Kapitel 7

ActiveX-Komponenten: Was ist was?

Ursprünglich hatte ich vor, den Inhalt dieses Kapitels noch in Kapitel 6, »Das Leben und die Lebensdauer einer ActiveX-Komponente«, unterzubringen. Doch wieder einmal bin ich der Meinung, daß sich die Bedeutung eines Kapitels nicht in Worten messen läßt.

Angesichts des »bunten Zoos« an verfügbaren ActiveX-Komponenten-Typen stellt sich erneut die Frage: Welchen Typ sollten Sie verwenden? Erfreulicherweise sind Sie ja nun Experte in Sachen ActiveX und COM-Technologie. Daher dürfte es Ihnen nicht schwerfallen, die Vor- und Nachteile bezüglich der Wahl des für Ihre Bedürfnisse richtigen ActiveX-Typs abzuwägen.

Als Außenstehender möchte ich aber noch eines klarstellen. Weder ich noch sonst irgendein Buchautor kann Ihnen die Lösung Ihrer konkreten Programmierprobleme vorschreiben. Ich kann nur hoffen, Ihnen dabei zu helfen, die richtigen Fragen zu stellen und Ihnen das für ihre Beantwortung notwendige Wissen zu vermitteln.

7.1 ActiveX-Controls und -Dokumente

Bisher haben wir uns auf drei Typen von ActiveX-Komponenten konzentriert: Klassen innerhalb einer Anwendung, DLL-Server und EXE-Server. Ignorieren wir für einen Moment die Klassen innerhalb einer Anwendung, da sie den Kernbaustein für alle anderen Typen von ActiveX-Komponenten darstellen. Der Hauptunterschied zwischen DLL- und EXE-Servern ergibt sich daraus, ob sie im gleichen Prozeßraum oder Out-of-Process laufen. Auf die weiteren Unterschiede werden wir später noch eingehen.

Oft werden diese Komponenten als ActiveX-*Code-Komponenten* bezeichnet, da die von diesen Komponenten offengelegten Objekte von Anwendungen häufig als Objekt- oder Funktionsbibliotheken verwendet werden. In unserer Darlehensanwendung war beispielsweise jeder Darlehenstyp eine Komponente, auf die vom Loan-Programm zugegriffen wurde – vom Code, den Sie geschrieben haben. Die Komponenten hatten keine visuelle Benutzeroberfläche und konnten nicht direkt vom Anwender manipuliert werden.

Nun, Code-Komponenten können durchaus eine Benutzeroberfläche haben. Sie können ihre eigenen Formulare oder Messageboxen anzeigen. In der Praxis ist dies jedoch nicht üblich, abgesehen von Anwendungen, die ihr eigenes Objektmodell zur Programmierung durch andere Anwendungen offenlegen. Es gibt eine Reihe von Gründen dafür:

- Einer der Hauptanlässe, eine ActiveX-Code-Komponente zu schreiben, ist die Wiederverwendbarkeit von Code. In den meisten Fällen werden solche Komponenten so allgemein wie möglich gehalten. Es wird vermieden, anwendungsspezifische Features einzubauen. Es gibt viele Situationen, in denen eine Benutzeroberfläche dieser Richtlinie zuwiderlaufen würde.

- Formulare, die von einer ActiveX-Code-Komponente geöffnet werden, sind nicht in der Forms-Sammlung der aufrufenden Anwendung enthalten. Bei einem EXE-Server existieren Sie noch nicht einmal im Task der aufrufenden Anwendung. Dies kann die Benutzeroberfläche der aufrufenden Anwendung mitunter durcheinanderbringen.
- Unter Visual Basic 4.0 konnten Formulare in DLL-Servern nur eingeschränkt verwendet werden. Diese Probleme scheinen zwar in den Versionen 5 und 6 beseitigt zu sein, doch ist nicht hundertprozentig sicher, daß nun funktionierende Techniken auch in Zukunft einwandfrei funktionieren werden.

Es ist gar nicht vorgesehen, daß ActiveX-Code-Komponenten eine umfangreiche Benutzeroberfläche haben sollen. Dennoch gibt es erwartungsgemäß Technologien, die unter dem Begriff *ActiveX* einzuordnen sind und sehr wohl Interaktion mit dem Anwender unterstützen.

Die erste und bekannteste dieser Technologien sind ActiveX-Controls (früher als OLE-Controls bezeichnet).

7.1.1 ActiveX-Controls

In Kapitel 2, »ActiveX – eine historische (aber auch technische) Betrachtung«, haben Sie ein wenig darüber gelesen, wie eine Kalkulationstabelle oder ein anderes Objekt in ein Textverarbeitungsdokument eingefügt wird. Eine Excel-Tabelle kann vor Ort bearbeitet werden (In-Place-Bearbeitung) – ein Konzept, auf das wir in Kapitel 2 näher eingegangen sind. In-Place-Bearbeitung bedeutet, daß man beispielsweise auf ein Excel-Tabellen-Objekt in einem Word-Dokument doppelklikken kann und Excel das Fenster übernimmt, damit die Tabelle direkt bearbeitet werden kann. Die Menüs von Excel ersetzen die Word-Menüs, solange die Tabelle aktiv ist und bearbeitet wird. Diese Form der In-Place-Bearbeitung bringt ein großes Maß an Overhead mit sich, was zum Teil der hohen Komplexität der Excel-Tabellen zuzuschreiben ist, zum Teil aber auch daran liegt, daß Excel in einem eigenen Prozeßraum läuft.

ActiveX-Controls stellen eine Sonderform von In-Place bearbeitbaren Objekten dar. Sie laufen im gleichen Prozeß, und sie werden aktiviert, sobald sie ausgewählt werden (den Fokus erhalten). Sie verfügen nicht nur über Methoden und Eigenschaften, sondern können darüber hinaus auch Ereignisse auslösen. Die Visual-Basic-Umgebung integriert ActiveX-Controls nahtlos.

Wie andere ActiveX-Komponenten bauen auch ActiveX-Controls auf COM-Objekten auf. Der Code, der diese COM-Objekte implementiert, ist in DLLs enthalten, die in der Regel die Dateierweiterung `.OCX` tragen. Ich möchte es nochmal betonen: Eine OCX-Datei ist eine DLL, nur mit einer anderen Dateierweiterung!

ActiveX-Controls stellen für Visual-Basic-Programmierer gegenwärtig die wichtigste Form von wiederverwendbaren Komponenten dar. Interessanterweise wer-

den jedoch, seitdem auch Code-Komponenten Ereignisse auslösen können, mehr und mehr Aufgaben, die früher nur mit ActiveX-Controls gelöst werden konnten, nun von Code-Komponenten übernommen.

7.1.2 ActiveX-Dokumente

Visual Basic unterstützt auch das Erstellen von Objekten, die *ActiveX-Dokumente* genannt werden und früher unter dem Namen *DocObjects* bekannt waren.

Zum Verständnis von ActiveX-Dokumenten betrachten wir zunächst einen vertrauteren Dokumenttyp – ein Textverarbeitungsdokument. Microsoft Word verwendet Dokumente mit der Dateierweiterung `.DOC`. Sie wissen, daß Sie mit DOC-Dateien arbeiten können, wenn Sie Word starten und darin das gewünschte Dokument öffnen. Die Daten sind in der DOC-Datei enthalten und der Code zum Ansehen und Bearbeiten des Dokuments im Word-Programm.

Sie wissen ebenfalls, daß Sie eine DOC-Datei auch bearbeiten können, indem Sie sie beispielsweise im Windows-Explorer öffnen, etwa per Doppelklick. Dabei wird Word automatisch gestartet und angewiesen, die Datei zu laden. Der Explorer ist dazu in der Lage, weil ein Eintrag in der System-Registrierung dem System mitteilt, daß Dateien mit der Dateierweiterung DOC von Microsoft Word geöffnet werden sollen – vorausgesetzt natürlich, Word ist auf dem betreffenden System installiert. Falls letzteres nicht zutrifft, ist die Dateierweiterung DOC vielleicht mit einem anderen Textverarbeitungsprogramm verknüpft, das das DOC-Dateiformat lesen kann.

Erweitern wir diese Idee zu einem neuen Dokumenten-Typ. Dieses Dokument nennen wir einmal *MeinDok*, das einen eindeutigen GUID hat.

Ein Visual-Basic-Programm namens `MeinDokServer.EXE` kann diesen Dokumententyp lesen und bearbeiten. Die Daten von *MeinDok* werden in einer Datei mit der Dateierweiterung `.VBD` gespeichert. Diese Datei enthält eine GUID, die den Typ des Objekts identifiziert. So kann die VBD-Dateierweiterung für jeden beliebigen Objekttyp verwendet werden, und die GUID ermöglicht die Ermittlung des passenden Objekt-Servers.

Wenn nun ein Container, der ActiveX-Dokumente unterstützt, das MeinDok-Dokument öffnen will, liest er die GUID aus der VBD-Datei aus und lädt den entsprechenden Server. Der Server kann das Dokument innerhalb der Container-Anwendung darstellen, ähnlich wie ein Visual-Basic-Formular ein ActiveX-Control darstellen kann. Der Server kann gegebenenfalls weitere Formulare anzeigen.

ActiveX-Dokumente werden zur Zeit als Weg zur Verbreitung von »smarten« Dokumenten über das Internet angepriesen. Der Internet Explorer kann sowohl den Server herunterladen (soweit dieser auf dem Client-System noch nicht installiert ist) als auch die VBD-Datei, die im Browser-Fenster dargestellt wird.

Der vielleicht interessanteste Aspekt ist, daß ActiveX-Dokumente einen sehr einfachen Weg darstellen, Visual-Basic-Anwendungen über ein Netzwerk zu vertei-

len. Denken Sie einmal darüber nach: Sie können ein Visual-Basic-Formular blitzschnell in ein ActiveX-Dokument konvertieren, das im Browser genauso aussieht. Dies könnte den Bedarf an HTML, Internet-Scripting-Sprachen (wie JavaScript oder VBScript) und Java-Applets auf einen Schlag erübrigen, zumindest auf Windows-Plattformen oder anderen, die ActiveX-Dokumente unterstützen könnten.

Das war die Idee. In der ersten Auflage dieses Buches habe ich mich zwar mit dieser Technologie beschäftigt, mich aber gefragt, ob sie Bestand haben würde. ActiveX-Dokumente leiden an einer Reihe von Beschränkungen. Sie laufen nur im Internet Explorer. Sie scheinen auch nicht sehr weit in der Programmierer-Gemeinschaft verbreitet zu sein, so daß Dokumentation und Support für diese Technologie rar bleiben.

Ich habe die Leser auch dazu eingeladen, mich wissen zu lassen, wenn ihnen eine gute Verwendung für ActiveX-Dokumente einfallen würde. Ich habe zwar ein paar Mitteilungen von Leuten erhalten, die ActiveX-Dokumente erfolgreich in ihren Unternehmens-Intranets einsetzen, doch insgesamt ist das Echo sehr mager gewesen.

Ich hatte mich entschieden, die Behandlung der ActiveX-Dokumenten-Technologie völlig aus dem Buch zu streichen. Doch wenn ich das getan hätte, hätte ich den mageren Bestand an Information zu diesem Thema nur noch weiter abgespeckt. So habe ich die Informationen im Buch belassen, verbinde sie aber mit der Warnung, sehr sorgfältig darüber nachzudenken, bevor Sie ActiveX-Dokumente in Ihrem Unternehmen zum Einsatz bringen.

7.2 Internet-Komponenten

Visual Basic 6.0 führte zwei neue Typen von Internet-bezogenen Projekten ein: IIS-Anwendungsprojekte und DHTML-Projekte (Dynamic HTML).

Eine IIS-Anwendung (bzw. Web-Klasse) ist eine Komponentenart, die auf einem Internet-Server läuft, speziell auf einem Internet-Server, auf dem Microsofts Internet Information Server mit Active Server Pages-Unterstützung läuft. Dieser Komponententyp ermöglicht es, Programme zu schreiben, die auf HTTP-Requests über das Internet antworten und Seiteninhalte dynamisch erzeugen können. Sie werden diese Art der Technologie bereits im Web gesehen haben – wenn Sie ein Formular ausfüllen, um Informationen abzusenden oder um eine Bestellung aufzugeben, hängen die zurückkommenden Daten von den eingegebenen Informationen ab. Dazu wickelt der Server einige komplexere Operationen im Hintergrund ab (Ausspionieren Ihrer Lesegewohnheiten, Abziehen Ihres Vermögens von Ihrem Girokonto usw.). Es gibt eine ganze Reihe von Technologien, die eine derartige serverseitige Programmierung ermöglichen. Dazu gehören CGI mit Perl-Programmierung und Active Server Pages. Der IIS-Anwendungs-Designer in Visual Basic bietet nun einen weiteren Ansatz.

Ein DHTML-Projekt ist ein Komponententyp, der eine HTML-Datei liest und es ermöglicht, mit dem Inhalt wie mit Objekten umzugehen, wobei mit Visual Basic als zugrundeliegender Sprache das Erscheinungsbild dieser Objekte und die Interaktion mit ihnen kontrolliert werden. Der Vorteil von DHTML ist, daß Sie komplexere Benutzeroberflächen erstellen können, die in einem Browser laufen, ohne ständige Aufmerksamkeit eines Servers zu erfordern.

Beim Bearbeiten dieser Ausgabe des Buches stellte sich mir die Frage, ob ich diese Komponententypen in das Buch aufnehmen sollte. Um diese Frage beantworten zu können, stand ich vor der unlösbaren Aufgabe, zu erraten, welche dieser Technologien wirklich von Bedeutung sei.

Sehen Sie, wie jeder Programmierer werde auch ich mit steigenden Mengen an Marketingmaterial in der Verkleidung solider Information überschwemmt. Ich habe gelernt, daß es sehr schwierig ist, zu entscheiden, welche Technologien wirklich von Bedeutung, und welche nur »coole« Ideen auf der Suche nach einem Markt sind. Ich bin ein wenig stolz auf das Eintreffen meiner Vorhersagen. In der ersten Ausgabe dieses Buchs habe ich vermutet, daß ActiveX-Controls für die Entwicklung von Anwendungen wichtiger werden würden als für die Entwicklung von Web-Sites – es sieht so aus, als ob sich das bewahrheitet hätte. Ich habe vermutet, daß ActiveX-Code-Komponenten ebenfalls größere Bedeutung erlangen würden – auch dies ist eingetroffen. Ich hatte die Bedeutung von ActiveX-Dokumenten in Frage gestellt – bisher habe ich nichts gesehen, das meine Zweifel ausgeräumt hätte. Ich sage dies alles nicht, um zu prahlen, sondern um meinen Entscheidungen, welche Technologien ich behandle, ein wenig Glaubwürdigkeit zu verleihen.

Weiter hinten in diesem Buch werden Sie das finden, was ich als hoffentlich gute Einführung in eine gemäßigt-anspruchsvolle Beschreibung von IIS-Anwendungen ansehe. Ich halte es für eine äußerst wichtige Technologie, die (wie ich hoffe), Active Server Pages dereinst völlig ersetzen wird (ich werde Ihnen später noch sagen, warum ...).

Über DHTML-Projekte werden Sie dagegen nur sehr wenige Informationen finden. Warum? Ich halte DHTML derzeit noch für eine Technologie, die etwas übertrieben angepriesen wird, bevor Sie wirklich ausgereift ist. Sie ist zur Zeit noch viel zu sehr Browser-abhängig. Das betrifft nicht nur konkurrierende Browser (ich nehme an, daß es solche noch geben wird, während Sie diese Zeilen lesen), sondern auch die einzelnen Versionen des Internet Explorers. Ich will nicht behaupten, daß die Idee dieser Technologie vollkommen daneben wäre – sie könnte auch möglicherweise den ActiveX-Dokumenten ihr endgültiges Grab schaufeln. DHTML könnte sich auch als die Top-Technologie für Unternehmens-Intranets erweisen, aber das ist noch längst nicht ausgemacht. Egal wie – irgendwie paßt das Thema auch nicht in dieses Buch hinein. DHTML-Programmierung unterscheidet sich derart stark von der übrigen Komponenten-Programmierung, daß sie ein eigenes Buch erfordern würde. Ich lade Sie herzlich dazu ein, dieses Buch zu schreiben ...

7.3 ActiveX: Vor- und Nachteile

Schauen wir uns einmal kurz die Vor- und Nachteile an, die bei der Auswahl eines ActiveX-Komponententyps eine Rolle spielen.

7.3.1 Klassen

Die Vorteile sind eine hervorragende Performance und keine Registrierungsprobleme.

Die Nachteile sind zweischneidig:

- Der Code wird in die Anwendung kompiliert. Wenn Sie eine Klasse in vielen Anwendungen einsetzen, kann diese Duplizität Verschwendung sein.
- Wenn Sie einen Fehler in einem Klassen-Modul entdecken, müssen Sie jede Anwendung rekompilieren, die diese Klasse verwendet, um das Problem zu beseitigen. Um die Fehlerbereinigung auszuliefern, müssen Sie alle ausführbaren Dateien und Komponenten ausliefern, die die Klasse verwenden.

Unterm Strich heißt das, daß Klassen ideal sind für Objekte, die spezifisch für eine Anwendung sind und nicht weithin wiederverwendet werden sollen.

7.3.2 ActiveX-DLLs (In-Process-Code-Komponenten)

Dies sind die Vorteile von DLLs:

- Code kann auf einfache Weise von vielen Anwendungen gemeinsam genutzt werden.
- DLLs bieten eine hervorragende Performance wegen der In-Process-Natur der Komponente.
- Die Bereinigung eines Fehlers in einer DLL-Implementierung erfordert nur die Auslieferung der aktualisierten DLL. Alle Anwendungen, die diese DLL verwenden, profitieren automatisch von der Fehlerbereinigung.
- DLLs können von jedem OLE-Automation-Client verwendet werden, einschließlich aller VBA-Anwendungen (wie Microsoft Office) und anderen Windows-Entwicklungssprachen.

Dies sind die Nachteile:

- Ist eine aktualisierte DLL inkompatibel zu ihrem Vorläufer, kann jede Anwendung zerstört werden, die diese DLL verwendet.
- Es können keine neuen Ausführungs-Threads angelegt werden.
- Die Komplexität der Einrichtung einer Anwendung wird erhöht.
- Sie erfordert Registrierung, Versionskontrolle und Komponenten-Verifizierung für eine sichere Distribution.

Unterm Strich bedeutet das, daß DLLs ideal für die Implementierung von Standard-Objekten geeignet sind, die wiederverwendet oder von mehreren Anwendungen genutzt werden sollen. Sie eignen sich auch hervorragend zur Definition von Interfaces, die in anderen Objekten implementiert werden sollen. Sie eignen sich bevorzugt für sehr schnelle Objekte, die keine Benutzeroberfläche benötigen.

7.3.3 ActiveX-EXE-Server (Out-of-Process)

Die Vorteile lauten:

- Objekte können in ihrem eigenen Thread ausgeführt werden.
- Objekte können sowohl von Client-Anwendungen angelegt und verwendet werden, wie auch der Server als eigenständige Anwendung laufen kann.

Dies sind die Nachteile:

- Die Performance ist deutlich schlechter als bei ActiveX-DLLs oder Klassen.
- Die Notwendigkeit eines separaten Tasks bedeutet einen höheren System-Overhead.
- Die Komplexität der Einrichtung einer Anwendung ist höher.
- Erforderlich sind Registrierung, Versionskontrolle und Komponentenverifizierung für eine sichere Distribution.

Unterm Strich bedeutet dies, daß EXE-Server sich besonders zum Offenlegen eines Anwendungsmodells für andere Anwendungen eignen. Sie sind ebenfalls sehr nützlich zur Implementierung von Objekten, die im Hintergrund asynchron zur Hauptanwendung in eigenen Threads laufen können.

7.3.4 ActiveX-Controls

ActiveX-Controls bieten viele Vorteile:

- Gute Performance. ActiveX-Controls laufen immer In-Process. Jedoch bedeuten sie etwas zusätzlichen Overhead, der bei einem ActiveX-DLL-Server nicht anfällt. Darüber werden wir in Teil II, »ActiveX-Controls«, näher eingehen.
- Controls sind zu vielen Containern kompatibel, einschließlich Microsoft Office-Anwendungen und Internet-Browser.
- Controls fügen sich nahtlos in die Visual-Basic-Umgebung ein.
- Eigenschaften-Seiten bieten eine Benutzeroberfläche zur Design-Zeit an, wie auch als Laufzeit-Oberfläche in Visual Basic.
- ActiveX-Controls können in den meisten Containern Werte von Eigenschaften persistent ablegen.

Dies sind die Nachteile:

- Controls sind spürbar schneller als ActiveX-EXE-Server, jedoch wegen des zusätzlichen Overheads ein wenig langsamer als ActiveX-DLL-Server.
- Das Erstellen von qualitativ hochwertigen ActiveX-Controls ist eine komplexe Angelegenheit.
- Controls erhöhen die Komplexität bei der Einrichtung von Anwendungen.
- Erforderlich sind Registrierung, Versionskontrolle und Komponentenverifizierung für eine sichere Distribution.

Unterm Strich bedeutet das, daß Controls bestens geeignet sind für die Implementierung von wiederverwendbaren Objekten, die über eine Benutzeroberfläche verfügen. Sie eignen sich in vielen Fällen zur Verbesserung der Modularität einer Anwendung.

7.3.5 ActiveX-Dokumente (DocObjects)

Dies sind die Vorteile von ActiveX-Dokumenten:

- ActiveX-Dokumente verbinden die Daten in einem Dokument mit einem Benutzeroberflächen-Objekt. Dies erlaubt die Verteilung von beliebig komplexen Daten über das Internet und über Intranets.
- ActiveX-Dokumente können einen effizienten Weg zur Verteilung von Software über das Internet darstellen. Sie offerieren eine potentielle Alternative zu VBScript und Java bei mancher Anwendung.
- ActiveX-Dokumente erlauben die einfache Konvertierung von eigenständigen Visual-Basic-Anwendungen in Anwendungen, die über ein Netzwerk verteilt laufen können.

Dies sind die zwei Nachteile:

- Die Fähigkeiten der verschiedenen Container sind unterschiedlich.
- Die Technologie hat keine weite Verbreitung und Akzeptanz erreicht. Daher sind Dokumentations- und Informationsquellen nur in begrenztem Maße verfügbar. Es stellt sich die Frage der langfristigen Unterstützung dieser Technologie.

Unterm Strich bedeutet das, daß sich ActiveX-Dokumente besonders gut für Anwendungen eignen, die über das Internet oder Intranets laufen sollen.

7.3.6 IIS-Anwendungen

Es ist nicht ganz fair, IIS-Anwendungen mit anderen Komponententypen zu vergleichen. Es wäre angemessener, diese Technologie mit Active Server Pages zu vergleichen.

Dies sind die Vorteile von IIS-Anwendungen gegenüber Active Server Pages:

- Sie verwenden die mächtigere Visual-Basic-Sprache anstelle von VBScript.
- Sie verwenden kompilierte Visual-Basic-Komponenten anstelle von interpretierten Scripten, was sich in höherer Performance auswirken sollte.
- Sie ermöglichen eine klare Trennung zwischen serverseitigem und clientseitigem Code, was ein klares objektorientiertes Design wie auch eine bessere Lesbarkeit und Wartbarkeit ergibt.
- Sie sind definitiv leichter zu testen und zu debuggen als Active Server Pages.
- Sie sind mit dem Visual-Basic-Designer wesentlich einfacher zu entwickeln.

Wegen der Nachteile muß ich auf Microsofts Dokumentation zurückgreifen. Dort heißt es: »Active Server Pages sind für Script-Entwickler, die Webseiten entwikkeln wollen, und stellen die einzigartige Möglichkeit der Mischung von Script und HTML zur Verfügung.«

Einzigartige Möglichkeit? Ist das wirklich gut?

Ich vermute, daß dies aus internen Gründen bei Microsoft so aufrechterhalten wird, um die fortgesetzte Beschäftigung von Leuten zu rechtfertigen, die Active Server Pages und Assistenten für deren Entwicklung schreiben.

Aus meiner Perspektive bieten Web-Klassen eine weitaus bessere Lösung für alle mehr als nur trivialen Fälle.

Jedoch werden Puristen anmerken, daß Web-Klassen nichts weiter als eine Visual-Basic-Oberfläche darstellen, die auf dem Active Server Pages-System aufsetzt, wobei zu jeder Web-Klasse eine Active Server Page gehört, deren einziger Zweck der Aufruf des Web-Klassen-Objekts und seiner Unterobjekte ist. Puristen werden ebenfalls anmerken, daß Programmierer dies auch selbst machen könnten, indem sie ordinäre Visual-Basic-Code-Komponenten schreiben und diese von einer einfachen Active Server Page aus aufrufen.

Natürlich haben sie in allen Punkten recht. Und genau so habe ich meinen Ansatz bei meiner eigenen Web-Site www.desaware-com realisiert. Ich werde jedenfalls mit großer Begeisterung auf diese neue Technologie umschalten.