrem Typ enthalten (z.B. `strNachname`, `intMonat`) und die berechneten längere sprechende Namen ohne Präfix haben (z.B. `GesamtPreis`)

Gleiche Namen für gleiche Objekte

Die Datenfelder, die zwar in diversen Tabellen liegen, aber gleiche Bedeutung und gleiche Werte haben, sollen auch identische Namen besitzen. Das gilt besonders für die Schlüsselfelder, die erstens als primäre, zweitens als fremde Schlüssel in den Tabellenbeziehungen auftreten.

Abfragen (Queries)

Präfix-Vorschläge

`qry` Abfrage mit einer konstanten Definition (SQL-Ausdruck)

`qrd` dynamische Abfrage, derer Definition während des Programmablaufs verändert wird

`qrs` Hilfsabfrage für die Entwicklung, die beim Programmablauf nicht verwendet wird

Vermeiden Sie zu komplexe Abfragen

Die Access-Fehlermeldung **„Abfrage zu komplex"** gehört zu den unangenehmsten Erscheinungen, denen Sie auf Ihrem Weg begegnen können. Sie kommt z.B. gerade dann, wenn Sie in einem fast fertigen Programm nur ein zusätzliches Feld in eine Abfrage einfügen wollen. Die Abhilfe zu schaffen, ist nicht immer einfach. Bemühen Sie sich, die Anzahl der Tabellen, die in einer Abfrage verknüpft werden, bzw. die Anzahl der selektierten Felder zu reduzieren. Wenn Sie mehrstufige Selektionsabfragen unbedingt brauchen, speichern Sie die Zwischenergebnisse in einer temporären Tabelle (Präfix `qry` - siehe oben). In Access Version 97 können Sie sich jedenfalls mehr leisten als in Access Version 2.

Verwenden Sie dynamische Abfragen

Sie müssen nicht jede Abfrage, die Sie im Programm brauchen, extra als ein benanntes Datenbank-Objekt speichern. Jede Abfrage kann dynamisch sein, das bedeutet, dass Sie einen SQL-Ausdruck beim Programmablauf im Kode definieren und dann der Abfrage (Präfix `qrd`...) zuweisen können. So können Sie z.B. nach der Eingabe der Selektionskriterien in einem Formular den SQL-Ausdruck gleich mit den konkreten Werten bilden und auf die Verweise auf die Formularfelder verzichten. Die Abfrage ist dann funktionsfähig auch wenn das Selektionsformular nicht mehr offen ist und Sie können sie einfacher testen.

SQL-Ausdruck direkt

Sie können in der Angabe der Datenquelle für Formulare, Berichte oder Steuerelemente anstelle eines Abfragennamen auch den SQL-Ausdruck direkt schreiben. Diese Technik ist allerdings nur für einfache Selektionen geeignet. Für komplexere Aufgaben ist eine Abfrage übersichtlicher und außerdem kann sie auf mehreren Stellen eingesetzt und bei Bedarf nur einmal geändert werden

Verwenden Sie Alias-Namen (Anweisung AS)

Der übliche Weg zu Darstellung der Daten in einem Formular oder Bericht geht von den Tabellen über Abfragen. Wenn Sie einem Abfragefeld einen Alias-Namen vergeben und das Formular- oder Berichtsfeld mit diesem Namen verbinden, machen Sie Ihr Programm richtig wartungsfreundlich. Wenn in der Zukunft der Auftrag kommt, ein anderes Feld anzuzeigen, brauchen Sie nur mehr die Abfrage verändern. Der im Formular oder Bericht eingetragene Alias-Name bleibt gleich.

Tabellenerstellungsabfragen haben ihre Tücken

Wenn Sie gleichnamige Felder aus mehreren Tabellen in die neu erstellte Tabelle einfügen, bekommen sie automatisch den Namen `<Tabelle>_<Feld>`, damit sie eindeutig sind. Wenn Sie später eine Tabelle nicht mehr brauchen und sie aus der Abfrage entfernen, können die Probleme mit der Eindeutigkeit weg sein. Dann verschwindet auch der Tabellenname aus dem Feldnamen der erstellten Tabelle. Probleme können dann bei gebundenen Feldern in Formularen und Berichten auftreten, wenn dort als Datenquelle der Name `<Tabelle>_<Feld>` steht. Wenn Sie mit Alias-Namen arbeiten, passiert so was natürlich nicht.

Die Datentypen der Felder in der erstellten Tabelle sind nicht immer automatisch richtig definiert. Es kann vorkommen, dass Access aus einem nummerischen Ausdruck ein Textfeld bildet. Definieren Sie zur Sicherheit alle nummerischen Feldtypen explizit mit den Typumwandlungsfunktionen wie `CDbl()`, `CInt()`, usw.

Der Abfragegenerator ist nicht immer das Gelbe vom Ei

Für die Anfänger ist er natürlich hervorragend. Ohne die kleinste Ahnung über SQL-Syntax kann man praktisch alles (mit ein paar Mausklicken und -Ziehungen) machen. Es ist aber oft besser, den SQL-Ausdruck direkt zu bearbeiten und das ist im SQL-Fenster des Abfragengenerators praktisch unmöglich. Microsoft verwendet eine proportionale Schrift und der Text wird immer neu umformatiert.

Ja, generieren Sie in der ersten Phase ihre Abfrage mit diesem Generator, es geht wirklich schnell. Übertragen Sie aber dann den SQL-Ausdruck über die Ablage in ein Modul und befüllen Sie eine String-Variable mit jedem Feldausdruck in einer einzelnen Zeile. Den Inhalt dieser Variablen weisen Sie dann einer Abfrage zu. Der Aufwand lohnt: Sie können jede Zeile kommentieren. Sie können den Aufbau des SQL-Ausdrucks durch andere Parameter

steuern. Sie gewinnen eine gute Übersichtlichkeit. Beachten Sie nur, dass Sie jetzt mit dem Suchen im Kode einen bestimmten Feldausdruck in der gesamten Datenbank finden können.

Formulare

Präfix-Vorschläge

`frm` Standard-Formular

`sub` Unterformular

`fs` Hilfsformular für die Entwicklung, der beim Programmablauf nicht verwendet wird (z.B. für die Eingabe der Werte in eine Systemtabelle)

Sparen Sie an der Anzahl der Formulare

Wenn Sie die logische Hierarchie eines Programms (z. B. Auswahl -> Eingabe -> Berechnung -> Ausdruck) zu exakt mit mehreren nacheinander zu öffnenden Formularen abbilden, kann es für den Anwender lästig sein. Wenn er nämlich zwischen verschiedenen Programmteilen wechseln will, muss er eventuell mehrmals auf „***OK***“, bzw. „***Zurück***“ klicken.

Die Formulare können mehr komplex sein, besonders dann, wenn Sie mit einer höheren Bildschirmauflösung (mindestens 800x600) rechnen können. Damit der Anwender den Überblick nicht verliert, können Sie ihm dadurch helfen, dass Sie je nach der Eingabe einige Steuerelemente deaktivieren oder sogar ganz ausblenden lassen. Sie können auch z.B. Gruppen von Feldern mit Hintergrundfarben unterscheiden.

Vermeiden Sie wiederholte Anweisungen

Wenn Sie die gleichen Anweisungen an eine Gruppe von Steuerelementen anwenden wollen (z.B. einen ganzen Block von Textfeldern ausblenden), bemühen Sie sich die ganze Gruppe in einer Schleife zu bearbeiten, ohne für jedes Element die gleiche Anweisung extra zu kodieren. Sie sparen sich erstens die Tipparbeit, zweitens können Sie die Anzahl der Elemente in der Gruppe beliebig ändern. Sie können sogar eine allgemeine Funktion erstellen, die Sie in einem Modul speichern und in jedem beliebigen Formular aufrufen können.

Das einzige Problem dabei ist die Zugehörigkeit der Elemente zu einer Gruppe zu erkennen. Dafür gibt es mehrere Möglichkeiten:

- die Elemente gezielt benennen und den Teil des Namens auswerten.
- die Eigenschaft `ControlType` abfragen
- die Eigenschaft `Tag` setzen und im Kode abfragen

Formulare ohne Klassenmodul

Wenn es Ihnen gelingt, den gesamten Kode für ein Formular in der Form allgemeiner Funktionen zu implementieren, können Sie ab Access Version 97 ein Formular entwerfen, das kein Kode-Modul (Klassenmodul) besitzt. Ein solches Formular wird schneller geöffnet und verlangt weniger Arbeitsspeicher. Die gleiche Regel gilt auch für die Berichte.

Für die Formulare, die mehrmals geöffnet werden sollen (mehrere Instanzen des gleichen Formulars), ist aber ein Klassenmodul unentbehrlich.

Berichte

Präfix-Vorschläge

`rpt` Standard-Bericht

`rsub` Unterbericht

`rps` Bericht für die Entwicklung, den der Anwender nicht aufrufen kann (z.B. für die Dokumentation)

Ausdruck eines Berichts darf keine Daten verändern

Packen Sie nie eine Berichtsausgabe mit einer Funktion zusammen, die die Daten verändert, auch wenn die beiden Tätigkeiten logisch richtig zusammenhängen (z.B. ein Abschlussbericht und anschließendes Aufsummieren der Detailangaben). Sie können nie vermeiden, dass der Anwender den Ausdruck wiederholt und damit die Daten in einen unvorgesehenen Zustand versetzt.

Berichte sind ein mächtiges Werkzeug

Machen Sie sich mit Funktionalität der Gruppierung in den Berichten gut vertraut. Sie können manchmal einige Berechnungen im Bericht direkt realisieren, ohne extra Abfragen erstellen zu müssen.

Das gleiche gilt für die Ereignis-Prozeduren, die beim Formatieren jedes beliebigen Berichtsbereichs zum Einsatz kommen. Damit können Sie verschiedene Bedingungen testen und das Layout des Berichts während des Ausdrucks anpassen – z.B. den ganzen Bereich oder einzelne Steuerelemente weglassen oder anders formatieren.

Preview immer groß

Wenn Sie den Bericht nicht nur direkt drucken, sondern auch auf dem Bildschirm (Preview) anzeigen wollen, maximieren Sie das Fenster sofort beim Öffnen des Berichts. Sie müssen dann beim Schließen des Berichts die ursprüngliche Größe wiederherzustellen, damit das Formular, zu dem das Programm zurückkehrt nicht mehr maximiert angezeigt wird.

Makros

Die Makros haben in der Schule ihre Berechtigung, weil sie einfach zu erlernen sind, sprechende Namen haben und die wichtigsten Operationen mit Daten und Datenbankobjekten anschaulich präsentieren.

Für die professionelle Entwicklung gibt‘s einen anderen Rat: **die Makros so wenig wie möglich verwenden**. Ihr Hauptnachteil ist der, dass sie schwierig zu dokumentieren sind. Alles, was Sie mit Makros realisieren können, geschieht mit Programmkode viel besser und übersichtlicher. Sie sind nur dann sinnvoll, wenn Sie beim Entwickeln eine Aktion außerhalb des geplanten Programmablaufs wiederholt aufrufen möchten. Ein Makroaufruf ist schon komfortabler als das Starten einer Funktion mit der Eingabe im Direktfenster.

Module

Präfix-Vorschläge

`bas` projektspezifische Module

`mod` Module mit allgemein brauchbaren Funktionen

Maximale Funktionalität in Modulen

Alle Funktionen, die Sie in einem Modul realisieren, sind von allen Objekten der Datenbank sichtbar. Diejenige, die Sie für mehrere Formulare oder Berichte brauchen, gehören eindeutig hierher. Dann haben Sie den notwendigen Kode im Programm nur einmal und das ist für die zukünftigen Änderungen besonders günstig.

Suchen Sie allgemeine Lösungen

Bemühen Sie sich, die entwickelten Funktionen allgemein zu formulieren und alle projektspezifische Daten an sie als Parameter zu übergeben. Solche Funktionen speichern Sie in die Module mit Präfix „`mod`“. Sie können sie später in ein anderes Programm kopieren oder – noch besser – in eine Bibliotheksdatenbank auslagern, die Sie dann in jedes beliebige Programm einbinden können.

Die wirklich optimale Lösung ist, eine solche Bibliothek gemeinsam für alle Entwickler Ihrer Firma anzulegen und ständig zu erweitern. Es ist aber dann auch mit einem zusätzlichen Aufwand für die Wartung zu rechnen. Außerdem erfordert der Einsatz einer gemeinsamen Funktionssammlung eine bestimmte Disziplin – es dürfen z.B. nicht die Aufrufparameter einer Funktion geändert werden, weil sie dann eventuell mit älteren Programmen nicht mehr kompatibel ist, usw.

Verweise auf andere Objekte

Wie können Sie in einem allgemeinem Modul auf ein konkretes Formular oder Bericht, bzw. ihre Steuerelemente verweisen? Zwei Methoden sind möglich. Erstens können Sie mit den Eigenschaften `ActiveForm` oder `ActiveReport` arbeiten. Der Vorteil dieser Technik ist, dass Sie die Funktion direkt im Eigenschaftsfenster ohne Klassenmodul aufrufen können, der Nachteil ist aber der, dass Sie sie nicht debuggen können. Die bessere Methode ist, die Funktion in einem Klassenmodul aufzurufen und die Referenz auf das aktuelle Formular oder Bericht mittels der Eigenschaft `Me` als Aufrufparameter zu übergeben.

Globale Variablen

Die aus allen Programmteilen sichtbaren globalen Variablen sind gefährlich, wenn sie nicht richtig eingesetzt werden. Die Gefahr besteht darin, dass man sie überall nicht nur lesen, sondern auch verändern kann. Sie sind aber trotzdem zu empfehlen, wenn Sie klar definieren, wann die Werte gesetzt werden (z.B. unmittelbar nach dem Programmstart und dann nie mehr).

Die Sichtbarkeit einer globalen Variablen hat aber seine Grenzen. Sie können sie weder in einer Formulareingeschaft noch in einer Abfrage direkt verwenden. Die Abhilfe schafft eine global definierte Zugriffsfunktion, die den Wert der globalen Variablen ausliest und so für alle Objekte zugänglich macht.

Konstanten

Definieren Sie alle im Programm verwendeten festen Werte als Konstanten. Schreiben Sie ihre Namen groß, damit Sie gut erkennbar sind.

Sie können die Konstanten auch in einer speziellen Tabelle speichern. Diese Lösung hat den Vorteil, dass der Wert der Konstante auch ohne den Eingriff in den Source-Kode modifiziert werden kann.

Globales Modul

Es ist empfehlenswert, alle globalen Definitionen – Variablen, Konstanten und Zugriffsfunktionen in einem Modul gemeinsam zu deklarieren. Dieses Modul können Sie trotz aller Namenskonventionen nur „`g`“ benennen. Dann können Sie überall im Programmkode die globale Variable mit „`g.`“ bezeichnen, wodurch sie besonders auffällt.

Texte auslagern

Schreiben Sie keine Texte (Abfragen, Fehlermeldungen) direkt in den Kode sondern speichern Sie sie in einer Tabelle. Verwenden Sie sprechende Abkürzungen als Schlüssel für das Auffinden des gewünschten Textes. So wird jeder Text nur einmal im Programm enthalten sein, unabhängig davon, wie oft er verwendet wird und kann in der Tabelle direkt verändert werden. Noch ein Vorteil ist wichtig: wenn einmal ihr Programm mehrsprachig sein sollte, tauschen Sie die Tabelle mit Texten gegen eine andere und die Umstellung ist fertig.

Tipps zum Kode-Entwurf

Funktionsanalyse

Projektdefinitionen in Tabellen speichern

Verwenden Sie die Tabellen nicht nur zur Speicherung der Anwenderdaten, sondern auch als Unterstützung für die Steuerung des Programmablaufs. So können Sie zum Beispiel die Combobox zur Auswahl eines Berichts an eine Systemtabelle binden, die für jeden Bericht den Namen der Berichtsvorlage, der Datenquelle, die Filterbedingung, bzw. weitere Parameter enthält, die beim Aufruf eines Berichts ausgewertet werden.

Die Einträge in den Projektdefinitionstabellen können auch mit einem Gültigkeitsintervall (z.B. Jahr von-bis) versehen werden und damit auch die Struktur und Layout des Programms automatisch je nach dem aktuellen Jahr ändern.

Prozedur oder Funktion?

Eine Funktion ist meist brauchbarer als eine Prozedur. Sie kann außer der Ausführung eines Programmteils auch einen Rückgabewert liefern. Er muss nicht das eigentliche Ergebnis der Arbeit der Funktion sein, sondern z.B. nur eine logische Variable, die mit dem Wert `True` den fehlerfreien Ablauf der Funktion darstellt. Sie können den Rückgabewert abfragen und das Programm dementsprechend reagieren lassen.

Eine Funktion kann im Gegensatz zur Prozedur auch direkt im Eigenschaftsfenster eines Formulars, Berichts oder Steuerelements oder in einem Makro aufgerufen werden. So kann ein Formular oder Bericht ohne Klassenmodul erstellt werden – siehe oben den Absatz „Formulare ohne Klassenmodul“.

Dekomposition, Modularisierung

Ein guter Software-Entwickler muss im bestimmten Sinne faul sein. Er muss bemüht sei, sich Tipparbeit zu sparen. Immer, wenn Sie im Programm ähnliche Aktionen durchzuführen haben, packen Sie den entsprechenden Kode-Teil in eine Funktion. Die unterschiedlichen Werte die für die einzelnen Aufrufe notwendig sind, übergeben Sie beim Funktionsaufruf als Parameter.

Auch wenn ein bestimmter Teil des Programms nur auf einer Stelle zum Einsatz kommt, dafür aber sehr lang ist, lohnt es, ihn in mehrere kleinere Funktionen zu teilen.

Die optimale Funktionsgröße beträgt nur ein paar Bildschirmseiten, sonst behalten Sie schwierig den Überblick und können z.B. die verschachtelten If-Strukturen nur mühsam verfolgen.

Eine effiziente, wenig bekannte Programmiertechnik ist die Rekursion. Sie ermöglicht automatisch verschachtelte Funktionsaufrufe, wie z.B. beim Durchsuchen einer Verzeichnisstruktur, ist aber etwas schwieriger zu verstehen.

Kontextfrei programmieren

Jede Funktion soll einen sprechenden Namen haben und genau nur die Aktion durchführen, die von ihr entsprechend ihrem Namen erwartet wird. Alle Nebenaktionen, egal wie logisch sie mit der Funktion zusammenhängen, sollen außerhalb kodiert werden. Ein gutes, übersichtliches Programm muss so gut lesbar sein wie ein Buch.

Die Schnittstellen – also die Definition der Aufrufparameter – sind das A und O des erfolgreichen Funktionsentwurfs. Eine Funktion soll globalen Variablen nur so wenig wie möglich Informationen entnehmen, die meisten sollen als Parameter übergeben werden. Die Parameter dürfen auch optional sein, das bedeutet, dass sie nicht angegeben werden müssen. In der Deklaration kann ein Defaultwert definiert werden, der zum Einsatz kommt, wenn der jeweilige Parameter weggelassen wird..

Zuverlässigkeit jedes kleinsten Programmteils

Die Kette ist so stark, wie ihr schwächstes Glied. Jede Funktion muss für die Außenwelt eine kompakte, zuverlässige Einheit bilden. Sie muss erlauben, mit jedem beliebigen Parameter aufgerufen zu werden; die Parameterwerte werden intern überprüft und eventuell klare und eindeutige Fehlermeldungen ausgegeben. Sie dürfen sich nie darauf verlassen, dass die Aufrufparameter immer in Ordnung sind. Das gilt besonders für die Überprüfung der manuell eingegebenen Werte in einem Formular. Auch ein erfahrener Anwender kann sich vertippen, und wenn ein ungültiger Wert die Ursache für einen Programmabsturz ist, ist es für Ihr Programm ein schlechtes Zeugnis.

Veränderungen der Daten in einer Tabelle

Vor dem Speichern eines neuen oder geänderten Datensatzes müssen Sie alle Werte überprüfen. Es geht dabei nicht nur um Werte in einzelnen Datenfeldern eines Datensatzes an und für sich, sondern vor allem um die Gefahr der Verletzung einiger Regeln, die für die ganze Tabelle, bzw. Verknüpfung von mehreren Tabellen gelten (Integritätsregeln für die Erhaltung der Datenkonsistenz). Einige dieser Regeln kann Access selbst überwachen, die Standard-Fehlermeldungen sind aber für die Anwender nicht immer verständlich genug. Es ist besser, die möglichen Fehler im Kode abzufangen und mit eigenen Erklärungen zur Kenntnis bringen.

Saubere Unterbrechung, bzw. Schließung des Programms

Wenn wegen eines Bedienungs- oder Programmfehlers eine Teilfunktion des Programms unterbrochen wird, muss das immer „sauber“ gemacht werden. Es ist erforderlich, eine verständlich Fehlermeldung auszugeben und alle offenen Objekte zu schließen, damit sie für andere Programmteile nicht gesperrt bleiben.

Interne Struktur einer Funktion

Alle lokalen Variablen deklarieren

Nehmen Sie von der scheinbaren Vereinfachung Abstand, dass Sie die Variablen nicht explizit deklarieren müssen, bevor Sie sie verwenden wollen. Es kann nur solange funktionieren, bis Sie sich einmal in einem Variablennamen vertippen und den Fehler dann stundenlang suchen. Deswegen ist `Option Explicit` am Anfang jedes Moduls (kann als eine Option für alle Module eingestellt werden) ein absolutes Muss.

Postfixe bei Variablennamen statt As

Gegen die Zeitersparnis bei der Variablendeklaration, indem Sie die Postfixe (z.B. `$` statt „`As String`“ für eine Stringvariable) verwenden, ist grundsätzlich nichts einzuwenden. Sie können sich auch zu lange sprechende Namen bei den lokalen Variablen sparen, weil es in einer Funktion nicht so schwierig ist, den Überblick zu bewahren.

Übersichtlichkeit des Kodes

`GoTo`-frei programmieren ist ein Schlagwort der letzten Jahre. Es ist natürlich nicht der Befehl selbst, sondern seine ungeeignete Verwendung am eventuellen Misserfolg schuld. Beim Entwickeln des regulären Funktionsablaufs sollten Sie auf ihn wirklich verzichten. Sinnvoll ist er dagegen beim Lösen von Fehlerzuständen oder beim vorzeitigen Beenden einer Funktion.

Vergessen Sie nicht auf das Einrücken bei den `If-Else` oder `Case`-Befehlen. Wenn die ausgewerteten Bedingungen komplexer sind, ist der Befehl `Case` dem `If`, bzw. `ElseIf` vorzuziehen, weil er übersichtlicher ist.

Interne Sprache

Die internen Funktions-, Objekt- oder Variablen benenne ich ganz gerne englisch. Die Anwender sehen sie sowieso nicht und man spart sich Tipparbeit, weil die englischen Ausdrücke meistens kürzer sind. Sie können so auch auf Umlaute und "ß" verzichten, die manchmal zu Problemen führen.

Testen - das Allerwichtigste

Wenn Sie ein Programm komplett erstellt haben, sind Sie bei weitem noch nicht fertig. Die schwierigste Phase – das Testen – beginnt.

Der beste Tester sind nicht Sie selbst, sondern z.B. der Arbeitskollege, der Ihr Programm nicht kennt. Wenn er dabei auch besondere Lust verspürt, ihr Werk zum Absturz zu bringen, ist er die richtige Person für Sie. Wenn Sie selbst testen, geben Sie immer unbewusst sinnvolle Daten ein, um zu sehen, wie gut das Programm ist und haben eine kleinere Chance, die Schwächen zu finden.

Wenn Sie eine Funktion ändern, die Daten in eine Tabelle schreibt, können Sie sie am schnellsten so testen, wenn Sie den Inhalt der Tabelle vor und nach der Änderung vergleichen. Dafür können Sie auf meiner Homepage ein Freeware-Tool finden.

Jeder Fehler, der „zu Hause“ entdeckt und nicht unter die Anwender verteilt wird, ist ein Gewinn. Wenn Ihre Kunden im gleichen Haus sitzen, ist es noch nicht so schlimm. Nachdem aber Ihr Produkt auf mehreren, nicht einfach erreichbaren Arbeitsplätzen läuft oder sogar am freien Markt verkauft wird, ist das Testen besonders ernst zu nehmen.

Ihre Anwender können Ihnen beim Testen sehr behilflich sein, wenn es Ihnen gelingt, Sie zu einer exakten Zusammenarbeit zu überreden. Bitten Sie sie, dass sie sich jede Fehlemeldung abschreiben und dazu auch notieren, unter welchen Umständen (Eingabewerten, Funktionsaufrufen) es zu dem Fehler gekommen ist. Oft bekommen Sie nämlich z.B. folgende Rückmeldung: *„Das Programm stürzt immer ab. Er schreibt etwas, aber ich weiß nicht mehr was ... Schauen Sie sich das, bitte, an.“*

Wartung und Weiterentwicklung

Die wahre Qualität eines Programms bewährt sich erst nach Jahren. Einige Lösungen „altern“ bevor sie richtig zum Einsatz kommen, einige „leben“ viele Jahre lang. Es hängt natürlich nicht nur von Ihnen als Entwickler, sondern auch z.B. von der sich ändernden Firmenstrategie und verschiedenen anderen Bedingungen ab.

Wie schon oben mehrmals betont wurde, ist die Dokumentation die wichtigste Voraussetzung für eine effiziente Wartung Ihres Werkes. Effizient bedeutet, dass Sie nicht viel mehr Zeit für das Einarbeiten in das Programm brauchen, als die wirkliche Realisierung der notwendigen Änderungen oder Erweiterungen erfordert.

Legen Sie auch einen Wert auf den dauerhaften Kontakt zu den Anwendern. Die offizielle Übergabe des Programms an den Auftraggeber ist die erste Voraussetzung für den erfolgreichen Abschluss Ihrer Arbeit, reicht aber selbst nicht. Man braucht unbedingt Rückmeldungen von Menschen, die mit dem Programm täglich arbeiten, um es laufend verbessern zu können. Notieren Sie sich jeden kleinsten Verbesserungsvorschlag (in der o.g. „ToDo“ Datei) und bauen Sie sie dann alle ein, wenn Sie bei einer größeren Änderung eine neue Version erstellen.

Die Menschen kommen und gehen und die neuen Anwender wissen manchmal nach Jahren nicht mehr, dass Sie der Autor des Programms sind und dass sie von Ihnen unterstützt werden können. Wenn Sie längere Zeit von der anderen Seite nichts hören, fragen Sie einfach selbst, ob alles in Ordnung ist, damit Sie die Kontakte wieder auffrischen.

Schlusswort

Sie sind in der schönen Welt der Softwareentwicklung nicht allein. Erfinden Sie das Rad nicht neu, fragen Sie diejenigen, die mehr Erfahrungen haben. Ein guter Rat aus der Praxis ist effizienter als stundenlanges Suchen im Handbuch oder in der Hilfe. Viele Informationen finden Sie heute im Internet auf den Seiten professioneller Entwickler, in verschiedenen Foren oder Newsgroups. In den **PCNEWS** sind in den letzten Jahren auch viele Bücher zu Microsoft-Office rezensiert worden.

Am schnellsten lernen Sie in einem guten Kollektiv, wo es sowohl Anfänger als Fortgeschrittene gibt und auf alle mehr als genug an interessanter Arbeit wartet. Kommen Sie einfach zu uns.

Unsere Firma heißt Metropolitan und ist eine Tochtergesellschaft der Wiener Städtischen Versicherung. Schauen Sie entweder auf http://www.metropolitan.at/ oder wenden Sie sich an mich persönlich unter kstipek@netway.at.

Für Ihre zukünftige Karriere wünsche ich Ihnen viel Erfolg.