Kapitel 26

Lizenzierung und Verteilung

26.1 Der Sinn der Lizenzierung 780
26.2 Lizenzierung von ActiveX-Steuerelementen 783
26.3 Alternative Ansätze für die Lizenzierung 785

Das Computergeschäft muß einer der seltsamsten Industriezweige sein, die es je gegeben hat. Es gibt unglaublich viele Änderungen, die uns alle betreffen. Und die Computer haben enormen Einfluß auf die Gesellschaft, wobei ich glaube, das ist erst der Anfang. Dies sind alles Themen für ein eigenes Buch. Jetzt möchte ich über Geld sprechen.

Manchmal glaube ich, wir Programmierer haben eine Haßliebe zum Geld. Einerseits steht ständig zur Debatte, daß Information und Software nichts kosten sollten. Es gibt Tausende von Freeware- und Public-Domain-Applikationen, die zum Teil wirklich hervorragend sind. Es gibt zahllose Programmierer, die großartige Software nur so zum Spaß schreiben, oder um das Prinzip zu verstehen. Mir fällt kein anderer Industriezweig ein, wo es etwas Vergleichbares gäbe.

Und wie viele Programmierer kennen Sie, die »ganz nebenbei« noch irgendwelche Software entwickeln, in der Hoffnung, irgendwann so berühmt wie Netscape oder Microsoft zu werden? Ich kenne selbst ein paar. (Und nein, ich erwarte nicht, in die Fußstapfen von Netscape oder Microsoft zu treten.)

Diese Dichtomie macht sich auch während eines typischen Entwicklungszyklus bemerkbar. Wir sind alle Perfektionisten, suchen nach der kreativsten und elegantesten Lösung für ein Softwareproblem, und schreiben manchmal lieber unseren eigenen Code, statt den von jemandem anderen zu kaufen. Es besteht jedoch auch der Druck, den eng gesteckten Regeln zu entsprechen und die ökonomischste Lösung für die vorliegende Aufgabe zu finden. Die beiden Kräfte wirken entgegengesetzt. Als Software-Entwickler sollten wir eine Lösung als Mittelweg zwischen beidem finden.

Ich weiß nicht, inwieweit die Informatiklehrpläne dieses Thema heute berücksichtigen. Ich weiß, daß die Ökonomie eines Softwareprojekts keinen großen Teil meines Lehrplans in der Schule ausmachte. Parallel zu meinem Informatikstudium machte ich mein Ingenieursdiplom. Die ökonomischen Implikationen eines Projekts wurden dort als sehr wichtig erachtet, es gab sogar einen Kurs, der ganz diesem Thema gewidmet war. Ich gebe zu, daß dieser ökonomische Hintergrund meine Softwareentwicklung beeinflußt hat. Womit wir zum Thema der Software-Lizenzierung kommen.

26.1 Der Sinn der Lizenzierung

Als ich begann, dieses Kapitel zu schreiben, versuchte ich, mir die Bedürfnisse des typischen Visual-Basic-Programmierers in Hinblick auf Lizenzierung und Verteilung vorzustellen.

Ich sehe das Ganze als ISV (Independent Software Vendor), der kommerzielle Steuerelemente entwickelt und vermarktet. Bis Visual Basic 5 hatte ich angenommen, daß die große Vielzahl der Leser kein Interesse hätten, Steuerelemente oder Komponenten zu verkaufen. Die Entwicklung von Steuerelementen in Visual C++ ist zu schwierig, als daß viele Entwickler daran interessiert wären.

Aber Visual Basic macht jetzt die Entwicklung von ActiveX-Codekomponenten und Steuerelementen einfach. Und dadurch entstehen mehr Steuerelemente.

Dieses Kapitel geht davon aus, daß Sie, egal ob Sie für eine große Firma arbeiten, ein unabhängiger Softwareentwickler sind oder kommerzielle oder Shareware-Steuerelemente schreiben möchten, an einem Lizenzierungsmechanismus interessiert sind, um Ihre Technologie zu schützen.

26.1.1 Sicherheitsabstufungen

Als erstes müssen Sie sich fragen, wie viel Sicherheit Sie brauchen. Hier braucht es eine echte Abwägung, denn je sicherer Ihre Komponente ist, desto schwieriger wird sie zu benutzen.

Man könnte sich eine Komponente vorstellen, für die die Sicherheit hardware-seitig realisiert wird, mit einem dieser »Dongles«, die an den parallelen Port aufgesteckt werden. Stellen Sie sich vor, jeder Komponentenhersteller würde diesen Ansatz verfolgen. Für ein komplexes Programm würden dann Dutzende von Dongles an der Rückseite jedes Computers erforderlich – ein schrecklicher Gedanke!

Ohne eine Hardwarelösung (und möglicherweise in Kombination mit einer Hardwarelösung) ist es fast unmöglich, ein völlig unfehlbares Lizenzierungsschema zu schaffen. Falls es eines gibt – ich kenne es nicht.

Die Ansätze, die Sie in diesem Kapitel bisher kennengelernt haben, behaupten nicht, perfekt zu sein. Sie behaupten nicht einmal, sicher zu sein. Statt dessen folgen sie dem Ansatz, der von Microsoft und den meisten anderen Komponentenanbietern übernommen wurde, wobei eine Komponentenlizenzierung existiert, die es unpraktisch machen würde, Komponenten zu verwenden, die nicht lizenziert sind. Mit anderen Worten, Sie verteilen das Steuerelement mit einem Lizenzübereinkommen, das schärfste Konsequenzen androht, falls die Lizenz verletzt werden sollte. Sie nehmen ein einfaches Lizenzierungsschema auf, das die Benutzer warnt, wenn sie in Begriff sind, die Lizenz zu verletzen. Und dann vertrauen Sie einfach auf ihre Ehrlichkeit.

26.1.2 Lizenzierungsmodelle

Die Visual-Basic-Dokumentation beschreibt die Lizenzierung von ActiveX-Steuerelementen im Detail. Dort wird jedoch nur der Ansatz für ein einziges Lizenzierungsmodell beschrieben, und es wird nicht auf ActiveX-Codekomponenten eingegangen. Was sind Lizenzierungsmodelle? Sie stellen verschiedene Ansätze für die Verteilung von Komponenten dar. Betrachten Sie die folgenden Beispiele:

- **Lizenzierung zur Entwurfszeit**. Dies ist das Modell, das Visual Basic für ActiveX-Steuerelemente unterstützt, und das von vielen Steuerelementanbietern genutzt wird. Steuerelemente müssen lizenziert sein, damit sie in die Entwurfsumgebung des Containers geladen werden. Steuerelemente müssen nicht für die Ausführung in der Laufzeitumgebung lizenziert sein. Dieser Ansatz

wird auch von der ActiveX-Spezifikation in Form einer speziellen Schnittstelle unterstützt, `IClassFactory2`, die in jedes Visual-Basic-Steuerelement eingebaut ist, um die Lizenzierung zu unterstützen. Die Funktionalität dieser Schnittstelle ist in Visual Basic eingebaut und für VB-Programmierer transparent.

- **Lizenzierung Datei/Komponente**. Dieser Ansatz kann sowohl für Steuerelemente als auch für ActiveX-DLL-Codekomponenten verwendet werden. Er nutzt separate Dateien oder ActiveX-Komponenten für die Lizenzierung, statt der Systemregistrierung Lizenzierungsschlüssel hinzuzufügen.
- **Laufzeit-Lizenzierung**. Dieser Ansatz verwendet eine separate Datei oder ActiveX-Komponente, um eine Lizenzierung in allen Situationen bereitzustellen. Er wird in Fällen verwendet, wo Sie die Verwendung von Komponenten sowohl für die Laufzeit als auch für die Entwurfszeit lizenzieren möchten. Die Laufzeit-Lizenzierung ist auf dem aktuellen Markt für Add-On-Komponenten zu Visual Basic nicht gebräuchlich. Dieser Ansatz könnte jedoch praktisch für Applikationen sein, die Sie auf einem Firmen-Intranet einsetzen möchten, um zu verhindern, daß sie außerhalb Ihrer Firma verwendet werden.
- **Separate Demo-Komponenten**. Dieses Modell ist insbesondere für ActiveX-Codekomponenten praktisch. Er besteht darin, separate Demo-Komponenten zu kompilieren, die entweder eine eingeschränkte Funktionalität aufweisen oder nur ausgeführt werden können, wenn sie innerhalb des Kontexts bestimmter Entwicklungswerkzeuge aufgerufen werden, beispielsweise in Visual Basic. Dieser Ansatz soll Ihnen ermöglichen, Demo-Versionen eines Steuerelements weiterzugeben, während Sie den Zugriff auf die Release-Version kontrollieren.
- **Komponenten mit beschränkter Gültigkeitsdauer**. Dieser Ansatz aktiviert ein Steuerelement oder eine ActiveX-Codekomponente für eine beschränkte Zeitdauer. Nach Ablauf dieser Zeit deaktiviert es sich selbst, möglicherweise bis ein bestimmtes Paßwort eingegeben wird.

Welchen dieser Ansätze sollten Sie verwenden? Abgesehen von der offensichtlichen Tatsache, daß einige davon für Codekomponenten und ActiveX-Dokumenten verwendet werden können, hängt die Entscheidung weitgehend davon ab, was Sie damit bezwecken wollen. Wenn Sie einen Mechanismus brauchen, damit andere Ihre Steuerelemente ausprobieren können, könnten Sie eine Demo-Version oder eine Version mit beschränkter Gültigkeitsdauer verwenden. Wenn Sie nur beabsichtigen, eine unerlaubte Verwendung Ihrer Steuerelemente zu verhindern, könnten Sie den datei- oder registrierungsbasierten Ansatz verwenden. Die verschiedenen Ansätze können beliebig kombiniert werden.

26.2 Lizenzierung von ActiveX-Steuerelementen

Beginnen wir mit einem kurzen Überblick über den direkt von Visual Basic unterstützten Lizenzierungsansatz.

Sie erzeugen ein lizenziertes Steuerelement, indem Sie auf der Registerkarte ALLGEMEIN im Dialogfeld PROJEKTEIGENSCHAFTEN das Kontrollkästchen LIZENZIERUNGSSCHLÜSSEL ERFORDERLICH markieren. Es passiert folgendes:

- Für das Steuerelement ist immer ein Lizenzierungsschlüssel erforderlich, wenn es geladen werden soll.
- Visual Basic erzeugt eine `.VBL`-Lizenzdatei für das Steuerelement.
- Beim Kompilieren des Steuerelements registriert Visual Basic den Lizenzierungsschlüssel.

Diese Dinge wollen wir jetzt genauer betrachten.

Jedes lizenzierte Steuerelement unterstützt die Schnittstelle `IClassFactory2`, die beim Anlegen des Steuerelements erzeugt wird. Es handelt sich dabei um eine ActiveX-Standardschnittstelle, die insbesondere für diese Art der Lizenzierung entwickelt wurde. Der Container, der das Steuerelement lädt, muß einen Lizenzierungsschlüssel übergeben, einen eindeutigen String, der durch das Steuerelement definiert wird. Wenn der vom Container übergebene Schlüssel mit dem des Steuerelements übereinstimmt, wird das Steuerelement erfolgreich geladen.

Woher kommt dieser Lizenzierungsschlüssel? Visual Basic ermittelt den Schlüssel aus einer von zwei Quellen, abhängig davon, ob sich der Container im Entwurfsmodus oder im Ausführungsmodus befindet. Befindet sich der Container im Entwurfsmodus, sucht Visual Basic in der Registrierung nach einem Lizenzierungsschlüssel für das Steuerelement.

Wenn ein Steuerelement in einer Applikation kompiliert wird, speichert Visual Basic die Lizenzierungsschlüssel für jedes Steuerelement zusammen mit der Applikation. Auf diese Weise kann Visual Basic den gespeicherten Schlüssel benutzen, wenn ein Steuerelement in die laufende Applikation geladen wird, und seine Ausführung erlauben.

Steuerelemente, die aus Standard-Steuerelementen erzeugt werden, werden auf dieselbe Weise behandelt. Jedes dieser einzelnen Standard-Steuerelemente muß geladen werden, damit das Haupt-Steuerelement ausgeführt werden kann. Befindet sich der Container für das Steuerelement im Entwurfsmodus, versucht Visual Basic, für jedes der verwendeten Standard-Steuerelemente einen Lizenzierungsschlüssel zu finden. Ist eines dieser Steuerelemente lizenziert und es kann kein Lizenzierungsschlüssel gefunden werden, kann das gesamte Steuerelement nicht geladen werden. Ist ein Steuerelement in einer Applikation kompiliert, speichert Visual Basic den Lizenzierungsschlüssel für jedes dieser Standard-Steuerelemente mit der Applikation, so daß sie sie zur Laufzeit erzeugen kann.

Das bedeutet, wenn Sie lizenzierte Standard-Steuerelemente in Ihren Steuerelementen verwenden, müssen Sie für jedes davon Lizenzierungsschlüssel weitergeben, wenn Ihr Steuerelement zur Entwurfszeit genutzt werden soll. Sie sollten jedoch nie einen Lizenzierungsschlüssel weitergeben, wenn Sie nicht die entsprechende Berechtigung besitzen. Setzen Sie sich mit den Herstellern der Komponenten in Verbindung, um das Recht für die Verwendung zu erhalten.

Lizenzierungsschlüssel werden in der Registrierung unter dem Schlüssel `HKEY_CLASSES_ROOT\Licenses` gespeichert. Angenommen, Sie erzeugen das Steuerelement `xyz.ocx` mit der GUID:

```
A17C780E-23E5-11D2-A754-00001C1AD1F8
```

Wenn Sie das Projekt erzeugen, wird eine `.VBL`-Datei angelegt, die wie folgt aussehen könnte:

```
REGEDIT
HKEY_CLASSES_ROOT\Licenses = Licensing: Copying the keys may be a _
                    violation of established copyrights.
HKEY_CLASSES_ROOT\Licenses\A17C780E-23E5-11D2-A754-00001C1AD1F8 = _
                    mkkkjdkdjkmdgkhkqefdgdiklemehdjdidse
```

Diese `.VBL`-Datei enthält Anweisungen für die Registrierung des Lizenzierungsschlüssels und den Schlüssel selbst – das ist in der Regel ein zufällig zusammengesetzter Datenstring. Diese GUID erscheint in der Registrierung wie folgt als Unterschlüssel:

```
HKEY_CLASSES_ROOT\Licenses\ A17C780E-23E5-11D2-A754-00001C1AD1F8 =
      onmghnnglnlgpgknsmmnmnlnomrhgnigtnth
```

Der Setup-Assistent von Visual Basic verwendet die `.VBL`-Datei, um Anweisungen für die Lizenzierungsregistrierung in das Setup-Programm einzufügen. Wenn Sie dieses Setup-Programm ausführen, trägt es den Lizenzierungsschlüssel in die Registrierung ein. Wenn die Benutzer das Steuerelement nur auf ihre Systeme kopieren, funktioniert es in Applikationen problemlos; sie haben den Lizenzierungsschlüssel in der Applikation kompiliert. Wenn ein Benutzer jedoch versucht, das Steuerelement zur Entwurfszeit in ein Visual-Basic-Formular zu laden, findet Visual Basic den entsprechenden Lizenzierungsschlüssel nicht und lädt das Steuerelement auch nicht.

Dieser Ansatz weist eine große Einschränkung auf: Er hilft Ihnen nichts in Umgebungen, in denen es keine separaten Entwurfszeit- und Laufzeitmodi gibt. Microsoft-Office-Applikationen beispielsweise prüfen immer die Registrierung, ob dort ein Lizenzierungsschlüssel vorhanden ist. Wenn Sie also möchten, daß Ihr Steuerelement in Office-Applikationen genutzt werden kann, müssen Sie den Lizenzierungsschlüssel bereitstellen, und das kommt dann letztlich auf dasselbe hinaus, als würden Sie überhaupt keine Lizenzierung verwenden. ActiveX-Dokumente sind ebenfalls nicht in der Lage, Lizenzierungsschlüssel zu speichern. Sie müssen

also den Lizenzierungsschlüssel bei jedem Laden des Steuerelements prüfen. Ein alternativer Ansatz, der eine zusätzliche Kontrolle über lizenzierte Umgebungen erlaubt, wird später beschrieben.

Eine leichte Abwandlung des hier beschriebenen Ansatzes zur Lizenzierung von Steuerelementen bezieht sich auf die Verwendung von Steuerelementen auf Web-Seiten. Für diese Steuerelemente sind offensichtlich Lizenzierungsschlüssel erforderlich, damit sie funktionieren. Klar ist auch, daß die Registrierung der Lizenzierungsschlüssel für jedes heruntergeladene Steuerelement nicht praktikabel ist. Es käme der Situation gleich, eine uneingeschränkte Lizenz für die Verwendung jedes heruntergeladenen Steuerelements bereitzustellen. Und die Bereitstellung des Schlüssels auf der Web-Seite würde den Zugriff auf den Schlüssel ein bißchen zu einfach machen.

Browser, die ActiveX-Steuerelemente unterstützen, stellen deshalb einen Mechanismus bereit, eine Lizenzpaket-Datei herunterzuladen, eine `.LPK`-Datei. Diese Datei wird mit Hilfe des Programms `lpk_tool.exe` erzeugt, das Sie im `Tools`-Verzeichnis Ihrer Visual Basic CD-ROM finden. Dieses Programm erlaubt Ihnen, die Lizenzierungsschlüssel für alle Steuerelemente auf einer Web-Seite in einer Datei mit der Erweiterung `.LPK` abzulegen. Anschließend können Sie dem `<OBJECT>`-Tag der Web-Seite wie folgt einen Parameter hinzufügen:

```
<PARAM NAME="LPKPath" VALUE = "mylpk.lpk">
```

Damit wird dem Objekt ein spezieller Parameter hinzugefügt, `LPKPath`, der dem Browser den Namen der `.LPK`-Datei mitteilt, die geladen werden soll. Dieser Dateiname sollte relativ zur Seite sein, keine absolute URL, damit es schwieriger wird, Seiten zu kopieren und sie auf dem lokalen System auszuführen.

Warum brauchen Sie ein spezielles Werkzeug, um die Lizenzpaket-Datei zu erzeugen? Warum kann das nicht Visual Basic für Sie übernehmen? Weil die Datei die Lizenzierungsschlüssel für alle Steuerelemente auf einer Web-Seite enthalten muß, und Visual Basic nicht wissen kann, welche Steuerelemente sich auf einer bestimmten Seite befinden werden. Die Lizenzierungsschlüssel, die in dieser Datei bereitgestellt werden, werden vom Browser nur verwendet, um die Steuerelemente zu laden. Sie werden nicht in die Registrierung eingetragen.

26.3 Alternative Ansätze für die Lizenzierung

Sie haben gesehen, daß der Standardansatz für die Lizenzierung von Steuerelementen für Container, die immer die Registrierung für Lizenzierungsschlüssel verwenden, nicht zufriedenstellend funktioniert. Auch für die Lizenzierung von ActiveX-Dokumenten und ActiveX-Codekomponenten ist er nicht geeignet. Ich werde auf beide Aspekte gleich noch einmal zurückkommen. Zuerst wollen wir jedoch betrachten, wie eine Demo-Komponente implementiert wird. Die Techniken, die dazu verwendet werden, stellen eine gute Einführung für die Techniken anderer Steuerelement-Lizenzschemata dar.

26.3.1 Demo-Komponenten erzeugen

Die Programmgruppe gtpDT.VBG demonstriert einen alternativen Ansatz für die Lizenzierung. Die Projekte gtpLicDT.VBP und gtpLicRT.VBP stellen beide die Klasse gtpLicensed bereit, die den folgenden Code enthält:

```
Option Explicit

Public Sub A()
   ' Tastenfunktionen aufheben
   If Not VerifyLicense() Then
 MsgBox "Lizenz-Fehler - Diese Operation ist nicht erlaubt"
 Exit Sub
   End If
   MsgBox "Routine A wurde aufgerufen"
End Sub

Private Sub Class_Initialize()
   MsgBox "gtpLicensed-Objekt erzeugt"
End Sub
```

In diesem Beispiel wird die Funktionalität von Funktion A blockiert, wenn die Überprüfung der Lizenz fehlschlägt. Sie könnten einen noch extremeren Ansatz verfolgen und einen Fehler aufwerfen, möglicherweise während der Klasseninitialisierung, falls die Lizenzierung fehlschlägt. Normalerweise sollten in einer Codekomponente keine zufälligen Fehler auftreten, aber in diesem Fall ist es Ihnen vielleicht egal, wenn die Applikation auf unfreundliche Weise fehlschlägt. Um sicher zu sein, zeigen Sie jedoch zuerst ein Nachrichtenfeld oder ein Dialogfeld an, das das Problem erklärt, so daß der Endbenutzer nicht Stunden damit verbringt, herauszufinden, was schiefgelaufen ist.

Die Funktion VerifyLicense finden Sie im Modul Licenser.bas, das den folgenden Code enthält:

```
' Guide to the Perplexed
' Copyright © 1997-1998 by Desaware Inc. All Rights Reserved

' Options supported by this module include:
' DEMOVERSION - Verifies ok in VB environment only
' DLLCHECK - Requires license file in VB environment


Option Explicit

Private Declare Function GetModuleFileName Lib "kernel32" Alias "GetModuleFileNameA"
(ByVal hModule As Long, ByVal lpFileName As String, ByVal_ nSize As Long) As Long
Private Declare Function GetModuleHandle Lib "kernel32" Alias "GetModuleHandleA"
_ (ByVal lpModuleName As Long) As Long
```

```
#If DEMOVERSION Or DLLCHECK Then
' True zurückgeben, wenn die
' Demo auf einer VB-Plattform ausgeführt werden kann, Entwurfszeit oder Laufzeit
   Public Function IsVBEnvironment() As Boolean
   Dim thismod&
   Dim thisfile$
   Dim basename$
   Dim thispos%
   Dim thischar$

   On Error GoTo nogo
   thismod = GetModuleHandle(0)
   thisfile = String$(262, Chr$(0))
   Call GetModuleFileName(thismod, thisfile, 261)
   thisfile = Left$(thisfile, InStr(thisfile, Chr$(0)) - 1)
   thispos = Len(thisfile)
   Do
 thischar = Mid$(thisfile, thispos, 1)
 If thischar = "\" Or thischar = ":" Then Exit Do
 thispos = thispos - 1
   Loop While thispos > 0
   basename = LCase$(Mid$(thisfile, thispos + 1))
   If basename = "vb.exe" Or basename = "vb32.exe" Or _
 basename = "vb5.exe" Or basename = "vb5cce.exe" Or _
 basename = "vb6.exe" Then
    IsVBEnvironment = True
   End If
   Exit Function
nogo:
End Function
#End If

Public Function VerifyLicense() As Boolean
   Static PriorVerification As Boolean
   Static VerifiedOnce As Boolean
   ' Beschleunigte Rückgabe beim zweiten Durchlauf
   If PriorVerification Then
 VerifyLicense = VerifiedOnce
 Exit Function
   End If
   PriorVerification = True
   #If DEMOVERSION Then
 If Not IsVBEnvironment() Then
    ' Bildschirm fehlerhafte Lizenz
    frmAbout.MessageType = 1
    frmAbout.Show vbModal
    VerifiedOnce = False
 Else
```

```
    ' Optionaler Eröffnungs-Bildschirm
    frmAbout.MessageType = 0
    frmAbout.Show vbModal
    VerifiedOnce = True
 End If
 VerifyLicense = VerifiedOnce
 Exit Function
   #End If
   #If DLLCHECK Then
 If IsVBEnvironment Then
    ' Hier auf DLL oder Objekt prüfen
    ' Wenn nicht gefunden VerifyLicense auf False setzen
    VerifiedOnce = False
 Else
    VerifiedOnce = True
 End If
 VerifyLicense = VerifiedOnce
 Exit Function
   #End If
   VerifiedOnce = True    ' Default ok
   VerifyLicense = True
End Function
```

Das Verhalten dieses Moduls ist von zwei Konstanten für die bedingte Kompilierung abhängig. Hier werden wir die Konstante `DEMOVERSION` betrachten.

ActiveX-Komponenten können den Unterschied zwischen Entwurfszeit und Laufzeit nicht erkennen. Die Konstante zur bedingten Kompilierung, `DEMOVERSION`, soll Ihnen ermöglichen, auf einfache Weise zwei Versionen einer Komponente zu kompilieren – eine, die immer läuft, und eine, die nur in der Visual-Basic-Umgebung läuft.

Wenn diese Komponente und `DLLCHECK` nicht definiert oder 0 sind, gibt die Funktion `VerifyLicense` immer `True` zurück. Damit ergibt sich eine sehr schnelle Überprüfung der regulären Komponente. Wenn die Komponente `True` ist, ruft die Funktion `VerifyLicense` zuerst die Funktion `IsVBEnvironment` auf, um zu prüfen, ob der aktuelle Prozeß Visual Basic ist. Dazu verwendet sie die Funktion `GetModuleHandle` mit einem Null-Parameter, um den Instanzhandle der laufenden Applikation zu ermitteln. Die Funktion `GetModuleFileName` kann dann genutzt werden, um den Namen der ausführbaren Datei für die Applikation zu ermitteln. Wenn es sich dabei um einen der möglichen Programmdateinamen von Visual Basic handelt, gibt die Funktion `True` zurück, d.h. die Komponente wird unter Visual Basic ausgeführt.

Natürlich können Sie die Liste erweitern, um beliebige andere Applikationen aufzunehmen. Sie können sogar eine separate Datei verwenden, oder mehrere Registrierungseinträge, um anzugeben, in welchen Applikationen die Demo-Komponente ausgeführt werden darf.

Die Funktion `VerifyLicense` führt diese Überprüfung nur einmal aus. Danach gibt sie die Ergebnisse der ersten Überprüfung zurück. Das dient zweierlei Zwecken. Erstens wird die Effizienz damit gesteigert. Zweitens ist es dadurch einfach, einen Eröffnungsbildschirm zu unterstützen, der nur beim ersten Laden der DLL erscheint.

Wenn `VerifyLicense` zum ersten Mal aufgerufen wird, wird das Formular `frmAbout` angezeigt. Der Code dafür sieht wie folgt aus:

```
' Guide to the Perplexed
' Beispiel für einen Lizenzierungs-Dialog

Option Explicit

Public MessageType As Integer

Private Sub Command1_Click()
   Unload Me
End Sub

Private Sub Form_Load()
   Select Case MessageType
 Case 0
    lblMessage.Caption = "This is a demo version of the gtpLicensed_
    component"
 Case 1
    lblMessage.Caption = "This component is unlicensed."
   End Select
End Sub
```

Die öffentliche Variable `MessageType` erlaubt dem aufrufenden Modul, die Warnmeldung auszugeben, den Standard-Eröffnungsbildschirm, der die Komponente beschreibt, oder eine Warnung, die die Lizenzierung betrifft. Etwas sollten Sie beachten, wenn Sie einen Eröffnungsbildschirm anzeigen wollen. Wenn Sie ein Objekt aus der DLL erzeugen und es dann freigeben, wird die DLL nach einer Weile aus dem Speicher entfernt. Wenn Sie sie irgendwann wieder laden, erscheint dann auch wieder der Eröffnungsbildschirm.

Es gibt nur drei Unterschiede zwischen den Projekten `gtpLicDT` und `gtpLicRT`:

- Für das Projekt `gtpLicDT` ist die Variable für die bedingte Kompilierung, `DEMOVERSION`, auf 1 gesetzt.
- Die Projektnamen sind unterschiedlich.
- Das Projekt `gtpLicRT` beinhaltet nicht das Formular `frmAbout`.

Warum sollten Sie eine bedingte Kompilierung verwenden, um zwischen den Komponenten zu unterscheiden, statt einfach einer globalen Konstanten? Das hat die folgenden Gründe:

- Sie bietet die bestmögliche Performance bei der Nicht-Demo-Version.
- Die beiden Projekte können genau dieselbe Codebasis verwenden. Der einzige Unterschied liegt in der Projektdatei.
- Die Laufzeitkomponente kann sicher Formulare ausschließen, die nur von der Demo-Version verwendet werden. In diesem Fall wird im Code, der für das Projekt `gtpLicRT` kompiliert wurde, nie auf das Formular `frmAbout` verwiesen.

Wenn Sie das Formular `frmLicTest.frm` im Projekt `LicTest` betrachten, sehen Sie den folgenden Code:

```
' Es ist ganz einfach, den Verweis zu ändern
Dim obj As New gtpLicensed

Private Sub Command1_Click()
   obj.A
End Sub
```

Angenommen, das Projekt wird zunächst so entwickelt, daß es die Demo-Komponente `gtpLicDT` verwendet. Wie können Sie zu der anderen Komponente wechseln? Ganz einfach. Ändern Sie nur den Verweis im Dialogfeld `Projektverweise`, so daß er auf das Projekt `gtpLicRT` verweist, statt auf das Projekt `gtpLicDT`. Andere Codeänderungen sind nicht erforderlich.

26.3.2 Dateibasierte Lizenzierung von Steuerelementen

Eine leichte Abwandlung dieses Ansatzes sehen Sie in der Programmgruppe `ctlTest.vbg`, die das Projekt `LicCtl.vbp` und ein Testprojekt, `ctltest.vbp`, enthält. Das Steuerelement verwendet dasselbe Lizenzierungs-Modul, wie Sie es im vorigen Beispiel gesehen haben, aber in diesem Fall wird die Variable für die bedingte Kompilierung, `DLLCHECK`, definiert.

Die Prozedur `VerifyLicense` kann bei der Initialisierung des Steuerelements oder während der Ereignisse `InitProperties` und `ReadProperties` aufgerufen werden. Wenn Sie das letztere wählen, achten Sie darauf, die Überprüfung der Lizenzierung während `InitProperties` und `ReadProperties` auszuführen, um damit das Anlegen neuer Steuerelemente und das Laden existierender Projekte abzudecken.

```
Option Explicit

Private Sub UserControl_Initialize()
   If Not VerifyLicense Then
```

```
 MsgBox "Licensing error"
 Err.Raise vbObjectError + 1000, "Control", "Das Steuerelement ist für _
diese Plattform nicht lizenziert"
   End If
End Sub

Private Sub UserControl_InitProperties()
   ' Ein alternativer Ansatz
   'If (Not Ambient.UserMode) And (Not VerifyLicense) Then
   ' MsgBox "Licensing error"
   ' Err.Raise vbObjectError + 1000, "Control", "Das Steuerelement ist _
für diese Plattform nicht lizenziert"
   'End If
End Sub
```

Diese Überprüfung sollte nur dann bei der Initialisierung ausgeführt werden, wenn es egal ist, ob sich das Steuerelement im Entwurfsmodus oder im Laufzeitmodus befindet. Beispielsweise könnten Sie diesen Ansatz nutzen, um das Steuerelement für die Verwendung in Office-Applikationen zu aktivieren oder zu deaktivieren. Wenn Sie die Überprüfung vom `UserMode` des Containers abhängig machen wollen, müssen Sie bis zu den Ereignissen `InitProperties` oder `ReadProperties` warten, wenn das `Ambient`-Objekt existiert. Bei ActiveX-Dokumenten sollten Sie bis zum `Show`-Ereignis warten, dann können Sie die Ausführung ebenfalls vom Container abhängig machen. Es steht über die `Parent`-Eigenschaft zur Verfügung.

Die eigentliche Überprüfung kann von beliebigen Kriterien abhängig gemacht werden. Ein gebräuchlicher Ansatz ist, eine separate DLL oder ActiveX-DLL bereitzustellen, die Informationen über die Entwurfszeit-Lizenzierung enthält. Sie können die Existenz der Versionsinformation in dieser DLL zu beliebigen Zeitpunkten abfragen.

```
#If DLLCHECK Then
 If IsVBEnvironment Then
    ' Hier auf DLL oder Objekt prüfen.
    ' Falls nicht gefunden, Set VerifyLicense auf False setzen
 Else
    VerifiedOnce = True
 End If
 VerifyLicense = VerifiedOnce
 Exit Function
  #End If
```

Wenn die Lizenzierung fehlschlägt, gibt es mehrere Möglichkeiten. Die Visual-Basic-Dokumentation warnt Sie zu Recht, Fehler aufzuwerfen, wenn der Container sie nicht erwartet, beispielsweise während InitProperties, ReadProperties und der Komponenteninitialisierung. Warum breche ich diese Regel hier? Weil ich voraussetze, daß der Versuch, auf eine nicht lizenzierte Komponente zuzugreifen,

ein fataler Fehler ist. Ich möchte, daß der Container, wenn möglich, abgebrochen wird. Das Aufwerfen eines Fehlers garantiert jedoch nicht, daß das Steuerelement nicht geladen werden kann (obwohl das in den Containern, mit denen ich es ausprobiert habe, schon der Fall war). Sie sollten also die Tatsache, daß die Lizenzierung fehlgeschlagen ist, in einer Variablen auf Modulebene festhalten, und diese Variable nutzen, um gegebenenfalls verschiedene Operationen des Steuerelements zu deaktivieren.

Wie zuvor zeigt das Steuerelement ein Nachrichtenfeld an, um den Endbenutzer darüber zu informieren, daß es ein Problem mit der Lizenzierung gibt. Ein besserer Ansatz wäre, ein modales Dialogfeld zu verwenden, das Informationen darüber enthält, wie man die Lizenz erhält.

Diese beiden einfachen Beispiele zeigen einige wichtige Grundregeln für die Lizenzierung. Betrachten Sie zunächst, welche Informationen Ihnen zur Verfügung stehen:

- Der Name der Programmdatei, die den aktuellen Prozeß ausführt (der aufrufende Prozeß für ActiveX-DLLs, DLL-basierte ActiveX-Dokumente und ActiveX-Steuerelemente).
- Ob sich der Container im Entwurfsmodus oder im Laufzeitmodus befindet (nur ActiveX-Steuerelemente).

Jetzt betrachten wir, welche Werkzeuge Ihnen zur Verfügung stehen, um anzuzeigen, daß ein Steuerelement lizenziert ist:

- Sie können eine Text-Lizenzdatei weitergeben.
- Sie können verborgene Dateien für Lizenzzwecke erzeugen.
- Sie können eine DLL oder ActiveX-DLL für Lizenzzwecke weitergeben.
- Sie können der Registrierung eigene Einträge hinzufügen.

Kombinieren Sie beides, und Sie sehen, daß es geradezu unendlich viele Möglichkeiten gibt, die Lizenzierung beliebiger ActiveX-Komponenten zu kontrollieren.

Beispielsweise könnte eine Komponente mit eingeschränkter Gültigkeitsdauer wie folgt implementiert werden:

- Wenn das Steuerelement zum ersten Mal geladen wird, fügt es einen Registrierungseintrag oder eine verborgene Datei ein, die das Datum der ersten Verwendung enthält.
- Das Steuerelement fragt diesen verborgenen Eintrag bei jedem Laden ab. Wenn das Steuerelement unter Visual Basic läuft und seine Lebensdauer abgelaufen ist, wird es mit einem Fehler beendet.

- Sie können ein Programm zur Registrierung verwenden, das Sie den Kunden liefern, wenn diese das Steuerelement kaufen. Dieses Programm aktualisiert den Eintrag in der Registrierung oder in der Datei, um damit anzuzeigen, daß das Steuerelement jetzt lizenziert ist.

Der Benutzer könnte natürlich herausfinden, welchen Eintrag Sie in die Registrierung geschrieben haben, oder die verborgene Datei ermitteln und die Testdauer auf den Anfang zurücksetzen. Sie könnten auch die Uhr in ihrem System zurücksetzen. Aber die Absicht war hier nur, den Benutzer zu warnen, weil alle diese Ansätze davon ausgehen, daß die meisten Entwickler ehrlich sind. Sie finden auch Produkte von Drittanbietern, die eine Verschlüsselung verwenden, um eine höhere Sicherheit zu realisieren, oder die das Verschlüsselungs-API verwenden, um ein eigenes, komplexeres Lizenzierungsschema zu implementieren, wenn Sie nicht so viel Vertrauen aufbringen können.