(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 698 12 139 T2** 2004.01.08

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 023 664 B1**
(21) Deutsches Aktenzeichen: **698 12 139.2**
(86) PCT-Aktenzeichen: **PCT/US98/15340**
(96) Europäisches Aktenzeichen: **98 935 976.5**
(87) PCT-Veröffentlichungs-Nr.: **WO 99/005600**
(86) PCT-Anmeldetag: **24.07.1998**
(87) Veröffentlichungstag der PCT-Anmeldung: **04.02.1999**
(97) Erstveröffentlichung durch das EPA: **02.08.2000**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **12.03.2003**
(47) Veröffentlichungstag im Patentblatt: **08.01.2004**

(51) Int Cl.[7]: **G06F 12/00**

(30) Unionspriorität:
901776 **28.07.1997** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

(73) Patentinhaber:
Apple Computer, Inc., Cupertino, Calif., US

(72) Erfinder:
GARST, Blaine, Belmont, US; SERLET, Bertrand, Palo Alto, US

(74) Vertreter:
BOEHMERT & BOEHMERT, 28209 Bremen

(54) Bezeichnung: **VERFAHREN UND VORRICHTUNG ZUR SOFTWARE-LIZENZ-ERZWINGUNG**

Beschreibung

HINTERGRUND DER ERFINDUNG

1. GEBIET DER ERFINDUNG

[0001] Die vorliegende Erfindung betrifft allgemein die Verbreitung von Computer-Software und insbesondere ein Verfahren und eine Vorrichtung zur automatischen Erzwingung von Lizenzen für Computer-Software.

2. STAND DER TECHNIK

[0002] Einige Computer-Software-Programme nutzen so genannte „Betriebsmittelbibliotheken", um einen Teil ihrer Funktionalität zu ermöglichen. Um eine Betriebsmittelbibliothek zu benutzen, ist meistens eine Lizenzgebühr notwendig. Bei aktuellen Verfahren ist es nicht immer möglich, allen Benutzern einer Betriebsmittelbibliothek die Lizenzgebühr in Rechnung zu stellen. Dieses Problem kann verstanden werden, wenn Software-Strukturen, die Betriebsmittelbibliotheken nutzen, mit Software-Grundstrukturen, die keine Betriebsmittelbibliothek nutzen, verglichen werden.

Software-Grundstruktur

[0003] **Fig.** 1 zeigt eine Software-Grundstruktur. In dem Beispiel von **Fig.** 1 weist die Software zwei Schichten auf. Diese Schichten sind das Betriebssystem **110** und das Anwendungsprogramm **120**. Das Betriebssystem **110** ist für die Steuerung der Zuordnung und Nutzung von Hardware-Betriebsmitteln, wie Speicher, Zentraleinheitszeit (CPU-Zeit), Speicherplatz auf der Festplatte und Peripheriegeräte, verantwortlich. Das Betriebssystem **110** stellt verschiedene spezielle Funktionen bereit, die von verschiedenen Software-Programmen, wie dem Anwendungsprogramm **120**, genutzt werden können. Das Software-Programm **120** stellt spezielle Endbenutzer-Funktionen bereit, wie Textverarbeitung, Datenbankverwaltung und anderes. Das Anwendungsprogramm **120** ist mit der Computer-Hardware über Funktionen verbunden, die vom Betriebssystem **110** bereitgestellt werden. Das Betriebssystem **110** stellt eine Schnittstelle zwischen der Hardware **100** und dem Anwendungsprogramm **120** bereit.

Betriebsmittelbibliotheken

[0004] **Fig.** 2 zeigt eine zweite Software-Struktur. Die Software-Struktur von **Fig.** 2 enthält eine weitere Software-Schicht, die Betriebsmittelbibliothek **215**, die zwischen das Anwendungsprogramm **120** und das Betriebssystem **110** geschaltet ist. Die Betriebsmittelbibliothek **215** stellt eine vorgepackte Gruppe von Betriebsmitteln oder Routinen bereit, auf die von Software-Programmen, wie dem Anwendungsprogramm **220**, bei der Abarbeitung zugegriffen werden kann. Diese Betriebsmittel stellen Funktionen auf einer höheren Ebene als die bereit, die von einem Betriebssystem **210** bereitgestellt werden. Beispielsweise können diese Betriebsmittel Routinen zum Verwalten einer grafischen Benutzeroberfläche, zum Kommunizieren mit anderen Computern über ein Netz oder zum Übertragen von Nachrichten zwischen Programmobjekten bereitstellen. Normalerweise stellt die Betriebsmittelbibliothek **215** ein oder mehr Betriebsmittel oder Funktionen bereit, die von vielen verschiedenen Software-Programmen verwendet werden können. Durch Verwendung der von der Betriebsmittelbibliothek **215** bereitgestellten vorgepackten Betriebsmittel kann ein Software-Programm, wie das Anwendungsprogramm **220**, kleiner gemacht werden und die Programmentwicklungszeit kann verkürzt werden, da das Programm selbst keinen Code haben muss, um die von der Betriebsmittelbibliothek **215** bereitgestellten Funktionen bereitzustellen.

[0005] Außer von Anwendungsprogrammen werden Betriebsmittel auch von anderen Arten von Software-Programmen, unter anderem Gerätetreibern, Hilfsprogrammen und anderen Betriebsmittelbibliotheken verwendet.

[0006] Die Betriebsmittelbibliothek **215** stellt eine Gruppe von ein oder mehr Betriebsmitteln dar, die getrennt von einem Anwendungsprogramm oder anderen Software-Programm vorliegen und die von mehr als einem Software-Programm verwendet werden können. Beispielsweise kann die Betriebsmittelbibliothek **215** eine Anwendungsprogramm-Schnittstelle (API), ein Toolkit, ein Framework, eine Betriebsmittelbibliothek, eine Dynamic Link Library (DLL), ein Applet oder ein anderes wiederverwendbares Betriebsmittel, unter anderem ein Anwendungsprogramm, auf das von einem anderen Programm [z. B. durch Verwendung der Objektverknüpfung und -einbettung (OLE)] zugegriffen werden kann, aufweisen. Beispiele für Betriebsmittelbibliotheken sind Windows-DLLs (DLLs, die mit der Microsoft-Windows™-Betriebssystemumgebung verwendet werden), das Apple-Macintosh™-Toolkit, die OpenStep-API von NeXT Software, Inc., OLE-aktivierte Anwendungsprogramme, wie Microsoft Word™, Java-Programmpakete und ActiveX-Applets.

[0007] Ein Software-Programm nutzt normalerweise ein Betriebsmittel, das von einer Betriebsmittelbibliothek bereitgestellt wird, indem es eine entsprechende Nachricht an die Betriebsmittelbibliothek sendet und die Parameter liefert, die für das abzuarbeitende Betriebsmittel erforderlich sind. Vorausgesetzt, die entsprechenden Parameter sind geliefert worden, wird das Betriebsmittel abgearbeitet und es wird eine entsprechende Antwortmeldung an das anfordernde Programm zurückgesendet.
[0008] Ein Software-Programm kann Betriebsmittel verwenden, die von mehreren verschiedenen Betriebsmittelbibliotheken bereitgestellt werden, eine Betriebsmittelbibliothek kann von mehreren verschiedenen Programmen genutzt werden, und eine Betriebsmittelbibliothek kann selbst andere Betriebsmittelbibliotheken nutzen. **Fig.** 3 zeigt ein Computersystem, das mehrere Programme und mehrere Betriebsmittelbibliotheken umfasst. In dem Beispiel von **Fig.** 3 gibt es zwei Anwendungsprogramme **300** und **310** und drei Betriebsmittelbibliotheken **320**, **330** und **340**. Das Anwendungsprogramm **300** verwendet Betriebsmittel, die vom Betriebssystem **110** und von den Betriebsmittelbibliotheken **320** und **330** bereitgestellt werden. Das Anwendungsprogramm **310** verwendet Betriebsmittel, die vom Betriebssystem **110** und von den Betriebsmittelbibliotheken **330** und **340** bereitgestellt werden. Die Betriebsmittel der Betriebsmittelbibliothek **330** werden somit von den Anwendungsprogrammen **300** und **310** gemeinsam genutzt.

Lizenzgebühr

[0009] In der Regel wird einem Endbenutzer eine Lizenz für Computer-Software für eine Gebühr erteilt. Der Endbenutzer zahlt einen einmaligen Kaufpreis oder eine einmalige Lizenzgebühr für das Recht, das Endbenutzer-Programm auf einem Computersystem zu benutzen. Betriebsmittelbibliotheken werden vom Hersteller des Programms oft im Paket mit einem Endbenutzer-Programm als „Bundle" angeboten, sodass der Endbenutzer eine Kopie der von einem Programm benötigten Betriebsmittelbibliotheken erhält, wenn er eine Kopie des Programms kauft. Der Preis der Betriebsmittelbibliothek ist in den Preis des Endbenutzer-Programms integriert. Der Entwickler des Endbenutzer-Programms entrichtet wiederum eine Nutzungsgebühr an den Verkäufer der Betriebsmittelbibliothek für das Recht, die Betriebsmittelbibliothek im Paket anzubieten und weiterzuverkaufen.
[0010] Da eine Betriebsmittelbibliothek mit mehreren Endbenutzer-Programmen genutzt werden kann, kann der Endbenutzer, wenn er eine Kopie der Betriebsmittelbibliothek erhält, die Betriebsmittelbibliothek mit einem anderen Programm benutzen, das mit der Betriebsmittelbibliothek kompatibel ist. In diesem Fall erhält der Verkäufer der Betriebsmittelbibliothek zusätzliche Einnahmen, wenn die Betriebsmittelbibliothek des Verkäufers mit weiteren Programmen genutzt wird. Daher wäre es für einen Verkäufer einer Betriebsmittelbibliothek wünschenswert, sicherstellen zu können, dass ein Endbenutzer die Betriebsmittelbibliothek nur mit Programmen nutzen kann, für die an den Verkäufer für die Benutzung der Betriebsmittelbibliothek eine Lizenzgebühr gezahlt worden ist. Somit besteht Bedarf an einem Software-Mechanismus zur Erzwingung von Software-Lizenzverträgen, der automatisch sicherstellt, dass eine Betriebsmittelbibliothek nur von Programmen genutzt werden kann, die vom Betriebsmittelbibliothek-Verkäufer zur Verwendung mit der Betriebsmittelbibliothek lizenziert worden sind.
[0011] In EP 0.570.123 sind ein Verfahren und eine Vorrichtung beschrieben, die ein System-Überwachungsprogramm aufweisen, das die Fähigkeit eines Programms, das gerade ausgeführt werden soll, auf die Verwendung von vorgegebenen Betriebsmitteln (z. B. Datendateien, Plattenbeschreibbarkeit usw.) beschränkt. Das System-Überwachungsprogramm verarbeitet eine Datenstruktur mit einer Gruppe von Befugnissen, die das, was ein Programm tun darf, und/oder das, woran das Programm gehindert werden soll, definieren. Die Gruppe von Befugnissen und/oder Beschränkungen, die einem auszuführenden Programm zugewiesen wird, wird als „Programm-Autorisierungsinformationen" bezeichnet. Wenn die Programm-Autorisierungsinformationen definiert sind, werden sie danach mit mindestens einem auszuführenden Programm assoziiert, um dadurch die Betriebsmittel und Funktionen zu beschreiben, die das Programm nutzen und/oder nicht nutzen darf.

KURZE DARSTELLUNG DER ERFINDUNG

[0012] Die vorliegende Erfindung umfasst ein Verfahren und eine Vorrichtung zur Erzwingung von Software-Lizenzen für Betriebsmittelbibliotheken. Der hier verwendete Begriff „Betriebsmittelbibliothek" bezieht sich auf jedes wiederverwendbare Software-Betriebsmittel, das von mehr als einem Programm oder einer anderen Betriebsmittelbibliothek verwendet werden kann. Der Begriff „Betriebsmittelbibliothek" beinhaltet unter anderem eine Anwendungsprogramm-Schnittstelle (API), ein Toolkit, ein Framework, eine Laufzeitbibliothek, eine Dynamit Link Library (DLL), ein Applet (z. B. ein Java- oder ActiveX-Applet), ein Anwendungsprogramm, auf dessen Funktionalität von anderen Programmen (z. B. mittels OLE) zugegriffen werden kann, oder ein anderes wiederverwendbares Betriebsmittel. Mit der vorliegenden Erfindung kann die Betriebsmittelbibliothek selektiv nur von berechtigten Endbenutzer-Software-Programmen genutzt werden. Die vorliegende Erfindung kann verwendet werden, um ein Lizenzierungssystem „für ein Programm" für eine Betriebsmittelbibliothek zu erzwin-

gen, wodurch die Betriebsmittelbibliothek nur zur Verwendung mit bestimmten Software-Programmen lizenziert wird, und sie kann auch zur Erzwingung von Standortlizenzen und anderen Lizenzierungssystemen verwendet werden.

[0013] Bei einer Ausführungsform werden ein Zugriffsberechtigungsanzeiger, wie etwa ein Lizenztextstring, und ein entsprechender Lizenzschlüssel in ein Programm eingebettet, das zur Nutzung einer Betriebsmittelbibliothek lizenziert worden ist. Der Lizenztextstring und der Lizenzschlüssel werden beispielsweise von einem Betriebsmittelbibliothek-Verkäufer an einen Programmentwickler geliefert, der die Betriebsmittelbibliothek mit einem Endbenutzer-Programm, das gerade entwickelt wird, benutzen will.

[0014] Der Lizenztextstring umfasst Informationen zu den Bedingungen der Lizenz, unter denen das Endbenutzer-Programm die Betriebsmittelbibliothek nutzen darf. Bei einer Ausführungsform ist der Lizenzschlüssel eine algorithmische Ableitung, wie beispielsweise eine digitale Signatur, des Lizenztextstrings, die zur Authentisierung des Lizenztextstrings verwendet wird. Die Betriebsmittelbibliothek wiederum hat eine Prüfroutine, die Mittel zum Lesen des Lizenztextstrings und des Lizenzschlüssels und zum Bestimmen mit dem Lizenzschlüssel, ob der Lizenztextstring authentisch ist und ob der Lizenztextstring geändert worden ist, aufweist. Die Betriebsmittelbibliothek-Funktionen werden nur einem Programm zur Verfügung gestellt, das einen authentischen und ungeänderten Lizenztextstring hat.

[0015] Bei einer Ausführungsform stellt der Lizenzschlüssel die digitale Betriebsmittelbibliothek-Verkäufer-Signatur des Lizenztextstrings dar. Die Betriebsmittelbibliothek hat eine Prüfroutine zur Überprüfung der digitalen Signatur des Betriebsmittelbibliothek-Verkäufers. Die Betriebsmittelbibliothek wird nur dann geöffnet und zur Verwendung mit dem anfordernden Programm zugänglich gemacht, wenn der Lizenztextstring von der Betriebsmittelbibliothek als authentisch bestätigt wird. Bei einem gegebenen Programm kann nur der Eigentümer der Betriebsmittelbibliothek einen Lizenzschlüssel für einen bestimmten Lizenzvertrag erzeugen, der die Betriebsmittelbibliothek für dieses Programm und nur für dieses Programm öffnet. Jede Modifikation des Lizenzschlüssels oder des Lizenzvertrags-Textstrings in dem anfordernden Software-Programm wird von der Prüfroutine erkannt, was dazu führt, dass die Betriebsmittelbibliothek geschlossen bleibt. Der Lizenztextstring kann auch ein Ablaufdatum für die Lizenz angeben; in diesem Fall wird die Betriebsmittelbibliothek nur geöffnet, wenn das Ablaufdatum noch nicht erreicht ist.

[0016] Bei einer Ausführungsform wird ein Erzwingungsverfahren für einen Standort bereitgestellt, bei dem ein Software-Programm, das an einem bestimmten Benutzer-Standort vorliegt, mit der Betriebsmittelbibliothek funktioniert, wenn die Betriebsmittelbibliothek den richtigen Lizenzschlüssel für einen Standort erhält.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0017] **Fig.** 1 zeigt ein Beispiel für eine Software-Struktur.

[0018] **Fig.** 2 zeigt ein Beispiel für eine Software-Struktur mit einer Betriebsmittelbibliothek.

[0019] **Fig.** 3 zeigt ein Beispiel für eine Software-Struktur mit mehreren Anwendungsprogrammen und Betriebsmittelbibliotheken.

[0020] **Fig.** 4 zeigt eine Ausführungsform eines Computersystems, das mit der vorliegenden Erfindung verwendet werden kann.

[0021] **Fig.** 5 zeigt eine Software-Struktur einer Ausführungsform der vorliegenden Erfindung.

[0022] **Fig.** 6 zeigt eine Software-Struktur einer Ausführungsform der vorliegenden Erfindung.

[0023] **Fig.** 7 ist ein Ablaufdiagramm, das die Funktionsweise einer Ausführungsform der vorliegenden Erfindung zeigt.

[0024] **Fig.** 8 zeigt eine Software-Struktur einer Ausführungsform der vorliegenden Erfindung.

[0025] **Fig.** 9 zeigt eine Software-Struktur einer Ausführungsform der vorliegenden Erfindung.

[0026] **Fig.** 10 ist Ablaufdiagramm, das die Funktionsweise einer Ausführungsform der vorliegenden Erfindung zeigt.

[0027] **Fig.** 11 ist Ablaufdiagramm, das die Funktionsweise einer Ausführungsform der vorliegenden Erfindung zeigt.

[0028] **Fig.** 12 ist Ablaufdiagramm, das die Funktionsweise einer Ausführungsform der vorliegenden Erfindung zeigt.

[0029] **Fig.** 13 zeigt eine Software-Struktur einer Ausführungsform der vorliegenden Erfindung, die eine OpenStep-API verwendet.

[0030] **Fig.** 14 zeigt eine Ausführungsform der Erfindung, bei der die Betriebsmittelbibliothek ein Applet ist.

DETAILLIERTE BESCHREIBUNG DER ERFINDUNG

[0031] Es werden ein Verfahren und eine Vorrichtung zur Erzwingung von Software-Lizenzen beschrieben. In der folgenden Beschreibung werden zahlreiche spezielle Einzelheiten dargelegt, um eine umfassendere Darstellung der vorliegenden Erfindung zu ermöglichen. Ein Fachmann dürfte jedoch erkennen, dass die vorlie-

gende Erfindung ohne diese speziellen Einzelheiten genutzt werden kann. In anderen Fällen sind bekannte Merkmale nicht näher beschrieben worden, damit die Erfindung nicht schwer verständlich wird.

Computersystem

[0032] Die vorliegende Erfindung kann auf einem von verschiedenen Computersystemen, unter anderem Netzwerkrechnern, Spezialrechnern und Universalrechnern, wie dem in **Fig.** 4 gezeigten Universalrechner, realisiert werden. Das in **Fig.** 4 gezeigte Computersystem umfasst eine CPU **400**, die einen Zentralprozessor (CPU), einen Hauptspeicher, periphere Schnittstellen, Eingabe/Ausgabe-Geräte, ein Netzteil und zugehörige Schaltkreise und Anordnungen aufweist; einen Bildschirm **410**, der ein Katodenstrahlröhren-Bildschirm, LCD-Bildschirm, Plasmabildschirm oder ein anderer Computer-Bildschirm sein kann; ein Eingabegerät **430**, das eine Tastatur, eine Maus, ein Digitizer oder ein anderes Eingabegerät sein kann; einen Permanentspeicher **420**, der magnetische, wiederbeschreibbare optische oder andere Massenspeicher umfassen kann; ein transportables Medienlaufwerk **425**, das magnetische, wiederbeschreibbare optische oder andere transportable Wechselmedien aufweisen kann; und einen Drucker **450**. Das Computersystem kann auch eine Netzwerk-Schnittstelle **440** aufweisen, die ein Modem aufweisen kann, mit dem das Computersystem über ein Kommunikationsnetz wie das Internet mit anderen Systemen kommunizieren kann. Es können auch verschiedene andere Konfigurationen von Computersystemen verwendet werden. Bei einer Ausführungsform weist das Computersystem eine Intel-Pentium™-CPU auf und läuft in der Microsoft-Windows-95™-Betriebssystemumgebung. Bei einer anderen Ausführungsform weist das Computersystem eine Motorola-CPU der Baureihe 680X0 auf und läuft unter dem Betriebssystem NeXTStep.

[0033] Wenn ein Computersystem die hier beschriebenen Prozesse und Prozessabläufe abarbeitet, ist es ein Mittel zur Erzwingung von Software-Lizenzen.

[0034] Die Erfindung kann in einem Computerprogrammcode in jeder gewünschten Computer-Programmiersprache realisiert werden.

Lizenzierungsmodul

[0035] **Fig.** 5 ist Blockdiagramm, das Software-Komponenten einer Ausführungsform der vorliegenden Erfindung zeigt. Wie in **Fig.** 5 gezeigt, weist diese Ausführungsform wie die herkömmliche Ausführungsform von **Fig.** 2 eine Computer-Hardware **100**, ein Betriebssystem **110**, ein Anwendungsprogramm **220** und eine Betriebsmittelbibliothek **215** auf. Die vorliegende Erfindung verwendet jedoch zwei zusätzliche Komponenten: ein Programmlizenzierungsmodul **500** und ein Betriebsmittelbibliotheks-Lizenzierungsmodul **510**. Diese Module sind in **Fig.** 6 näher dargestellt.

[0036] **Fig.** 6 zeigt das Programmlizenzierungsmodul **500** und das Betriebsmittelbibliotheks-Lizenzierungsmodul **510** in einer Ausführungsform der vorliegenden Erfindung. Wie in **Fig.** 6 gezeigt, enthält das Programmlizenzierungsmodul **500** den Lizenztextstring **600** und den Lizenzschlüssel **610**. Der Lizenztextstring **600** enthält Daten, die die Bedingungen des Software-Lizenzvertrags angeben, unter denen der Betriebsmittelbibliothek-Verkäufer das das Programmlizenzierungsmodul **510** enthaltende Programm lizenziert hat, um die Betriebsmittelbibliothek des Verkäufers zu nutzen. Beispielsweise kann der Lizenztextstring **600** den folgenden Text aufweisen:

Tabelle 1: Beispiel eines Lizenztextstrings

[0037] „© Urheberrecht **1997**. Das Programm A der Resource Library Vendor, Inc. ist für die Nutzung der Betriebsmittelbibliothek D lizenziert. Kein Ablaufdatum. Es ist rechtlich unzulässig, diese Lizenz zu kopieren oder auf ein anderes Programm zu übertragen."

[0038] In dem in Tabelle 1 gezeigten Beispiel gibt der Lizenztextstring **600** den Namen des Betriebsmittelbibliothek-Verkäufers („Resource Library Vendor, Inc."), den Namen des Programms, das für die Nutzung der Betriebsmittelbibliothek lizenziert ist („Programm A"), und den Namen der Betriebsmittelbibliothek an, die lizenziert worden ist („Betriebsmittelbibliothek D"). Der Lizenztextstring **600** gibt auch an, dass die Lizenz „Kein Ablaufdatum" hat.

[0039] Der Lizenzschlüssel **610** wird algorithmisch vom Lizenztextstring **600** abgeleitet. Bei einer Ausführungsform weist der Lizenzschlüssel **610** die digitale Signatur des Betriebsmittelbibliothek-Verkäufers auf.

[0040] Eine digitale Signatur ist ein Mechanismus, der entwickelt worden ist, um dazu beizutragen, die Unversehrtheit einer elektronischen Nachricht zu gewährleisten. Eine digitale Signatur dient dazu, die Authentizität einer elektronischen Nachricht zu überprüfen und festzustellen, ob eine elektronische Nachricht geändert worden ist.

[0041] Eine Form der digitalen Signatur verwendet eine Nachrichtenzusammenfassung (Digest). Eine Nachrichtenzusammenfassung ist ein Wert, der erzeugt wird, wenn eine elektronische Nachricht ein Einweg-Codie-

rungsverfahren („Zusammenfassungsverfahren"), wie etwa eine Hash-Routine, durchläuft. Ein ideales Zusammenfassungsverfahren ist ein Zusammenfassungsverfahren, für das die Wahrscheinlichkeit, dass zwei verschiedene elektronische Nachrichten die gleiche Nachrichtenzusammenfassung erzeugen, fast Null ist.

[0042] Bei dieser Form der digitalen Signatur müssen der Absender und der Empfänger wissen, welches Zusammenfassungsverfahren gerade verwendet wird. Der Absender erzeugt die elektronische Nachricht und erzeugt eine Nachrichtenzusammenfassung, indem er die elektronische Nachricht das Zusammenfassungsverfahren durchlaufen lässt. Der Absender signiert digital die resultierende Nachrichtenzusammenfassung, indem er beispielsweise eine algorithmische Operation an der Nachrichtenzusammenfassung unter Verwendung des privaten Schlüssels des Absenders ausführt. Anstatt eine Nachrichtenzusammenfassung zu erzeugen und die Nachrichtenzusammenfassung zu signieren, kann ein Absender alternativ die Nachricht selbst signieren.

[0043] Um die Authentizität der digital signierten Nachricht zu überprüfen, erhält der Empfänger die elektronische Nachricht und die digitale Signatur des Absenders. Der Empfänger überprüft die digitale Signatur mit einem entsprechenden Überprüfungsverfahren. Bei einer Ausführungsform beispielsweise überprüft der Empfänger die digitale Signatur, indem er ein algorithmisches Verfahren an der digitalen Signatur unter Verwendung des öffentlichen Schlüssels des Absenders durchführt. Das Überprüfungsverfahren bestätigt, dass (1) die elektronische Nachricht vom Absender digital signiert wurde und (2) der Inhalt der elektronischen Nachricht ab dem Zeitpunkt, zu dem sie signiert wurde, bis zu dem Zeitpunkt, zu dem die digitale Signatur überprüft wurde, nicht geändert wurde.

[0044] Bei der vorliegenden Ausführungsform der Erfindung ist die „Nachricht", die digital signiert wird, der Lizenztextstring **600**. Der Unterzeichner ist der Betriebsmittelbibliothek-Verkäufer. Das Ergebnis ist der Lizenzschlüssel **610**.

[0045] Mit dem Lizenztextstring **600** und dem Lizenzschlüssel **610** überprüft das Betriebsmittelbibliotheks-Lizenzierungsmodul **510**, dass ein anforderndes Programm für die Nutzung der Betriebsmittelbibliothek lizenziert worden ist. Wie in **Fig.** 6 gezeigt, weist das Betriebsmittelbibliotheks-Lizenzierungsmodul **510** ein Lizenzüberprüfungsmodul **620** auf. Wenn ein Programm den Zugriff auf die Betriebsmittelbibliothek anfordert, liest das Betriebsmittelbibliotheks-Lizenzierungsmodul **510** den Lizenztextstring **600** und den Lizenzschlüssel **610** aus dem anfordernden Programm. Bei einer Ausführungsform werden der Lizenztextstring **600** und der Lizenzschlüssel **610** vom anfordernden Programm zusammen mit einer Anforderung für den Zugriff auf die Betriebsmittelbibliothek an die Betriebsmittelbibliothek gesendet. Bei einer anderen Ausführungsform liest das Betriebsmittelbibliotheks-Lizenzierungsmodul **510** den Lizenztextstring **600** und den Lizenzschlüssel **610** aus einem Konstantendefinitionsabschnitt des anfordernden Programms.

[0046] Das Betriebsmittelbibliotheks-Lizenzierungsmodul **510** verwendet den Lizenzschlüssel **610** zur Überprüfung des Inhalts des Lizenztextstrings **600** in der gleichen Weise, wie eine digitale Signatur zur Überprüfung einer elektronischen Nachricht verwendet wird. Unter Verwendung des Lizenzüberprüfungsmoduls **620** überprüft das Betriebsmittelbibliotheks-Lizenzierungsmodul **510**, dass der Lizenztextstring **600** authentisch ist (d. h. vom Betriebsmittelbibliothek-Verkäufer erzeugt wurde) und nicht geändert wurde. Wenn das Überprüfungsverfahren nicht erfolgreich ist, was darauf hinweist, dass die digitale Signatur nicht in Ordnung ist, lehnt das Betriebsmittelbibliotheks-Lizenzierungsmodul **510** die Anforderung des anfordernden Programms für den Zugriff auf die Betriebsmittelbibliothek ab. Wenn das Überprüfungsverfahren erfolgreich ist, überprüft das Betriebsmittelbibliotheks-Lizenzierungsmodul **510** die Lizenz, um im Lizenztextstring **600** enthaltene Lizenzbeschränkungen festzustellen.

[0047] Der vorstehend in Tabelle 1 angegebene exemplarische Lizenztextstring **600** identifiziert das „Programm A" als das Programm, das für die Nutzung der Betriebsmittelbibliothek lizenziert ist, und gibt an, dass die Lizenz „Kein Ablaufdatum" hat. Das Betriebsmittelbibliotheks-Lizenzierungsmodul **510** erhält den Namen „Programm A" aus dem Lizenztextstring **600** und kontrolliert, ob das anfordernde Programm das Programm A ist. Wenn das anfordernde Programm nicht das Programm A ist, wird der Zugriff auf die Betriebsmittelbibliothek verwehrt.

[0048] Anstatt wie in dem vorliegenden Beispiel „Kein Ablaufdatum" anzugeben, kann der Lizenztextstring **600** ein Ablaufdatum und/oder ein Anfangsdatum für die Lizenz angeben. Wenn solche Daten im Lizenztextstring **600** angegeben sind, überprüft das Betriebsmittelbibliotheks-Lizenzierungsmodul **510**, dass das aktuelle Datum in die Laufzeit der Lizenz fällt, bevor es den Zugriff auf die Betriebsmittelbibliothek gewährt. Wenn das aktuelle Datum nicht in die Laufzeit der Lizenz fällt, wird dem anfordernden Programm der Zugriff auf die Betriebsmittelbibliothek verwehrt.

Zugriffsverfahren

[0049] Das von einer Betriebsmittelbibliothek verwendete Verfahren, einem anfordernden Programm bei einer Ausführungsform der Erfindung den Zugriff zu gewähren oder zu verwehren, ist in **Fig.** 7 dargestellt. Bei einer Ausführungsform wird dieses Verfahren zum ersten Mal durchgeführt, wenn ein Programm den Zugriff auf eine Betriebsmittelbibliothek anfordert. Bei einer anderen Ausführungsform wird das Verfahren immer dann

durchgeführt, wenn die Betriebsmittelbibliothek eine Anforderung für den Zugriff empfängt.

[0050] Wie in **Fig.** 7 gezeigt, beginnt das Verfahren damit, dass im Schritt **700** ein anforderndes Programm die Benutzung der Betriebsmittelbibliothek anfordert. Im Schritt **705** erhält die Betriebsmittelbibliothek den Lizenztext und den Lizenzschlüssel des anfordernden Programms. Der Lizenztext und der Lizenzschlüssel können beispielsweise in der Anforderung enthalten sein, oder die Betriebsmittelbibliothek kann den Lizenztext und den Lizenzschlüssel aus einem Konstantendeklarationsbereich des anfordernden Programms lesen, oder die Betriebsmittelbibliothek kann den Lizenztext und den Lizenzschlüssel mit anderen Mitteln erhalten.

[0051] Nachdem die Betriebsmittelbibliothek den Lizenztext und den Lizenzschlüssel erhalten hat, überprüft sie im Schritt **710** die Authentizität des Lizenztextes mit Hilfe des Lizenzschlüssels. Im Schritt **725** stellt die Betriebsmittelbibliothek fest, ob die Überprüfung erfolgreich ist. Wenn die Authentizität des Lizenztextes nicht bestätigt wird, wird im Schritt **730** der Zugriff auf die Betriebsmittelbibliothek verwehrt.

[0052] Wenn die Überprüfung der Authentizität des Lizenztextes erfolgreich ist, prüft die Betriebsmittelbibliothek im Schritt **735** die im Lizenztext enthaltenen Lizenzbedingungen. Im Schritt **740** stellt die Betriebsmittelbibliothek fest, ob im Lizenztext eine begrenzte Laufzeit angegeben ist. Wenn keine Laufzeit angegeben ist, geht das Verfahren zum Schritt **755**. Wenn eine Laufzeit angegeben ist, prüft die Betriebsmittelbibliothek im Schritt **745**, ob die Laufzeit abgelaufen ist. Die Laufzeit ist abgelaufen, wenn das aktuelle Datum vor einem im Lizenztext angegebenen Anfangsdatum liegt oder wenn das aktuelle Datum nach einem im Lizenztext angegebenen Ablaufdatum liegt. Wenn die Laufzeit abgelaufen ist, wird im Schritt **750** der Zugriff auf die Betriebsmittelbibliothek verwehrt.

[0053] Wenn die Laufzeit nicht abgelaufen ist, geht die Verarbeitung zum Schritt **755** weiter. Im Schritt **755** stellt die Betriebsmittelbibliothek fest, ob das anfordernde Programm das Programm ist, das im Lizenztext angegeben ist. Wenn das anfordernde Programm nicht das im Lizenztext angegebene Programm ist, wird im Schritt **760** der Zugriff auf die Betriebsmittelbibliothek verwehrt. Wenn das anfordernde Programm das im Lizenztext angegebene Programm ist, prüft im Schritt **765** die Betriebsmittelbibliothek, ob es weitere Lizenzbedingungen gibt, die im Lizenztext enthalten sind. Wenn es keine weiteren Lizenzbedingungen gibt, wird im Schritt **770** der Zugriff auf die Betriebsmittelbibliothek gewährt. Wenn es weitere Lizenzbedingungen gibt, prüft die Betriebsmittelbibliothek im Schritt **775**, ob diese Bedingungen erfüllt sind. Wenn die Bedingungen nicht erfüllt sind, wird im Schritt 780 der Zugriff auf die Betriebsmittelbibliothek verwehrt. Wenn die Bedingungen erfüllt sind, wird im Schritt **785** der Zugriff auf die Betriebsmittelbibliothek gewährt.

[0054] Die Erfindung kann in der Sprache Objective C realisiert werden. Objective C ist im Wesentlichen die Sprache ANSI C mit Objekt-Messaging-Erweiterungen. Eine vollständige Beschreibung der Sprache Objective C erscheint in „Object-Oriented Programming and the Objective-C Language" („Objektorientierte Programmierung und die Sprache Objective C"), herausgegeben von Addison-Wesley (ISBN 0-201-63251-9) (1993). Die Erfindung kann jedoch auch in einer anderen geeigneten Computer-Programmiersprache realisiert werden.

[0055] Wie vorstehend beschrieben, kann die Erfindung durch Einbetten entsprechender Segmente des Programmcodes in den Quellcode eines Programms, das eine Betriebsmittelbibliothek nutzt, und in den Quellcode der Betriebsmittelbibliothek selbst realisiert werden. Die Betriebsmittelbibliothek ist so kompiliert, dass sie eine ausführbare Implementierung erzeugt, die mit einer kompilierten und ausführbaren Version des Programms verknüpft werden kann.

Anwendungsprogramm-Schnittstelle (API)

[0056] Bei einer Ausführungsform der Erfindung ist die Betriebsmittelbibliothek eine Anwendungsprogramm-Schnittstelle (application program interface/„API"). Eine API hat drei Hauptfunktionen: Sie empfängt Anforderungen von einem Anwendungsprogramm, um Grundoperationen, wie Empfangen von Benutzer-Eingangssignalen oder Anzeigen von Ausgangssignalen, durchzuführen; sie wandelt jede Anforderung in eine Form um, die von dem jeweiligen Betriebssystem, das gerade in Gebrauch ist, verstanden werden kann; und sie empfängt Antworten und Ergebnisse vom Betriebssystem, formatiert sie einheitlich und sendet sie an das Anwendungsprogramm zurück.

[0057] APIs werden in der Regel in einer ausführbaren Implementierung hergestellt, die speziell für das zugrundeliegende Betriebssystem kompiliert wird. Das ist notwendig, da unterschiedliche Betriebssysteme unterschiedliche Aufrufmechanismen und Kommunikationsverfahren für solche einfachen Operationen wie Lesen aus einem und Schreiben in einen Massenspeicher haben. Beispielsweise kann eine API eine „Draw(x,y)"-Funktion haben, die von einem Anwendungsprogramm aufgerufen werden kann, um einen Punkt mit den Koordinaten (x,y) auf dem Bildschirm eines Computersystems zu zeichnen. Nach Erhalt einer Draw(x,y)-Anforderung von einem Anwendungsprogramm wandelt die API die Anforderung in einen Befehls- oder Funktionsaufruf um, der für das Betriebssystem bestimmt ist, das gerade in Gebrauch ist. Beispielsweise könnte die API die Draw(x,y)-Anforderung in eine Reihe von Maschinenbefehlen umwandeln, um Register mit x,y-Werten zu laden und eine Betriebssystemfunktion aufzurufen oder einen Interrupt zu erzeugen. Die Person, die das Anwendungsprogramm schreibt, braucht sich um solche Einzelheiten nicht zu kümmern.

[0058] In einigen Fälle verweist die API auf Funktionen oder ruft Funktionen auf, die in einer externen Funktionsbibliothek lokalisiert sind, wie etwa eine Gruppe von Gerätetreibern, anstatt das Betriebssystem direkt aufzurufen. Gerätetreiber sind kleine ausführbare Programme, die es dem Betriebssystem ermöglichen, bestimmte Hardware-Geräte, wie Grafikkarten und Drucker, zu adressieren und mit ihnen zu arbeiten. Gerätetreiber stellen ebenfalls eine Form der Betriebsmittelbibliothek dar.

[0059] In Abhängigkeit vom Betriebssystem kann die API in einem von mehreren ausführbaren Formaten, wie Laufzeitbibliothek, Geräte-verknüpfte Bibliothek (device linked library/DLL) oder einer anderen ausführbaren Datei, hergestellt werden. Die API wird dem Endbenutzer in einer von diesen Objektcodeversionen oder „Implementierungen" der API zur Verfügung gestellt. Im Branchengebrauch kann sich der Begriff API auf eine Definition oder Festlegung von Funktionen in der API, auf den Quellcode der API, die diese Funktionen implementiert, oder auf die ausführbare Version dieses Quellcodes beziehen, die schließlich an die Endbenutzer verteilt und von diesen benutzt wird. Beispiele für APIs sind die OpenStep-API, die von NeXT Software, Inc., Redwood City, Kalifornien, lieferbar ist, und die Visual-Basic-DLL, die von Microsoft Corporation, Redmond, Washington, erhältlich ist.

[0060] Der hier benutzte Begriff API umfasst auch die Programmiersprache Java. Anstatt sie in ausführbarer Form zu vertreiben, werden Java-Programme als Pakete von „Bytecodes" vertrieben. Die Bytecodes werden mit einer virtuellen Java-Maschine (Java Virtual Machine, JVM), die in dem Computer resident ist, auf dem das Java-Programm läuft, in der Laufzeit zu einem ausführbaren Code kompiliert. Für unterschiedliche Rechnerprozessoren und Betriebssysteme werden unterschiedliche JVMs verwendet. Alle JVMs lesen jedoch den gleichen Bytecode.

[0061] Daher sind Java-Bytecode-Programme und -Pakete Plattform-unabhängig. Java-Bytecode-Programme und -Pakete brauchen nur in einer Form geschrieben zu werden. Die JVMs achten darauf, den Bytecode an unterschiedliche Rechnerplattformen anzupassen. Java-Bytecode-Pakete können von verschiedenen Java-Programmen verwendet werden und stellen an sich Betriebsmittelbibliotheken dar.

[0062] In der Regel kann der Endbenutzer die ausführbare Version der API-Implementierung getrennt von einem bestimmten Anwendungsprogramm von dessen Erzeuger oder Verkäufer erwerben, oder der Endbenutzer kann die API-Implementierung als Paket mit einem Anwendungsprogramm kaufen, das die API zum Laufen braucht und verwendet.

[0063] In jedem Fall wird die API-Implementierung in ausführbarer Form im Computersystem des Endbenutzers installiert (normalerweise durch Kopieren der API-Implementierung in einen Massenspeicher, wie etwa eine Festplatte). Nach der Installation der API-Implementierung kann der Endbenutzer ein Anwendungsprogramm, das die API-Implementierung verwendet, starten (kann er mit der Abarbeitung des Programms beginnen). Das Anwendungsprogramm lokalisiert die API-Implementierung auf der Festplatte und referenziert die API-Implementierung, ruft sie auf oder wird mit ihr verknüpft. Wenn bei der Abarbeitung das Anwendungsprogramm eine Operation ausführen muss, die in der API-Implementierung implementiert ist, wie etwa Zeichnen einer Linie auf dem Bildschirm, ruft das Anwendungsprogramm die entsprechende Funktion in der API-Implementierung auf. Die entsprechende Funktion teilt dann dem Betriebssystem (oder den Geräte-unabhängigen Fenstertechnik-Funktionen oder einem anderen Gerätetreiber) mit, wie die gewünschte Operation auszuführen ist.

[0064] Ein wichtiger Vorteil der Verwendung von APIs besteht darin, dass ein Anwendungsprogramm, wie etwa ein Textverarbeitungsprogramm, so geschrieben werden kann, dass es nur mit der API und nicht mit dem Betriebssystem kommuniziert. Ein solches Anwendungsprogramm kann zu einem anderen Betriebssystem verschoben oder portiert werden, ohne den Programm-Quellcode zu modifizieren. Daher heißt es, dass Anwendungsprogramme, die für APIs geschrieben sind, Betriebssystem-unabhängig sind, was bedeutet, dass der Anwendungsprogramm-Quellcode ohne Modifikation zu einem anderen Computersystem mit einem anderen Betriebssystem verschoben werden kann und neukompiliert und mit einer für dieses Betriebssystem erstellten API-Implementierung verknüpft werden kann. Die Fähigkeit, den unmodifizierten Anwendungsprogramm-Quellcode zu anderen Betriebssystemen zu verschieben, ist ein wichtiger Vorzug der Verwendung von APIs.

[0065] Vom Standpunkt von API-Verkäufern aus haben APIs jedoch auch den wesentlichen Nachteil, dass ein Endbenutzer nur eine Kopie der API braucht, um mehrere Anwendungsprogramme laufen zu lassen, die mit der API kompatibel sind. Da die API generische Eingabe-, Ausgabe- und Verarbeitungsfunktionen bereitstellt, arbeitet sie mit vielen verschiedenen Endbenutzer-Anwendungsprogrammen. Einige Software-Verkäufer wollen die Verwendung ihrer API-Implementierungen auf eine Anwendung beschränken oder wollen den Endbenutzer auffordern, einen Schlüssel für die API für jede vom Endbenutzer erworbene Anwendung zu kaufen, sodass der Endbenutzer eine andere oder höhere Gebühr für die Verwendung weiterer Anwendungsprogramme zahlt.

[0066] Die vorliegende Erfindung bietet eine Möglichkeit, eine Betriebsmittelbibliothek so anzuordnen, dass eine API nur mit bestimmten zugelassenen Anwendungs- oder anderen Endbenutzer-Programmen funktioniert.

API-Lizenz-Ausführungsform

[0067] Wie auf dem Fachgebiet bekannt, kann der Quellcode eines Computerprogramms in mehrere Komponenten unterteilt werden, die einen Variablendeklarationsbereich, einen Konstantendeklarationsbereich und einen Prozedurvereinbarungsbereich umfassen. **Fig.** 9 zeigt eine Ausführungsform der vorliegenden Erfindung, die mit einer API verwendet wird. Wie in **Fig.** 9 gezeigt, hat bei dieser Ausführungsform das Anwendungsprogramm **900** eine LicenseKeyString-Konstante **902** und eine LicenseAgreementString-Konstante **904** im Konstantendeklarationsbereich **901** des Quellcodes des Anwendungsprogramms. Bei der Ausführungsform von **Fig.** 9 sind der LicenseKeyString **902** und der LicenseAgreementString **904** als globale String-Konstanten deklariert.

[0068] Bei einer Ausführungsform enthält der LicenseAgreementString **904** einen vom Verkäufer der API erstellten Textstring, der in einem für den Menschen lesbaren Text die Lizenzbeschränkungen für die Verwendung der für das Anwendungsprogramm verwendbaren API beschreibt. Beispielsweise kann der LicenseAgreementString wie folgt lauten: „Diese API ist für die eigene interne Nutzung nur für die gleichzeitige Verwendung nur mit dem Textverarbeitungs-Anwendungsprogramms lizenziert." Der spezielle Text des LicenseAgreementStrings wird vom Lizenzgeber der API erstellt. Der Text kann eine beliebige Kombination aus Wörtern, Symbolen oder Zahlen sein.

[0069] Der LicenseKeyString **904** enthält einen Schlüssel, der dem LicenseAgreementString **902** entspricht und auf diesem basiert. Beispielsweise kann der LicenseKeyString eine digitale Signatur des LicenseAgreementStrings sein, die dadurch erstellt wird, dass der LicenseAgreementString und ein privater Schlüssel des API-Verkäufers einem Digitalsignatur-Verfahren unterzogen werden. Das genaue Verfahren zur Erzeugung des LicenseKeyStrings ist nicht entscheidend, wenn nur der Lizenzgeber der API einen eindeutigen LicenseKeyString, der dem LicenseAgreementString entspricht, erzeugen kann. Die Werte der beiden Strings werden vom Verkäufer der API erzeugt und der Person oder Firma zur Verfügung gestellt, die gerade das Endbenutzer-Anwendungsprogramm entwickelt (beispielsweise kann der API-Verkäufer die beiden String-Werte per E-Mail an den Anwendungsprogramm-Entwickler senden). Der Anwendungsprogramm-Entwickler wird vom API-Verkäufer angewiesen, die String-Deklarationen in den Quellcode des Endbenutzer-Anwendungsprogramms des Entwicklers einzufügen. Die beiden Werte können öffentlich sein, sodass der API-Verkäufer oder der Entwickler die Werte nicht vor Benutzern des Endbenutzer-Anwendungsprogramms geheim halten oder verbergen muss. Die beiden Strings werden in die ausführbare Form (oder bei Java zu Bytecode-Paketen) des Anwendungsprogramms kompiliert. Dadurch werden der LicenseKeyString und der LicenseAgreementString in den ausführbaren Code (oder Bytecode) des Anwendungsprogramms eingebunden.

[0070] Wie außerdem in **Fig.** 9 gezeigt, hat die API **920** eine UNLOCK-Funktion **923** und eine CHECK-LICENSE-Funktion **921**, um zu prüfen, ob der LicenseKeyString mit dem LicenseAgreementString zusammenpasst. Bei der Ausführungsform von **Fig.** 9 weist die CHECK-LICENSE-Funktion **921** eine Teilfunktion CHECK **922** auf.

API-Verfahren

[0071] **Fig.** 10 zeigt ein Ablaufdiagramm der Verarbeitungsschritte der UNLOCK-Funktion **923**. Das Verfahren von **Fig.** 10 kann beispielsweise während der Laufzeit durchgeführt werden, wenn das Anwendungsprogramm und die API kompiliert, verknüpft und abgearbeitet werden.

[0072] Die UNLOCK-Funktion wird von der API nach Initialisierung der API aufgerufen, beispielsweise nachdem sie vom Anwendungsprogramm **900** oder einer anderen Aufruffunktion oder einem anderen Objekt oder Programm (der „aufrufenden Einheit") aufgerufen worden ist. Die Verarbeitung beginnt im Schritt **1002**. Die UNLOCK-Funktion prüft zunächst, ob die API eine Standortlizenz erhalten hat, mit der die API mit einer aufrufenden Einheit auf dem Computer verwendet werden kann, auf dem sie installiert worden ist. Bei dieser Ausführungsform wird die Standortlizenz durch Hinzufügen eines entsprechenden LicenseKeyStrings und LicenseAgreementStrings zu der API beim Installieren der API angezeigt. Dieses Verfahren wird nachstehend näher beschrieben. Ein entsprechender LicenseAgreementString kann beispielsweise lauten: „API-Standortlizenz ist erteilt. Diese API darf mit einem Anwendungsprogramm an dem Standort verwendet werden, an dem sie installiert ist." Der entsprechende geeignete LicenseKeyString kann beispielsweise durch Verwenden des privaten Schlüssels des API-Verkäufers und eines Digitalsignatur-Verfahrens für den LicenseAgreementString abgeleitet werden.

[0073] Der Prozess der Kontrolle der Standortlizenz beginnt im Schritt **1004**, wo die UNLOCK-Funktion einen LicenseKeyString und einen LicenseAgreementString aus der API lokalisiert und gewinnt (insofern sie für die API bereitgestellt worden sind). Die Steuerung geht dann zum Schritt **1006**, wo die Funktion prüft, ob die API nach einer Standortlizenz für unbeschränkte Verwendung mit einem Anwendungsprogramm lizenziert ist. Die Prüfung von Schritt **1006** erfolgt durch Überprüfen der Authentizität des aus der API gewonnenen LicenseKeyStrings und LicenseAgreementStrings, und wenn sie authentisch sind, Feststellen, ob der LicenseAgree-

mentString angibt, dass eine Standortlizenz erteilt worden ist.

[0074] Die Authentizität des LicenseAgreementStrings und des LicenseKeyStrings wird ermittelt, indem der LicenseAgreementString, der LicenseKeyString und eine Kopie des bei der API-Implementierung gespeicherten öffentlichen Schlüssels des API-Verkäufers dem CHECK-Verfahren **922** unterzogen werden. Das CHECK-Verfahren **922** verwendet ein Digitalsignatur-Authentisierungsverfahren ("DSA"-Verfahren), um die Authentizität des LicenseAgreementStrings zu überprüfen.

[0075] Das vom CHECK-Verfahren **922** verwendete DSA-Verfahren kann jedes Digitalsignatur-Authentisierungsverfahren sein, das in der Lage ist, einen Eingabe-String und einen Schlüssel, der die digitale Signatur des Eingabe-Strings darstellen soll, zu lesen, ein entsprechendes Authentisierungsverfahren anzuwenden und die Gültigkeit des Eingabe-Strings durch Prüfen, ob der Schlüssel die digitale Signatur des Eingabe-Strings darstellt, festzustellen. Ein exemplarisches DSA-Verfahren ist beispielsweise in der US-Patentanmeldung Nr. 08/484.264 mit dem Titel „Method and Apparatus for Digital Signature Authentication" („Verfahren und Vorrichtung zur Digitalsignatur-Authentisierung"), das an den Abtretungsempfänger des vorliegenden Patents übertragen worden ist und jetzt als US-PS 5.581.616 erteilt ist. Das DSA-Verfahren der RSA Data Security, Inc. kann ebenfalls für die Verwendung mit der Erfindung abgeändert werden. Zur Erhöhung der Durchführungsgeschwindigkeit des CHECK-Verfahrens kann ein Persessions-Cache verwendet werden.

[0076] Wenn festgestellt wird, dass der LicenseKeyString die gültige digitale API-Verkäufer-Signatur des LicenseAgreementStrings ist, wird der LicenseAgreementString geprüft, um festzustellen, ob er angibt, dass eine Standortlizenz erteilt worden ist. Wenn der LicenseAgreementString dies angibt, ist die Prüfung von Schritt **1006** erfolgreich und die Steuerung geht zum Schritt **1014**. An dieser Stelle sendet die UNLOCK-Funktion ein positives Ergebnis an die aufrufende Einheit zurück und erlaubt der aufrufenden Einheit, die API zu verwenden.

[0077] Wenn die Prüfung von Schritt **1006** keinen Erfolg hat, geht die Steuerung zum Schritt **1008**, wo die UNLOCK-Funktion den LicenseKeyString und den LicenseAgreementString aus einem Datensegment (beispielsweise dem kompilierten Konstantendeklarationsbereich) der aufrufenden Einheit gewinnt und liest. Alternativ kann die aufrufende Einheit den LicenseKeyString und den LicenseAgreementString an die API senden. Nach Erhalt des LicenseKeyStrings und LicenseAgreementStrings der aufrufenden Einheit geht die Steuerung zum Schritt **1010**, wo die Funktion prüft, ob die aufrufende Einheit für die Verwendung der API lizenziert ist. Diese Prüfung umfasst zwei Teile. Der eine Teil, der das vorstehend beschriebene CHECK-Verfahren **922** verwendet, stellt fest, ob der LicenseAgreementString ein LicenseAgreementString ist, der vom API-Verkäufer berechtigterweise ausgegeben wurde. Ein zweiter Teil prüft den LicenseAgreementString auf die Bedingungen der enthaltenen Lizenz und stellt fest, ob diese Bedingungen erfüllt sind. Wenn das Ergebnis positiv ist, geht die Steuerung zum Schritt **1014**. An dieser Stelle wird die Verwendung der API mit der aufrufenden Einheit zugelassen und die API gibt die Steuerung zur aufrufenden Einheit zurück, sodass die aufrufende Einheit mit der normalen Abarbeitung fortfährt.

[0078] Wenn das Ergebnis negativ ist, erhält die aufrufende Einheit keine Genehmigung zur Verwendung der API und die Steuerung geht zum Schritt **1012**. Im Schritt **1012** erzeugt die API eine Fehlermeldung, wie „API ist nicht für die Verwendung mit diesem Anwendungsprogramm lizenziert", und lehnt den Zugriff auf die aufrufende Einheit ab.

[0079] Die Schritte **1006** und **1010** führen Lizenzprüfungen durch, indem sie die in den **Fig.** 9 und 11 gezeigte CHECK-LICENSE-Funktion **921** aufrufen. Die Verarbeitungsschritte der CHECK-LICENSE-Funktion **921** sind in **Fig.** 11 dargestellt.

[0080] Der Prozessablauf der CHECK-LICENSE-Funktion beginnt im Schritt **1102**. Die Steuerung geht zum Schritt **1104**, wo die CHECK-LICENSE-Funktion in Vorbereitung auf den Aufruf der CHECK-Funktion **922** den LicenseKeyString **902**, den LicenseAgreementString **904** und eine Kopie des öffentlichen Schlüssels **1106** des API-Verkäufers als Funktionsaufruf-Argumente assembliert. Wie nachstehend näher dargelegt wird, wird der öffentliche Schlüssel **1106** vom API-Verkäufer aufgrund eines geheimen privaten Schlüssels erstellt. Die drei Argumente werden an die CHECK-Funktion **1108** weitergeleitet.

[0081] Wenn die CHECK-Funktion (die später näher beschrieben wird) einen FAIL- oder Falsch-Zustand zurücksendet, geht die Steuerung zum Schritt **1124** und die CHECK-LICENSE-Funktion sendet selbst einen Falsch-Zustand zurück. Wenn die CHECK-Funktion einen PASS- oder Richtig-Zustand zurücksendet, geht die Steuerung zum Schritt **1112**, wo die CHECK-LICENSE-Funktion die Bedingungen der im LicenseAgreementString festgelegten Lizenz prüft. Im Schritt **1114** prüft die CHECK-LICENSE-Funktion, ob der Name der aufrufenden Einheit der Gleiche wie der Name der lizenzierten Einheit ist, die im LicenseAgreementString angegeben ist. Wenn der Name der aufrufenden Einheit fehlerhaft ist, geht die Steuerung zum Schritt **1124**, wo eine Falsch-Nachricht an die UNLOCK-Funktion gesendet wird.

[0082] Wenn der Name der aufrufenden Einheit richtig ist, prüft die CHECK-LICENSE-Funktion im Schritt **1116**, ob der LicenseAgreementString ein Ablaufdatum enthält. Ein Ablaufdatum kann vom API-Verkäufer in den LicenseAgreementString eingefügt werden, um einen Ablaufzeitpunkt festzulegen, nach dem die Verwendung der API durch die aufrufende Einheit nicht mehr zugelassen wird. Die CHECK-LICENSE-Funktion kann

beispielsweise auf ein Ablaufdatum prüfen, indem sie einen Textstring sucht, der ein Ablaufdatum, beispielsweise „Ablaufdatum" oder „Gültig bis", angibt.

[0083] Wenn die Prüfung von Schritt **1116** positiv ist, geht die Steuerung zum Schritt **1118**, wo die CHECK-LICENSE-Funktion prüft, ob das aktuelle Datum, das beispielsweise von einer Computer-Uhr oder einem -Betriebssystem aufrechterhalten wird, größer als das im LicenseAgreementString gefundene Ablaufdatum ist. Wenn die Prüfung von Schritt **1118** bestanden ist, geht die Steuerung zum Schritt **1120**. Wenn die Prüfung von Schritt **1118** nicht bestanden ist, sendet die CHECK-LICENSE-Funktion im Block **1124** eine FAIL-Nachricht zurück.

[0084] Im Schritt **1120** prüft die CHECK-LICENSE-Funktion, ob der LicenseAgreementString weitere Lizenzbedingungen angibt. Wenn es keine weiteren Bedingungen gibt, sendet die CHECK-LICENSE-Funktion im Block **1126** eine PASS-Nachricht zurück. Wenn es weitere Bedingungen gibt, stellt die CHECK-LICENSE-Funktion im Block **1122** fest, ob diese Bedingungen erfüllt sind. Wenn eine der zusätzlichen Bedingungen nicht erfüllt ist, sendet die CHECK-LICENSE-Funktion im Block **1124** eine FAIL-Nachricht zurück. Wenn alle zusätzlichen Bedingungen erfüllt sind, sendet die CHECK-LICENSE-Funktion im Block **1126** eine PASS-Nachricht zurück.

[0085] Die Funktionsweise der CHECK-Funktion, die von der CHECK-LICENSE-Funktion im Block **1108** aufgerufen wird, ist in **Fig.** 12 dargestellt. Wie in **Fig.** 12 gezeigt, besteht der Zweck der CHECK-Funktion darin, die Authentizität eines Lizenzvertragsstrings zu überprüfen, indem sie überprüft, dass ein entsprechender Lizenzschlüsselstring eine gültige digitale Signatur des Lizenzvertragsstrings darstellt. Die CHECK-Funktion beginnt im Schritt **1202** und empfängt im Schritt **1203** als Eingangssignal einen LicenseKeyString, einen LicenseAgreementString und einen öffentlichen Schlüssel des Verkäufers. Der öffentliche Schlüssel wird vom Betriebsmittelbibliothek-Verkäufer mit einem bekannten Verfahren zur Erzeugung eines öffentlichen/privaten Schlüsselpaars erzeugt, wie es auf dem Gebiet der Kryptographie bekannt ist. Die Schlüsselerzeugung kann beispielsweise durch Schnelle Elliptische Verschlüsselung (Fast E1liptical Encryption; FEE) erfolgen, oder es kann die Diffie-Hellman-Schlüsselerzeugung genutzt werden.

[0086] Im Schritt **1204** überprüft die CHECK-Funktion, dass der LicenseKeyString die digitale Signatur des LicenseAgreementStrings aufweist. Im Schritt **1208** prüft die CHECK-Funktion, ob die Überprüfung von Schritt **1204** erfolgreich überprüft hat, dass der LicenseKeyString die digitale Signatur des LicenseAgreementStrings aufweist. Wenn ja, ist der LicenseAgreementString gültig und die CHECK-Funktion sendet einen Booleschen wahren oder Richtig-Wert zurück. Wenn nicht, ist der LicenseAgreementString ungültig und die CHECK-Funktion sendet eine Falsch- oder Negativ-Nachricht zurück.

[0087] Da der LicenseKeyString der vorliegenden Ausführungsform die digitale Signatur des LicenseAgreementStrings aufweist, kann der LicenseAgreementString nicht in irgendeiner Weise geändert werden, ohne dass die Änderung erkannt wird. Da, allgemeiner gesagt, der Identifikator (z. B. der LicenseKeyString) der Erfindung ein eindeutiger Schlüssel ist, der mathematisch von einem bestimmten Textstring abgeleitet wurde, der die Lizenzbedingungen für ein bestimmtes Endbenutzer-Programm (z. B. den LicenseAgreementString) festlegt, kann der Identifikator zum Feststellen von Änderungen an den Lizenzbedingungen verwendet werden. Dadurch wird eine unbefugte Modifikation des Textstrings, von einer Verlängerung der Benutzung einer Betriebsmittelbibliothek bis hin zu einem unlizenzierten Programm, vermieden. Wenn beispielsweise ein Endbenutzer das Ablaufdatum mit einem Debugger oder Maschinenspracheneditor zu ändern versucht, passt der Identifikator nicht mehr mit dem Lizenztextstring zusammen. Ohne den privaten Schlüssel des Verkäufers zu kennen, kann der Endbenutzer keinen passenden Identifikator erzeugen.

[0088] Wenn ein privater 127-Bit-Schlüssel vom Verkäufer benutzt wird, um den in der vorliegenden Erfindung verwendeten Identifikator zu erzeugen, müsste ein fest entschlossener Hacker, der den privaten Schlüssel zu fälschen versucht, umfassend den 127-Bit-Raum durchsuchen, was umfangreiche Rechen-Betriebsmittel und eine unpraktische Menge Zeit erfordert. Somit kann der Schutz, den die vorliegende Erfindung bietet, nicht ohne weiteres geknackt werden, und die Sicherheit der Erfindung ist insgesamt extrem hoch.

[0089] Zusätzlich dazu, dass eine Betriebsmittelbibliothek-Lizenzierung für ein Programm möglich ist, wenn der API-Verkäufer oder -Lizenzgeber dem Endbenutzer eine Standortlizenz für die API erteilen will, sodass die API für die Verwendung mit jeder Anzahl von Anwendungsprogrammen lizenziert wird, kann die API auch einen LicenseKeyString und einen LicenseAgreementString erhalten, die diese unbeschränkte Verwendung ermöglichen. Bei dieser Ausführungsform stellt der API-Verkäufer dem Endbenutzer einen Standortlizenzschlüsselstring als Genehmigung, die API mit jeder Anzahl von Anwendungen und anderen Endbenutzer-Programmen an diesem Standort zu benutzen, zur Verfügung. Der Standortlizenzschlüsselstring weist eine digitale Signatur eines vom API-Verkäufer erzeugten Lizenzvertragsstrings auf. Der Lizenzvertragsstring kann vom Verkäufer in die API voreingebettet werden. Beim Installieren der API fragt ein mit der API geliefertes Installationsprogramm den Endbenutzer, ob er den Standortlizenzschlüssel kennt. Wenn ja, gibt der Endbenutzer den Standortlizenzschlüssel ein und das Installationsprogramm schreibt den Standortlizenzschlüssel an eine dafür vorgesehene Stelle in der API. Wenn sich danach die API initialisiert, prüft die API das Vorhandensein des Standortlizenzschlüssels. Wenn er vorhanden ist und eine gültige digitale Signatur für den an anderer Stelle in

der API gespeicherten Standortlizenztextstring aufweist, darf die API mit jedem Anwendungsprogramm, das sie gerade aufruft, verwendet werden.

OpenStep-API

[0090] Bei einer Ausführungsform der Erfindung ist die verwendete API die in **Fig.** 8 gezeigte objektorientierte OpenStep-API **820**. Eine Beschreibung der OpenStep-API ist von NeXT Software, Inc. unter dem Titel „OpenStep Specification" („OpenStep-Beschreibung") am 18.10.1994 herausgegeben worden. Zu den Implementierungen der OpenStep-API gehören Implementierungen für die Betriebssysteme Windows NT und Solaris, die von NeXT Software, Inc. bzw. SunSoft, Inc. bezogen werden können.
[0091] Wie in **Fig.** 8 gezeigt, weist die OpenStep-API **820** einen Programmcode auf, der als Application Kit **802**, Foundation Kit **808** und Display-Postscript™-System **804** organisiert ist. (Display Postscript™ ist ein Warenzeichen von Adobe Systems Incorporated.) Das Application Kit **802** stellt grundlegende Betriebsmittel für interaktive Anwendungsprogramme bereit, die Windows nutzen, auf dem Bildschirm zeichnen und auf Benutzer-Aktionen an der Tastatur und Maus reagieren. Das Application Kit **802** enthält Komponenten, die die Benutzeroberfläche definieren. Diese Komponenten umfassen Klassen, Protokolle, Funktionen der Sprache C, Konstanten und Daten-Arten, die von praktisch jeder Anwendung, die unter OpenStep-API läuft, verwendet werden sollen. Ein Hauptzweck des Application Kits **802** besteht darin, einen Rahmen zur Implementierung einer grafischen ereignisgesteuerten Anwendung bereitzustellen.
[0092] Das Foundation Kit **808** stellt wesentliche Software-Funktionen oder Baueinheiten bereit, die Anwendungsprogramme verwenden, um Daten und Betriebsmittel zu verwalten. Das Foundation Kit **808** definiert grundlegende Hilfsprogrammklassen und -dienstmerkmale zur Verarbeitung von Mehrbyte-Zeichensätzen, Objekt-Beständigkeit und -Verteilung und stellt eine Schnittstelle zu normalen Betriebssystem-Einrichtungen bereit. Das Foundation Kit **808** bietet somit ein Maß an Betriebssystem-Unabhängigkeit, sodass der Entwickler ein Anwendungsprogramm besser von einem Betriebssystem zu einem anderen portieren kann.
[0093] Das Display-Postscript-System **804** ist ein Geräte-unabhängiges Abbildungsmodell zum Anzeigen von Dokumenten auf einem Bildschirm. Display Postscript ist von Adobe Systems Incorporated definiert. Das Display-Postscript-System **804** ist eine Anwendungs-unabhängige Schnittstelle zu Postscript.
[0094] Getrennt von der API **820**, aber ebenfalls logisch zwischen dem Anwendungsprogramm **800** und dem Betriebssystem **810** lokalisiert, ist eine Gruppe von Geräte-abhängigen Fenstertechnik-Erweiterungen **806**. Die Erweiterungen **806** aktivieren das Display-Postscript-System **804**, um mit spezieller Grafik- und Bildschirm-Hardware im Computersystem des Endbenutzers, wie etwa Bildschirmspeicher- oder anderer Bildschirm- Hardware, zu kommunizieren.
[0095] **Fig.** 13 zeigt eine Ausführungsform der Erfindung, die mit der OpenStep-API von **Fig.** 8 verwendet wird. Wie in **Fig.** 13 gezeigt, werden bei dieser Ausführungsform der Lizenztextstring und der Lizenzschlüsselstring der Erfindung in einem Eigenschaftslistenbereich **1302** (Info.plist) des Anwendungsprogrammcodes **800** implementiert. Es werden zwei String-Eigenschaften zum Eigenschaftslistenbereich **1302** hinzugefügt: NSLicenseAgreement **1304**, der die für das Anwendungsprogramm **800** geltenden Softwarelizenzbedingungen speichert, und NSLicenseKey **1306**, der den Lizenzschlüssel, der dem NSLicenseAgreement **1304** entspricht, speichert. Bei dieser Ausführungsform wird wie bei der Ausführungsform von **Fig.** 9 der NSLicenseKey **1306** vom NSLicenseAgreement-String **1304** abgeleitet, der mit einem Digitalsignatur-Verfahren und einem privaten Schlüssel des Verkäufers aus dem Lizenzvertrag-String erzeugt wird.
[0096] Beispielwerte der beiden in die Info.plist eingefügten Strings sind in Tabelle 2 angegeben.
[0097]

Tabelle 2: Info.plist-Strings

NSLicenseKey = „Ab76LY2GbbO0GqK2KY17BqHy35“;

NSLicenseAgreement = „© Urheberrecht 1996, EOF AddOnTools Inc., ReportWriter-Lizenzvertrag: Das ist eine Demo-Software, die bis zum 02.11.1996 gültig ist. Es ist rechtlich unzulässig, diese Software zu kopieren.“;

Bei der OpenStep-Ausführungsform von **Fig.** 13 ist die UNLOCK-Funktion **1308** als Teil des Application Kits **802** implementiert. Bei einer Ausführungsform wird die UNLOCK-Funktion **1308** durch Hinzufügen eines entsprechenden Codes zu einer nicht neudefinierbaren privaten Application-Kit-Funktion (beispielsweise _NXAppZone() in NSApplication.m) implementiert. Ein Beispiel für den Quellcode, der hinzugefügt werden kann, ist in Tabelle 3 angegeben.
[0098]

Tabelle 3: Bei der OpenStep-API-Implementierung hinzugefügter UNLOCK-Code

```
static BOOL licenseChecked = NO;
if (! LicenseChecked)
        {
        NSDictionary *info;
        NSString *key, *agreement;
        /* First check the unlimited (per-site) license */
        info = [NSDictionary
dictionaryWithContentsOfFile:@“ /OpenStep/AppKit.dll/Info
.plist“]; // real path TBD
        key = [info objectForKey:@“NSLicenseKey“];
        agreement = [info
objectForKey:@“NSLicenseAgreement“];
        if (!NSCheckLicense(key, agreement))
                {
                /* now check for the per-app license */
                info = [[NSBundle mainBundle] infoDictionary];
                key = [info objectForKey:@“NSLicenseKey“];
                agreement = [info
objectForKey:@“NSLicenseAgreement“];
                if (!NSCheckLicense(key, agreement))
                        {
                        NSLog (@“*** Sorry no valid license for
%@“, [NSApp appName]);
                        }
                }
        licenseChecked = YES;
        }
```

Die NSCheckLicense()-Funktion **1310**, die in dem Code-Segment von Tabelle 3 zweimal aufgerufen wird, ist, wie in **Fig.** 13 gezeigt, im Foundation-Kit-Teil **808** der OpenStep-API **820** implementiert. Die NSCheckLicense()-Funktion **1310** entspricht der CHECK-LICENSE-Funktion **921**, die in **Fig.** 9 dargestellt ist. Die NSCheckLicense()-Funktion 1310 überprüft den NSLicenseAgreement-String **1304** mit dem NSLicenseKey-String **1306** und einem Digitalsignatur-Authentisierungsverfahren. Die NSCheckLicense()-Funktion 1310 hat folgende Definition:

Extern BOOL NSCheckLicense(NSString *licenseKey, NSString *licenseAgreement);

Wie die CHECK-LICENSE-Funktion **921** von **Fig.** 9 verwendet die NSCheckLicense()-Funktion **1310** mit Hilfe des öffentlichen Schlüssels des API-Verkäufers eine CHECK-Funktion **1312** für den NSLicenseAgreement-String **1304** und den NSLicenseKey **1306**, um die Gültigkeit des NSLicenseAgreement-Strings **1304** festzustellen. Bei der Ausführungsform von **Fig.** 13 enthält die CHECK-Funktion **1312** in ihrem Code eine Kopie des öffentlichen Schlüssels (Public Key) des API-Verkäufers **1314**.

[0099] Bei der Ausführungsform von **Fig.** 13 weist die API **820** ein „GEN"-Verfahren 1316 auf, mit dem ein API-Verkäufer Lizenzschlüsselstrings zur Verwendung durch die CHECK-Funktion **1312** schnell erzeugen kann. Bei dem GEN-Verfahren **1316** werden als Eingangssignale ein Lizenzvertragsstring und ein geheimer privater Schlüssel empfangen und als Ausgangssignal wird mit einem Digitalsignatur-Erzeugungsverfahren ein Lizenzschlüsselstring erzeugt. Der private Schlüssel kann beispielsweise ein privater 127-Bit-Schlüssel sein, obwohl auch ein privater Schlüssel einer anderen Größe verwendet werden kann. Das vom GEN-Verfahren **1316** genutzte Signatur-Erzeugungsverfahren ist mit dem von der CHECK-Funktion **1312** genutzten Digitalsignatur-Authentisierungsverfahren kompatibel. Das GEN-Verfahren **1316** selbst kann völlig öffentlich gemacht werden und in der API implementiert werden, wenn der private Schlüssel des API-Verkäufers geheimgehalten wird. Beispielsweise kann das GEN-Verfahren Teil des in **Fig.** 13 gezeigten OpenStep-API-Foundation-Kits **808** sein. Das GEN-Verfahren kann auch in einem gesonderten Programmmodul gehalten werden.

[0100] Die logische Beziehung zwischen dem GEN-Verfahren und der CHECK-Funktion lautet:

CHECK(GEN(LicenseAgreementString, PrivateKey), Public Key, LicenseAgreementString) ⇒ YES

CHECK(random1, random2) ⇒ NO with a very high probability Bei einer Ausführungsform der Erfindung wird eine Shell für das GEN-Verfahren bereitgestellt. Die Shell kann als Eingangssignal einen Lizenzvertrag-Doku-

mentvorlagenstring empfangen, wie
© Urheberrecht 1995, %@, Lizenzvertrag vom %@; Demo-Software gültig bis %@; Es ist rechtlich unzulässig, diesen Vertrag zu kopieren
wobei %@ das zusätzliche Datum ist, das vom API-Verkäufer zu liefern ist. Die Shell fordert dann den Benutzer (d. h. den API-Verkäufer) auf, die zusätzlichen Daten, beispielsweise einen Firmennamen, einen Produktnamen und ein Ablaufdatum, aus denen die Shell einen speziellen Lizenzvertragsstring erzeugt, einzugeben. Die Shell fragt dann nach dem privaten Schlüssel und erzeugt mit dem GEN-Verfahren einen entsprechenden Lizenzschlüssel.
[0101] Diese Shell kann mit anderen Dokumentvorlagen auch für Lizenzschlüssel für ein Programm oder Lizenzschlüssel für einen Standort verwendet werden.
[0102] Bei einer Ausführungsform der Erfindung wird ein Installationsprogramm zum Installieren einer Betriebsmittelbibliothek auf dem Endbenutzer-Computer bereitgestellt. Das Installationsprogramm hat eine Programmfunktion, mit der der Endbenutzer während der Installation einen Standortlizenzschlüssel bereitstellen kann. Wenn beispielsweise die Betriebsmittelbibliothek die OpenStep-API ist, wird ein zusätzlicher Code zum OpenStep-API-Installationsprogramm hinzugefügt. Der Benutzer wird beim Installieren der Betriebsmittelbibliothek gefragt, ob er eine Lizenz für einen Standort erhalten hat. Wenn der Benutzer mit Ja antwortet, wird er aufgefordert, den Standortlizenzschlüsselstring einzugeben. Bei einer Ausführungsform wird der Benutzer auch aufgefordert, den Standort-Lizenzvertragsstring einzugeben. Bei einer anderen Ausführungsform wird der Standort-Lizenzvertragsstring in der Betriebsmittelbibliothek gespeichert, beispielsweise in der Betriebsmitteldatei Info.plist des Application Kits der DLL der OpenStep-API. Die Gültigkeit des Standortlizenzschlüssels und des Standortlizenzvertrags wird, wie vorstehend beschrieben, mit der CHECK-LICENSE-Funktion überprüft. Die Benutzung der Betriebsmittelbibliothek wird nur dann erlaubt, wenn der vom Benutzer eingegebene Standortlizenzschlüsselstring dem Standortlizenzvertragsstring entspricht (d. h. wenn festgestellt wird, dass er die digitale Betriebsmittelbibliothek-Verkäufer-Signatur des Standortlizenzvertragsstrings aufweist).

Java

[0103] Die vorliegende Erfindung kann mit Betriebsmittelbibliotheken, wie Java-Klassendateien, Java-Applets und Java-Bytecode-Paketen, verwendet werden. **Fig.** 14 zeigt eine Ausführungsform der Erfindung, bei der die Betriebsmittelbibliothek ein Java-Applet ist. Bei der in **Fig.** 14 gezeigten Ausführungsform wird ein Applet von einer HTML-Seite **1402** über ein Applet-Tag **1404** aufgerufen. Das Applet-Tag **1404** weist den Namen der Klassendatei des Applets und Applet-Parameter **1406** auf. Die Applet-Parameter **1406** umfassen einen Lizenzvertragsstring-Parameter **1408** und einen Lizenzschlüsselstring-Parameter **1410**. Der Lizenzvertragsstring-Parameter **1408** gibt einen Lizenzvertragsstring an, der Bedingungen einer Lizenz für die Verwendung des angeforderten Applets enthält. Der Lizenzschlüsselstring-Parameter **1410** gibt einen Lizenzschlüssel an, der zur Authentisierung des Lizenzvertragsstrings verwendet wird. Wie bei anderen Ausführungsformen der Erfindung weist bei dieser Ausführungsform der Lizenzschlüsselstring eine digitale Signatur des Betriebsmittelbibliothek-(Applet)Verkäufers des Lizenzvertragsstrings auf. Tabelle 4 zeigt ein Beispiel für das Applet-Tag **1404**.

Tabelle 4

```
<APPLET CODE="Applet.class"  WIDTH=250  HEIGHT=75>
<PARAM NAME=LicenseAgreementString  VALUE="Web page
orderform.html licensed to use applet 'Applet.class'>
<PARAM NAME=LicenseKeyString  VALUE="4kd094kak2rtx0kzq">
</APPLET>
```

[0104] In dem Beispiel von Tabelle 4 gibt der Lizenzvertragsstring den Namen der HTML-Seite („orderform-html") und den Namen des lizenzierten Applets („applet.class") an.
[0105] Wie in **Fig.** 14 gezeigt, wird auf das Applet **1434** zugegriffen, wenn die HTML-Seite **1402** von einem HTML-Browser **1430** geladen wird, der auf einem Client-Computer **1420** läuft. Bei der Ausführungsform von **Fig.** 13 läuft der HTML-Browser **1430** auf einer API **1424**, die wiederum auf einem Betriebssystem **1422** läuft. Der HTML-Browser **1430** weist eine virtuelle Java-Maschine **1432** zum Abarbeiten von Java-Applets auf.
[0106] Wenn der HTML-Browser **1430** beim Laden der HTML-Seite **1402** auf ein Applet-Tag **1404** stößt, ruft er die den Applet **1434** bildenden Klassendateien aus den Speicherzellen im Client-Computer **1420** und/oder gegebenenfalls aus einem oder mehr Server-Computern ab. Eine der Klassendateien ist die CheckLicense-Klassendatei **1436**. Nachdem der HTML-Browser **1430** alle erforderlichen Komponenten des Applets **1434** abgerufen hat, wird das Applet **1434** initialisiert. Während der Initialisierung oder zu einem späteren Zeitpunkt wird die von der CheckLicense-Klassendatei **1436** bereitgestellte CheckLicense-Funktion aufgerufen. Wie bei anderen Ausführungsformen stellt die CheckLicense-Funktion fest, ob die anfordernde Einheit (HTML-Seite

1402) eine gültige Lizenz zur Verwendung des angeforderten Betriebsmittels (Applet **1434**) besitzt, indem sie mit dem vom LicenseKeyString-Parameter **1410** festgelegten Lizenzschlüssel und dem öffentlichen Schlüssel des Applet-Verkäufers **1438** die Authentizität der vom LicenseAgreementString-Parameter **1408** festgelegten Lizenz prüft. Wenn die CheckLicense-Funktion feststellt, dass die HTML-Seite **1402** eine gültige Lizenz besitzt, darf das Applet **1434** abgearbeitet werden. Wenn nicht, wird die Abarbeitung des Applets **1434** beendet und eine Fehlermeldung wird an den HTML-Browser **1430** gesendet.

[0107] Somit ist ein verbessertes Verfahren und eine verbesserte Vorrichtung zur Erzwingung von Software-Lizenzen vorgestellt worden. Obwohl die vorliegende Erfindung in Bezug auf bestimmte exemplarische Ausführungsformen beschrieben worden ist, dürften Fachleute erkennen, dass die vorliegende Erfindung nicht auf diese speziellen Ausführungsformen beschränkt ist. Obwohl die Erfindung beispielsweise für die Verwendung in Einzelplatzrechnersystemen beschrieben worden ist, kann die Erfindung auch zur Erzwingung von Lizenzen in einer Netzwerk-Umgebung verwendet werden. Und obwohl die Funktionsweise von bestimmten Ausführungsformen unter Verwendung bestimmter Software-Programme und bestimmter detaillierter Prozessschritte detailliert beschrieben worden ist, kann andere Software verwendet werden und einige der Schritte können weggelassen werden oder andere ähnliche Schritte können ersetzt werden, ohne vom Schutzumfang der Erfindung abzuweichen. Fachleute werden weitere Ausführungsformen erkennen, die die Erfindungsmerkmale der vorliegenden Erfindung aufweisen.

Patentansprüche

1. Vorrichtung zum Begrenzen einer Nutzung eines Software-Betriebsmittels (**215**) in einer Computer-Betriebsumgebung mit einem Softwareprogramm und dem Software-Betriebsmittel (**215**), mit:
– einem Zugriff-Autorisierungsanzeiger (**500**), der mit dem Softwareprogramm in Verbindung steht;
–- Mittel zum Lesen des Zugriff-Autorisierungsanzeigers in dem Software-Betriebsmittel (**215**);
– Mittel in dem Software-Betriebsmittel (**215**) zum Bestimmen, ob der Zugriff-Autorisierungsanzeiger (**500**) gültig ist; und
– Mittel zum Erlauben eines Zugriffs mit Hilfe des Softwareprogramms auf das Software-Betriebsmittel (**215**), nur für den Fall, daß bestimmt wird, daß der Zugriff-Autorisierungsanzeiger (**500**) gültig ist;
dadurch gekennzeichnet, daß der Zugriff-Autorisierungsanzeiger (**500**) wenigstens eine Lizenzbestimmung zur Nutzung der Software und einen Lizenzschlüssel (**610**) umfaßt, der der wenigstens einen Lizenzbestimmung zur Nutzung der Software entspricht.

2. Vorrichtung nach Anspruch 1, wobei der Zugriff-Autorisierungsanzeiger (**500**) Ber wenigstens einen Lizenzbestimmung zur Nutzung der Software entstimmungen einer Standortlizenz umfaßt.

3. Vorrichtung nach Anspruch 1, wobei der Zugriff-Autorsierungsanzeiger (**500**) in das Softwareprogramm eingebettet ist.

4. Vorrichtung nach Anspruch 1, wobei das Software-Betriebsmittel (**215**) eine API ("application program interface" – Anwendungsprogrammschnittstelle) aufweist.

5. Vorrichtung nach Anspruch 1, wobei das Software-Betriebsmittel (**215**) eine Laufzeit-Bibliothek umfaßt.

6. Vorrichtung nach Anspruch 1, wobei das Software-Betriebsmittel (**215**) eine dynamische Verknüpfungsbibliothek umfaßt.

7. Vorrichtung nach Anspruch 1, wobei das Software-Betriebsmittel (**215**) ein Applet umfaßt.

8. Vorrichtung nach Anspruch 1, wobei das Software-Betriebsmittel (**215**) ein Bytecode-Paket umfaßt.

9. Vorrichtung nach Anspruch 1, wobei das Software-Betriebsmittel (**215**) ein OLE ("object linking and embedding" – Objektverknüpfung und -einbettung) aktiviertes Anwendungsprogramm umfaßt.

10. Vorrichtung nach Anspruch 3, wobei der Zugriff-Autorisierungsanzeiger (**500**) in einem Vereinbarungsbereich für Konstanten des Softwareprogramms spezifiziert ist.

11. Vorrichtung nach Anspruch 3, wobei der Zugriff-Autorisierungsanzeiger (**500**) ein Eigentum an einer Eigentumsliste des Softwareprogramms umfaßt.

12. Vorrichtung nach Anspruch 1, wobei der Lizenzschlüssel (**610**) ein mit dem Zugriff-Autorisierungsan-

zeiger verbundener Identifizierer ist und wobei die Mittel zum Bestimmen der Gültigkeit des Zugriff-Autorisierungsanzeigers (**500**) Mittel umfassen, um zu bestimmen, ob der Zugriff-Autorisierungsanzeiger (**500**) auf Basis des Identifizierers gültig ist.

13. Vorrichtung nach Anspruch 12, gekennzeichnet durch Mittel zum Empfangen des Identifizierers von einem Endnutzer.

14. Vorrichtung nach Anspruch 13, gekennzeichnet durch Mittel zum Speichern des Identifizierers in dem Software-Betriebsmittel (**215**).

15. Vorrichtung nach Anspruch 12, wobei der Identifizierer in das Softwareprogramm eingebettet ist.

16. Vorrichtung nach Anspruch 12, wobei der Identifizierer eine digitale Signatur des Zugriff-Autorisierungsanzeigers (**500**) umfaßt.

17. Vorrichtung nach Anspruch 15, wobei der Identifizierer in einem Vereinbarungsbereich für Konstanten des Softwareprogramms spezifiziert ist.

18. Vorrichtung nach Anspruch 15, wobei der Identifizierer ein Eigentum an einer Eigentumsliste des Softwareprogramms umfaßt.

19. Vorrichtung nach Anspruch 16, wobei die Mittel zum Bestimmen, ob der Zugriff-Autorisierungsanzeiger (**500**) auf Basis des Identifizierers gültig ist, Mittel für eine digitale Signaturbeglaubigung umfassen.

20. Vorrichtung nach Anspruch 1, gekennzeichnet durch Mittel zum Bestimmen, ob die wenigstens eine Lizenzbestimmung zutrifft.

21. Vorrichtung nach Anspruch 16, wobei das Softwareprogramm den Zugriff-Autorisierungsanzeiger (**500**) und den Identifizierer umfaßt.

22. Verfahren zum Begrenzen einer Nutzung eines Software-Betriebsmittels (**215**) in einer Computer-Betriebsumgebung (**100**, **110**, **220**), wobei das Verfahren die folgenden Schritte umfaßt:
– Empfangen einer Anforderung eines Softwareprogramms zum Nutzen des Betriebsmittels (**215**);
– Erhalten eines Zugriff-Autorisierungsanzeigers (**500**), der mit dem Softwareprogramm in Verbindung steht, wobei der Zugriff-Autorisierungsanzeiger (**500**) wenigstens eine Lizenzbestimmung zur Nutzung des Software-Betriebsmittels umfaßt;
– Erhalten eines Lizenzschlüssels (**610**), der der wenigstens einen Lizenzbestimmung für die Nutzung der Software entspricht;
– Bestimmen, ob der Zugriff-Autorisierungsanzeiger (**500**) gültig ist; und
– Erlauben einer Nutzung des Software-Betriebsmittels (**215**) durch das Softwareprogramm, nur wenn bestimmt wird, daß der Zugriff-Autorisierungsanzeiger (**500**) gültig ist.

23. Verfahren nach Anspruch 22, wobei die wenigstens eine Lizenzbestimmung eine Standortlizenz umfaßt.

24. Verfahren nach Anspruch 22, wobei der Zugriff-Autorisierungsanzeiger (**500**) in das Softwareprogramm eingebettet ist.

25. Verfahren nach Anspruch 22, wobei das Software-Betriebsmittel (**215**) eine API ("application program interface" – Anwendungsprogrammschnittstelle) umfaßt.

26. Verfahren nach Anspruch 22, wobei das Software-Betriebsmittel (**215**) eine Laufzeit-Bibliothek umfaßt.

27. Verfahren nach Anspruch 22, wobei das Software-Betriebsmittel (**215**) eine dynamische Verknüpfungsbibliothek umfaßt.

28. Verfahren nach Anspruch 22, wobei das Software-Betriebsmittel (**215**) ein Applet umfaßt.

29. Verfahren nach Anspruch 22, wobei das Software-Betriebsmittel (**215**) ein Bytecode-Paket umfaßt.

30. Verfahren nach Anspruch 22, wobei das Software-Betriebsmittel (**215**) ein OLE ("object linking and embedding" – Objektverknüpfung und -einbettung) aktiviertes Anwendungsprogramm umfaßt.

31. Verfahren nach Anspruch 24, wobei der Zugriff-Autorisierungsanzeiger (**500**) in einem Vereinbarungsbereich für Konstanten des Softwareprogramms spezifiziert ist.

32. Verfahren nach Anspruch 24, wobei der Zugriff-Autorisierungsanzeiger (**500**) ein Eigentum an einem Eigentumslistenbereich des Softwareprogramms umfaßt.

33. Verfahren nach Anspruch 22, wobei der Lizenzschlüssel (**610**) ein mit dem Zugriff-Autorisierungsanzeiger (**500**) in Verbindung stehender Identifizierer ist und wobei bei dem Bestimmen der Gültigkeit des Zugriff-Autorisierungsanzeigers (**500**) bestimmt wird, ob der Zugriff-Autorisierungsanzeiger (**500**) auf Basis des Identifizierers gültig ist, welcher mit dem Zugriff-Autorisierungsanzeiger (**500**) verbunden ist.

34. Verfahren nach Anspruch 33, gekennzeichnet durch das Akzeptieren des Identifizierers von einem Nutzer.

35. Verfahren nach Anspruch 34, gekennzeichnet durch das Speichern des Identifizierers in dem Software-Betriebsmittel.

36. Verfahren nach Anspruch 33, wobei der Identifizierer in das Softwareprogramm eingebettet ist.

37. Verfahren nach Anspruch 33, wobei der Identifizierer eine digitale Signatur des Zugriff-Autorisierungsanzeigers umfaßt.

38. Verfahren nach Anspruch 36, wobei der Identifizierer in einem Vereinbarungsbereich für Konstanten des Softwareprogramms spezifiziert wird.

39. Verfahren nach Anspruch 36, wobei der Identifizierer ein Eigentum an einem Eigentumslistenbereich des Softwareprogramms umfaßt.

40. Verfahren nach Anspruch 33, wobei Beglaubigungsmittel für eine digitale Signatur genutzt werden, um zu bestimmen, ob der Zugriff-Autorisierungsanzeiger auf Basis des Identifizierers gültig ist.

41. Verfahren nach Anspruch 22, wobei bestimmt wird, ob die wenigstens eine Lizenzbestimmung übereinstimmt.

42. Verfahren nach Anspruch 37, wobei das Softwareprogramm den Zugriff-Autorisierungsanzeiger (**500**) und den Identifizierer umfaßt.

43. Programmspeichereinrichtung, die mit Hilfe einer Maschine lesbar ist und auf der ein Programm mit von der Maschine ausführbaren Instruktionen verständlich ausgeführt ist, um ein Verfahren zum Begrenzen einer Nutzung eines Software-Betriebsmittels (**215**) auszuführen, wobei das Verfahren die folgenden Schritte umfaßt:
– Empfangen einer Anforderung eines Softwareprogramms zum Nutzen des Betriebsmittels (**215**);
–Erhalten eines Zugriff-Autorisierungsanzeigers (**500**), der mit dem Softwareprogramm in Verbindung steht, wobei der Zugriff-Autorisierungsanzeiger (**500**) wenigstens eine Lizenzausbestimmung zur Nutzung des Software-Betriebsmittels (**215**) umfaßt;
– Erhalten eines Lizenzschlüssels (**610**), der der wenigstens einen Lizenzbestimmung für die Nutzung der Software entspricht;
– Bestimmen, ob der Zugriff-Autorisierungsanzeiger (**500**) gültig ist; und
–Erlauben einer Nutzung des Software-Betriebsmittels (**215**) durch das Softwareprogramm, nur wenn bestimmt wird, daß der Zugriff-Autorisierungsanzeiger (**500**) gültig ist.

44. Programmspeichereinrichtung nach Anspruch 43, wobei die wenigstens eine Lizenzbestimmung eine Standortlizenz umfaßt.

45. Programmspeichereinrichtung nach Anspruch 43, wobei der Zugriff-Autorisierungsanzeiger (**500**) in das Softwareprogramm eingebettet ist.

46. Programmspeichereinrichtung nach Anspruch 43, wobei das Software-Betriebsmittel (**215**) ein API (" application program interface" – Anwendungsprogrammschnittstelle) umfaßt.

47. Programmspeichereinrichtung nach Anspruch 43, wobei das Software-Betriebsmittel (**215**) eine Laufzeit-Bibliothek umfaßt.

48. Programmspeichereinrichtung nach Anspruch 43, wobei das Software-Betriebsmittel (**215**) eine dynamische Verknüpfungsbibliothek umfaßt.

49. Programmspeichereinrichtung nach Anspruch 43, wobei das Software-Betriebsmittel (**215**) ein Applet umfaßt.

50. Programmspeichereinrichtung nach Anspruch 43, wobei das Software-Betriebsmittel ein Bytecode-Paket umfaßt.

51. Programmspeichereinrichtung nach Anspruch 43, wobei das Software-Betriebsmittel (**215**) ein OLE ("object linking and embedding" – Objektverknüpfung und -einbettung) aktiviertes Anwendungsprogramm umfaßt.

52. Programmspeichereinrichtung nach Anspruch 45, wobei der Zugriff-Autorisierungsanzeiger (**500**) in einem Vereinbarungsbereich für Konstanten des Softwareprogramms spezifiziert ist.

53. Programmspeichereinrichtung nach Anspruch 45, wobei der Zugriff-Autorisierungsanzeiger (**500**) ein Eigentum an einem Eigentumslistenbereich des Softwareprogramms umfaßt.

54. Programmspeichereinrichtung nach Anspruch 43, wobei der Lizenzschlüssel (**610**) ein mit dem Zugriff-Autorisierungsanzeiger (**500**) verbundener Identifizierer ist und wobei beim Bestimmen der Gültigkeit des Zugriff-Autorisierungsanzeigers (**500**) be- stimmt wird, ob der Zugriff-Autorisierungsanzeiger (**500**) auf Basis des Identifizierers gültig ist, welcher mit dem Zugriff-Autorisierungsanzeiger (**500**) verbunden ist.

55. Programmspeichereinrichtung nach Anspruch 54, wobei das Verfahren des Akzeptieren des Identifizierers von einem Nutzer umfaßt.

56. Programmspeichereinrichtung nach Anspruch 55, wobei das Verfahren das Speichern des Identifizierers in dem Software-Betriebsmittel (**215**) umfaßt.

57. Programmspeichereinrichtung nach Anspruch 54, wobei der Identifizierer in das Softwareprogramm eingebettet ist.

58. Programmspeichereinrichtung nach Anspruch 54, wobei der Identifizierer eine digitale Signatur des Zugriff-Autorisierungsanzeigers (**500**) umfaßt.

59. Programmspeichereinrichtung nach Anspruch 57, wobei der Identifizierer in einem Vereinbarungsbereich für Konstanten des Softwareprogramms spezifiziert wird.

60. Programmspeichereinrichtung nach Anspruch 57, wobei der Identifizierer ein Eigentum an einem Eigentumslistenbereich des Softwareprogramms umfaßt.

61. Programmspeichereinrichtung nach Anspruch 54, wobei Beglaubigungsmittel für eine digitales Signatur benutzt werden, um zu bestimmen, ob der Zugriff-Autorisierungsanzeiger (**500**) auf Basis des Identifizierers gültig ist.

62. Programmspeichereinrichtung nach Anspruch 43, wobei bestimmt wird, ob die wenigstens eine Lizenzbestimmung übereinstimmt.

63. Programmspeichereinrichtung nach Anspruch 58, wobei das Softwareprogramm den Zugriff-Autorisierungsanzeiger (**500**) und den Identifizierer umfaßt.

64. Herstellungsartikel mit:
einem computerlesbaren Medium mit einem computerlesbaren Programmcode, der hierauf ausgeführt ist, um

einem Softwareprogramm einen Zugriff auf ein Software-Betriebsmittel zu erlauben, wobei der computerlesbare Programmcode in dem Herstellungsartikel die folgenden Merkmale aufweist:
– computerlesbarer Programmcode, der ein Softwareprogramm verkörpert; und
– computerlesbarer Programmcode, der einen Zugriff-Autorisierungsanzeiger (**500**) verkörpert;
dadurch gekennzeichnet, daß der Zugriff-Autorisierungsanzeiger (**500**) wenigstens eine Lizenzbestimmung für eine Nutzung des Software-Betriebsmittels (**215**) und einen Lizenzschlüssel (**610**) umfaßt, welcher der wenigstens einen Lizenzbestimmung zur Nutzung der Software entspricht.

65. Herstellungsartikel nach Anspruch 64, wobei der computerlesbare Programmcode ein Softwareprogramm umfaßt und wobei der Zugriff-Autorisierungsanzeiger (**500**) in das Softwareprogramm eingebettet ist.

66. Herstellungsartikel nach Anspruch 64, wobei das Software-Betriebsmittel (**215**) ein API („application program Interface" – Anwendungsprogrammschnittstelle) umfaßt.

67. Herstellungsartikel nach Anspruch 64, wobei das Software-Betriebsmittel eine Laufzeit-Bibliothek umfaßt.

68. Herstellungsartikel nach Anspruch 64, wobei das Software-Betriebsmittel (**215**) eine dynamische Verknüpfungsbibliothek umfaßt.

69. Herstellungsartikel nach Anspruch 64, wobei das Software-Betriebsmittel (**215**) ein Applet umfaßt.

70. Herstellungsartikel nach Anspruch 64, wobei das Software-Betriebsmittel (**215**) ein Bytecode-Paket umfaßt.

71. Herstellungsartikel nach Anspruch 64, wobei das Software-Betriebsmittel (**215**) ein OLE („object linking and embedding" – Objektverknüpfung und -einbettung) aktiviertes Anwendungsprogramm umfaßt.

72. Herstellungsartikel nach Anspruch 65, wobei der Zugriff-Autorisierungsanzeiger (**500**) in einem Vereinbarungsbereich für Konstanten des Softwareprogramms spezifiziert ist.

73. Herstellungsartikel nach Anspruch 65, wobei der Zugriff-Autorisierungsanzeiger (**500**) ein Eigentum an einer Eigentumsliste des Softwareprogramms umfaßt.

74. Herstellungsartikel nach Anspruch 64, wobei der Lizenzschlüssel (**610**) ein mit dem Zugriff-Autorisierungsanzeiger (**500**) verbundener Identifizierer ist und wobei der Herstellungsartikel computerlesbaren Programmcode umfaßt, der den mit dem Zugriff-Autorisierungsanzeiger (**500**) verbundenen Identifizierer verkörpert.

75. Herstellungsartikel nach Anspruch 74, wobei der Identifizierer in das Softwareprogramm eingebettet ist.

76. Herstellungsartikel nach Anspruch 74, wobei der Identifizierer eine digitale Signatur des Zugriff-Autorisierungsanzeigers (**500**) umfaßt.

77. Herstellungsartikel nach Anspruch 74, wobei der Identifizierer in einem Vereinbarungsbereich für Konstanten des Softwareprogramms spezifiziert ist.

78. Herstellungsartikel nach Anspruch 74, wobei der Identifizierer ein Eigentum an einer Eigentumsliste des Softwareprogramms umfaßt.

79. Herstellungsartikel nach Anspruch 76, wobei das Softwareprogramm den Zugriff-Autorisierungsanzeiger (**500**) und den Identifizierer umfaßt.

Es folgen 12 Blatt Zeichnungen

Anhängende Zeichnungen

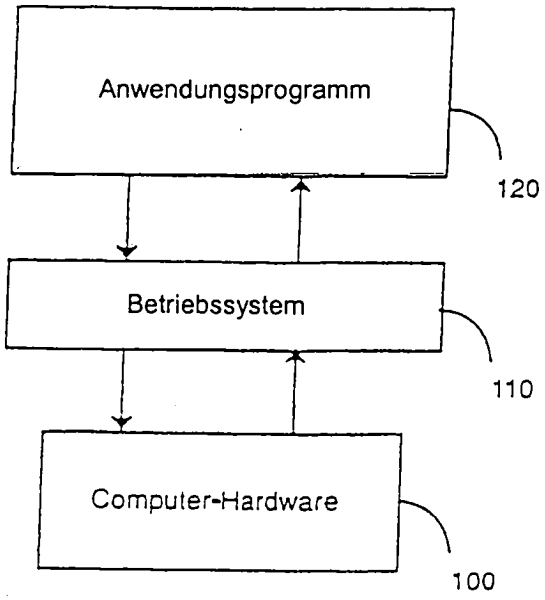


FIG. 1

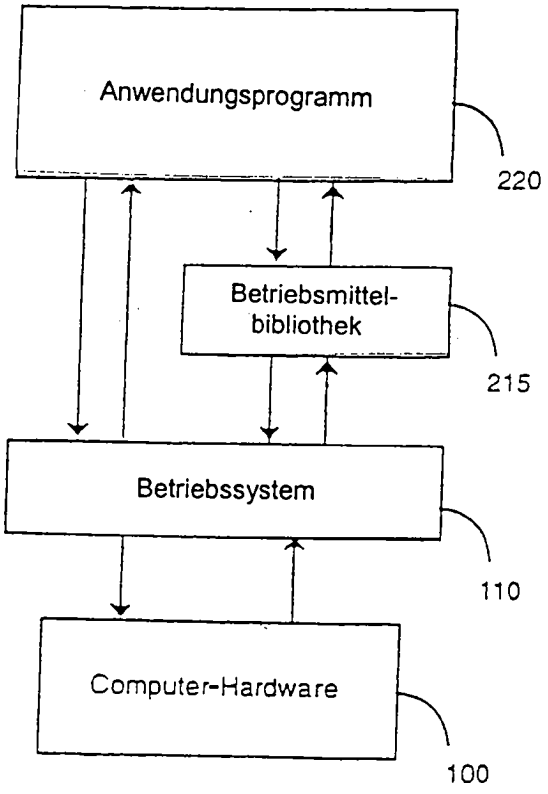


FIG. 2

FIG. 3

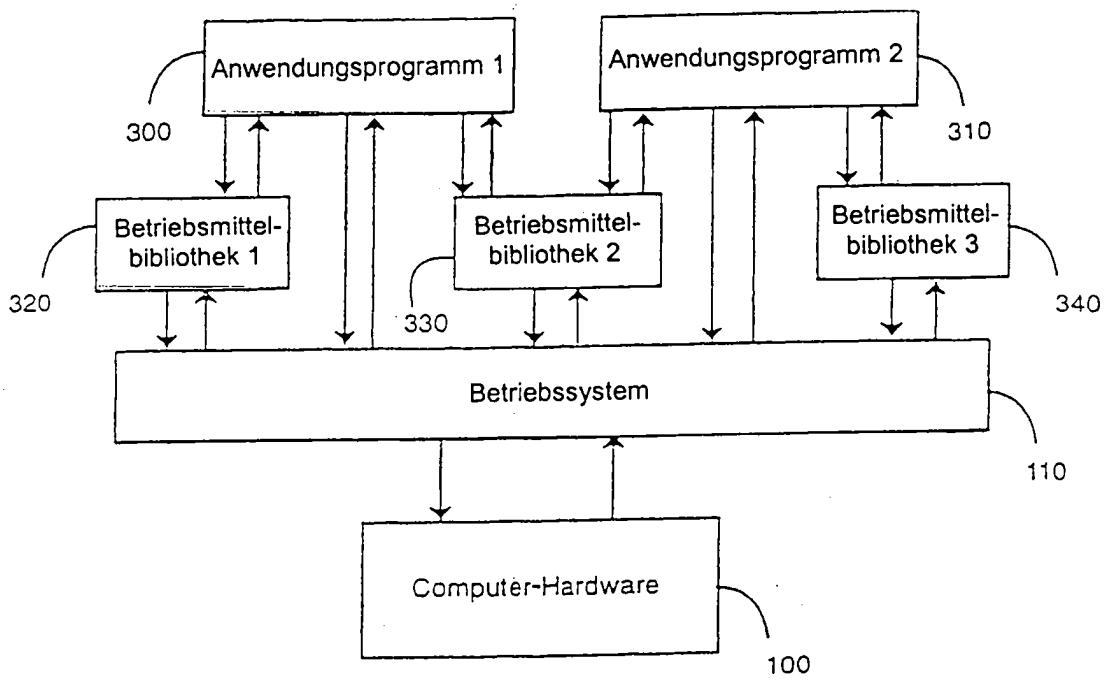
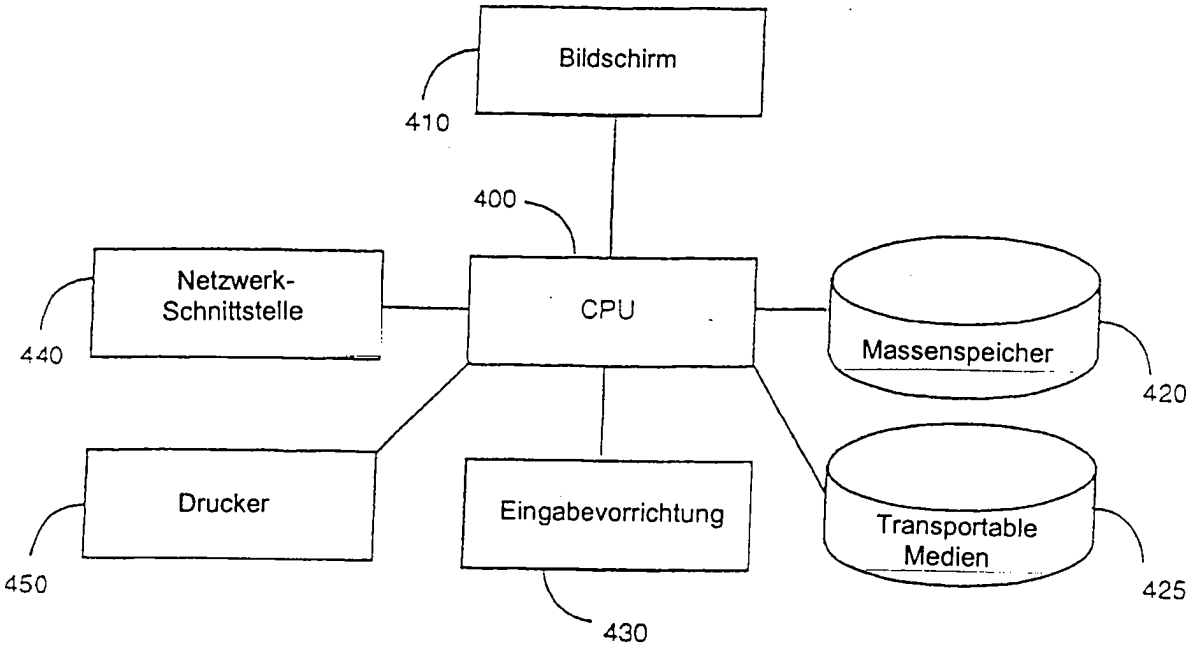

Anwendungsprogramm 1
Anwendungsprogramm 2
300
310
Betriebsmittel-bibliothek 1
Betriebsmittel-bibliothek 2
Betriebsmittel-bibliothek 3
320
330
340
Betriebssystem
110
Computer-Hardware
100

FIG. 4


Bildschirm
410
400
Netzwerk-
Schnittstelle
440
CPU
Massenspeicher
420
Drucker
450
Eingabevorrichtung
430
Transportable
Medien
425

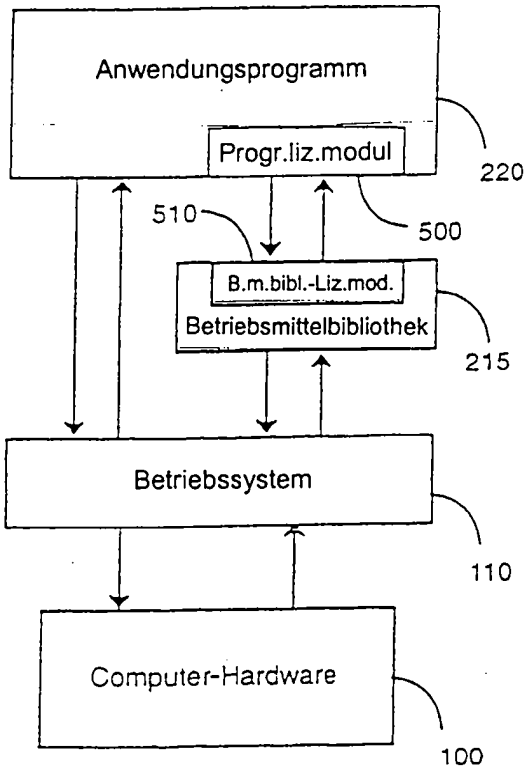

Anwendungsprogramm
Progr.liz.modul
220
510
500
B.m.bibl.-Liz.mod.
Betriebsmittelbibliothek
215
Betriebssystem
110
Computer-Hardware
100

FIG. 5

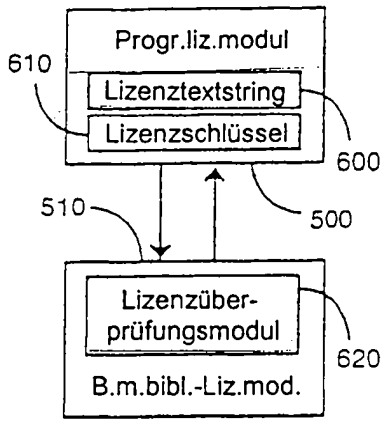

Progr.liz.modul
610
Lizenztextstring
Lizenzschlüssel
600
510
500
Lizenzüber-
prüfungsmodul
620
B.m.bibl.-Liz.mod.

FIG. 6

FIG. 7

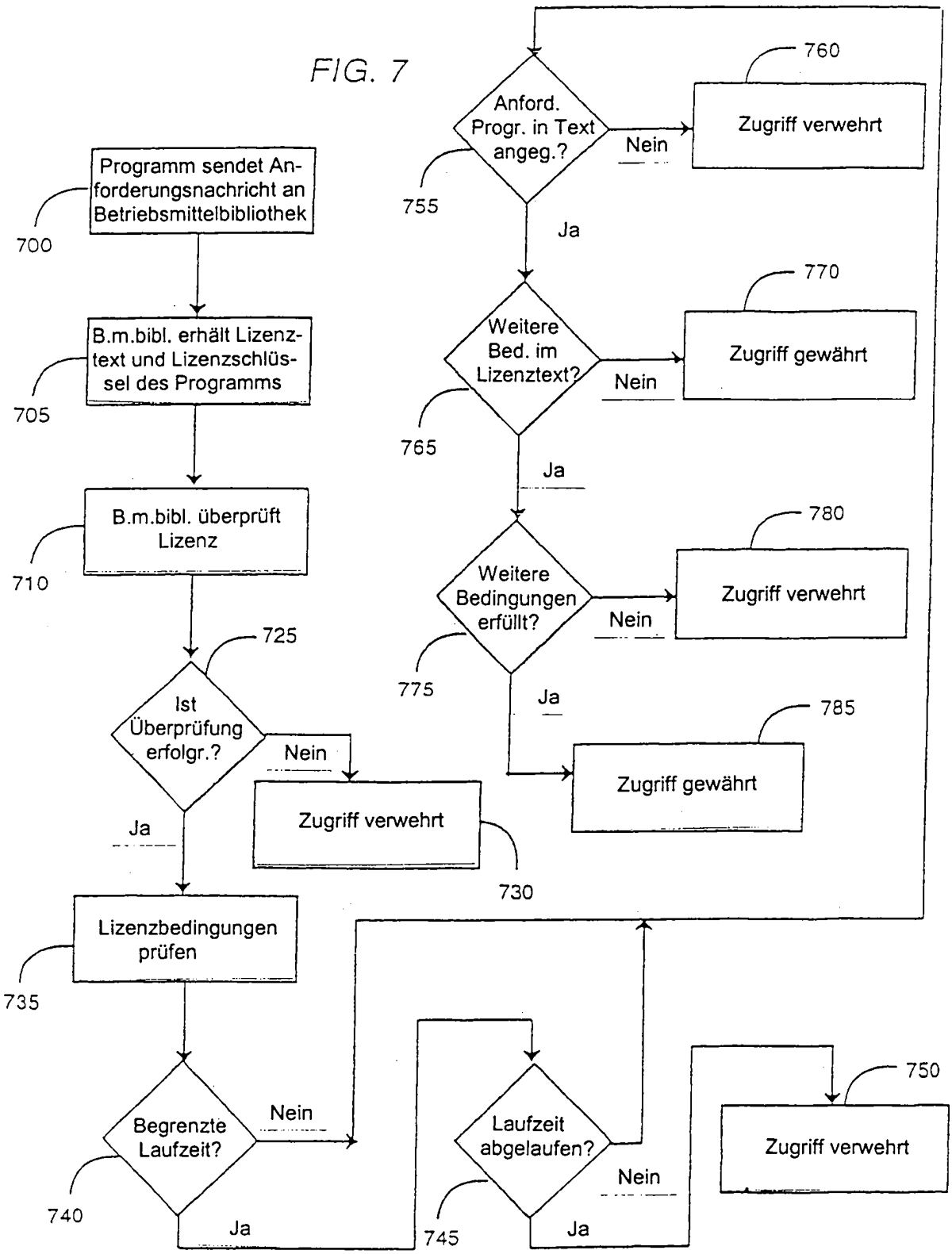

Programm sendet Anforderungsnachricht an Betriebsmittelbibliothek
700
B.m.bibl. erhält Lizenztext und Lizenzschlüssel des Programms
705
B.m.bibl. überprüft Lizenz
710
Ist Überprüfung erfolgr.?
725
Nein
Zugriff verwehrt
730
Ja
Lizenzbedingungen prüfen
735
Begrenzte Laufzeit?
740
Nein
Ja
Anford. Progr. in Text angeg.?
755
Nein
Zugriff verwehrt
760
Ja
Weitere Bed. im Lizenztext?
765
Nein
Zugriff gewährt
770
Ja
Weitere Bedingungen erfüllt?
775
Nein
Zugriff verwehrt
780
Ja
Zugriff gewährt
785
Laufzeit abgelaufen?
745
Nein
Ja
Zugriff verwehrt
750

FIG. 8

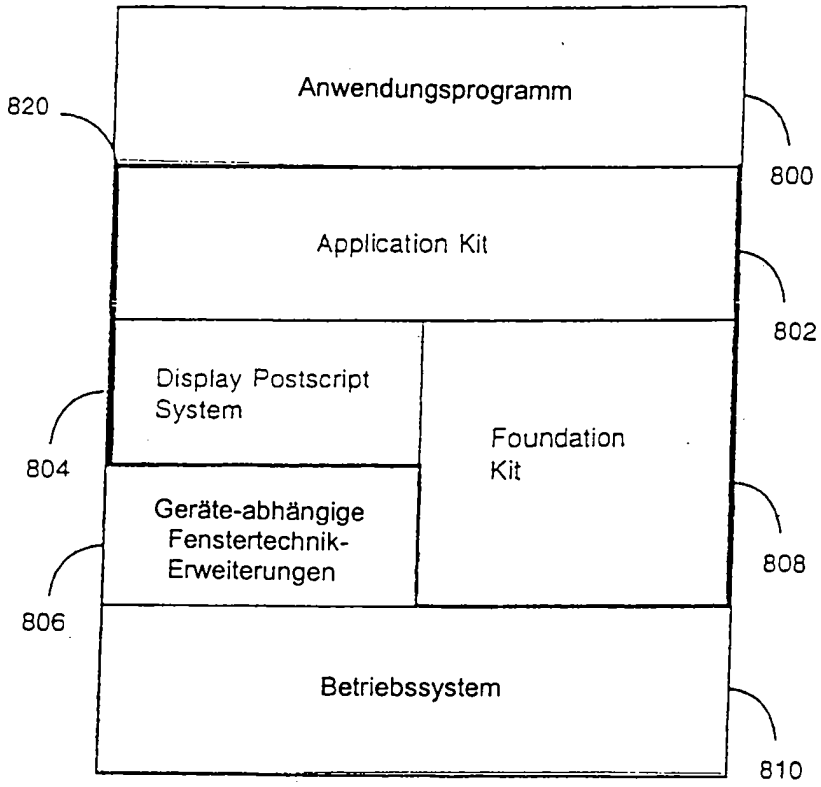
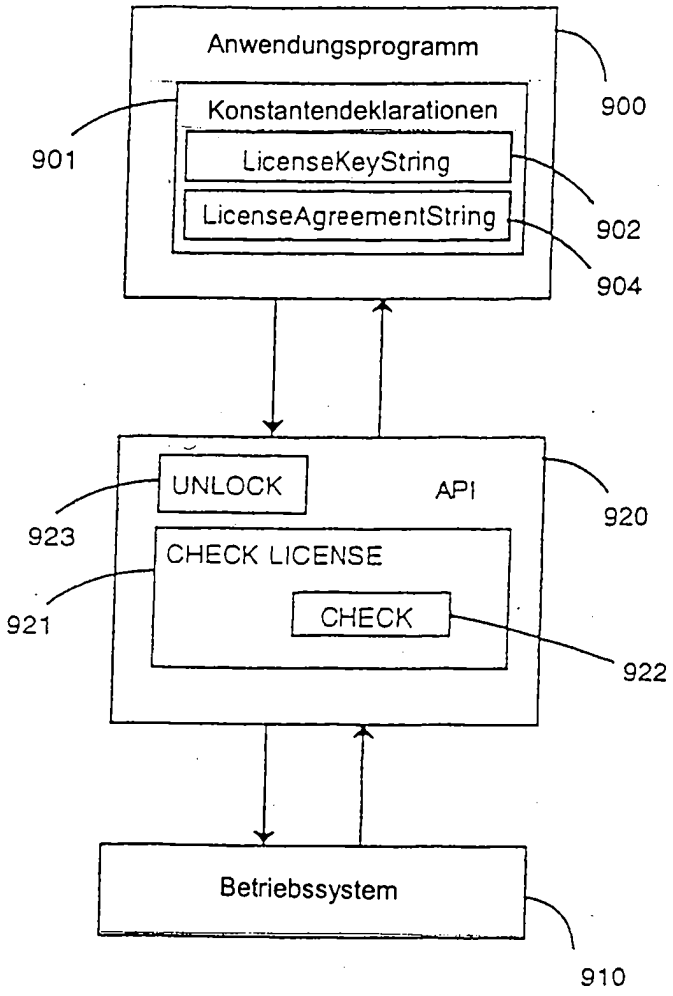

Anwendungsprogramm
820
800
Application Kit
802
Display Postscript System
804
Foundation Kit
Geräte-abhängige Fenstertechnik-Erweiterungen
806
808
Betriebssystem
810

FIG. 9


Anwendungsprogramm
900
Konstantendeklarationen
901
LicenseKeyString
902
LicenseAgreementString
904
UNLOCK
API
920
923
CHECK LICENSE
921
CHECK
922
Betriebssystem
910

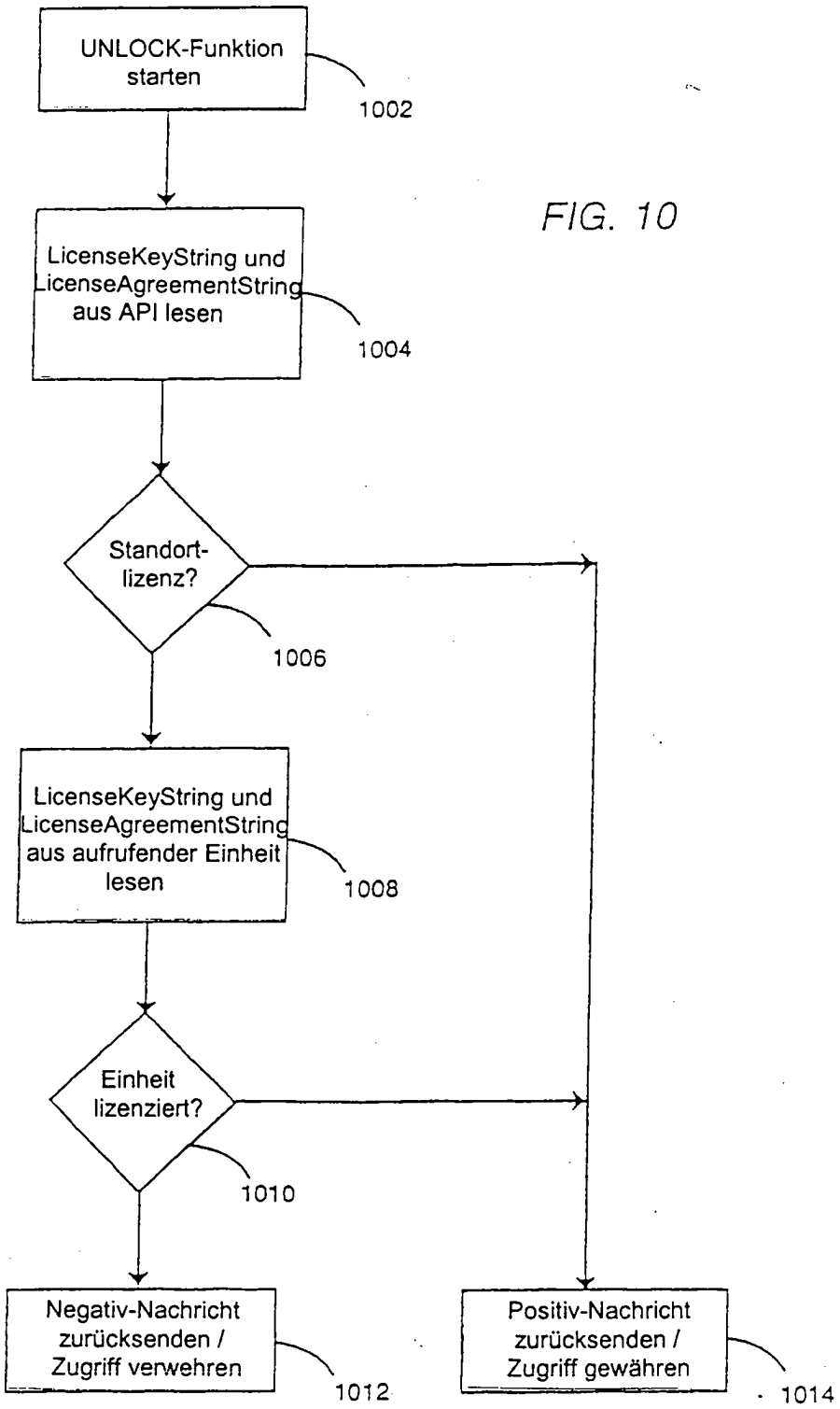

UNLOCK-Funktion starten
1002
LicenseKeyString und LicenseAgreementString aus API lesen
1004
Standort-lizenz?
1006
LicenseKeyString und LicenseAgreementString aus aufrufender Einheit lesen
1008
Einheit lizenziert?
1010
Negativ-Nachricht zurücksenden / Zugriff verwehren
1012
Positiv-Nachricht zurücksenden / Zugriff gewähren
1014

FIG. 10

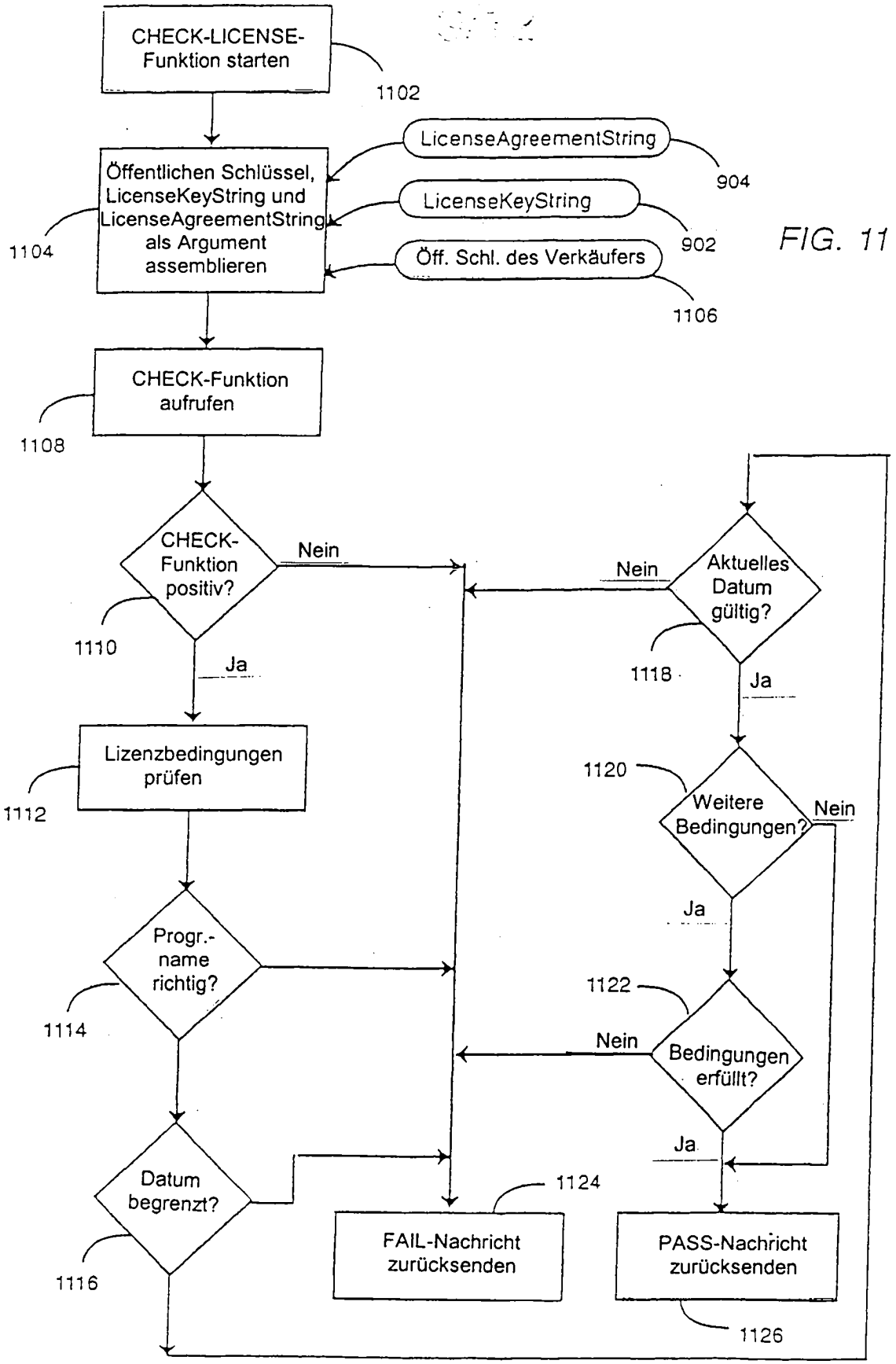

CHECK-LICENSE-Funktion starten
1102
LicenseAgreementString
904
Öffentlichen Schlüssel, LicenseKeyString und LicenseAgreementString als Argument assemblieren
1104
LicenseKeyString
902
Öff. Schl. des Verkäufers
1106
CHECK-Funktion aufrufen
1108
CHECK-Funktion positiv?
1110
Nein
Ja
Lizenzbedingungen prüfen
1112
Progr.-name richtig?
1114
Datum begrenzt?
1116
Aktuelles Datum gültig?
1118
Nein
Ja
1120
Weitere Bedingungen?
Nein
Ja
1122
Bedingungen erfüllt?
Nein
Ja
1124
FAIL-Nachricht zurücksenden
PASS-Nachricht zurücksenden
1126

FIG. 11

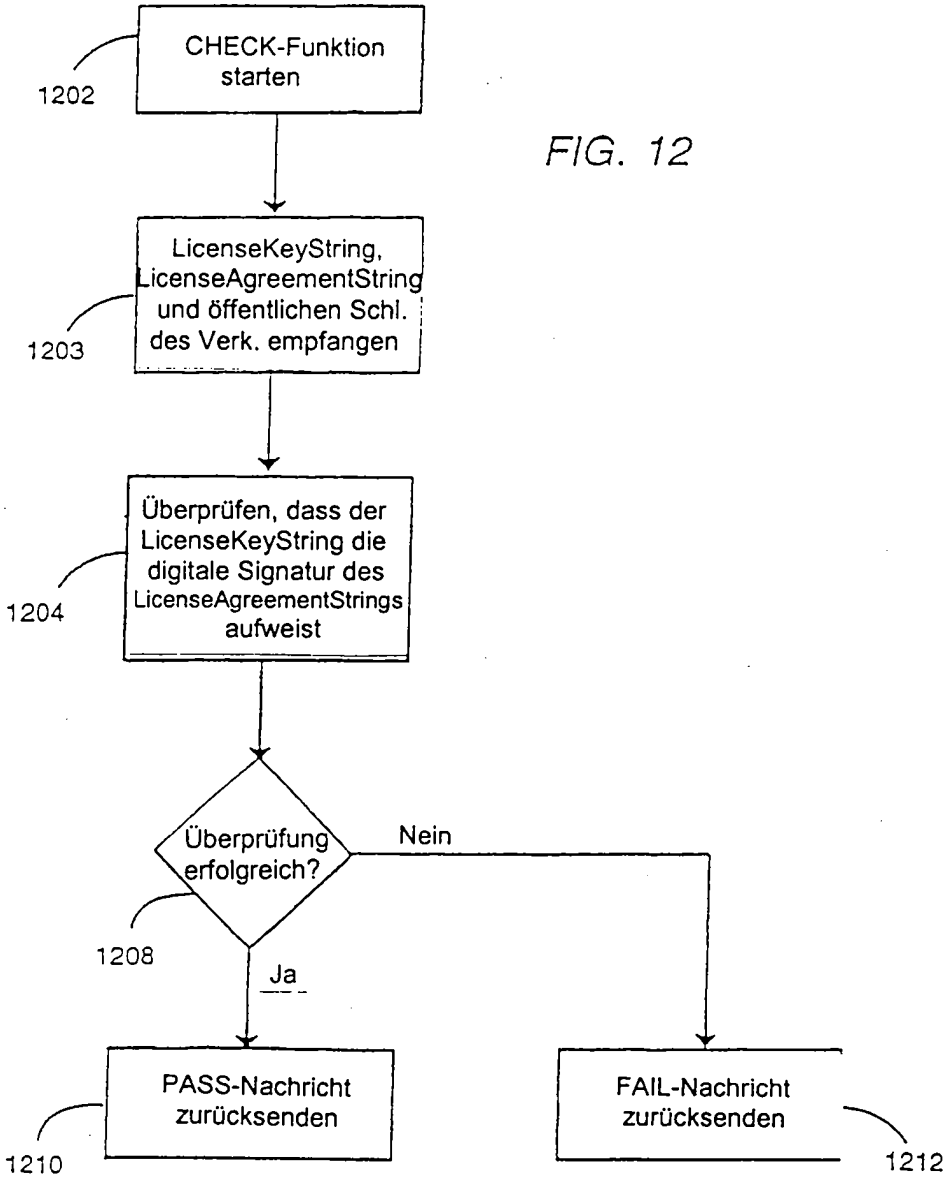

CHECK-Funktion starten
1202
LicenseKeyString, LicenseAgreementString und öffentlichen Schl. des Verk. empfangen
1203
Überprüfen, dass der LicenseKeyString die digitale Signatur des LicenseAgreementStrings aufweist
1204
Überprüfung erfolgreich?
1208
Nein
Ja
PASS-Nachricht zurücksenden
1210
FAIL-Nachricht zurücksenden
1212

FIG. 12

FIG. 13

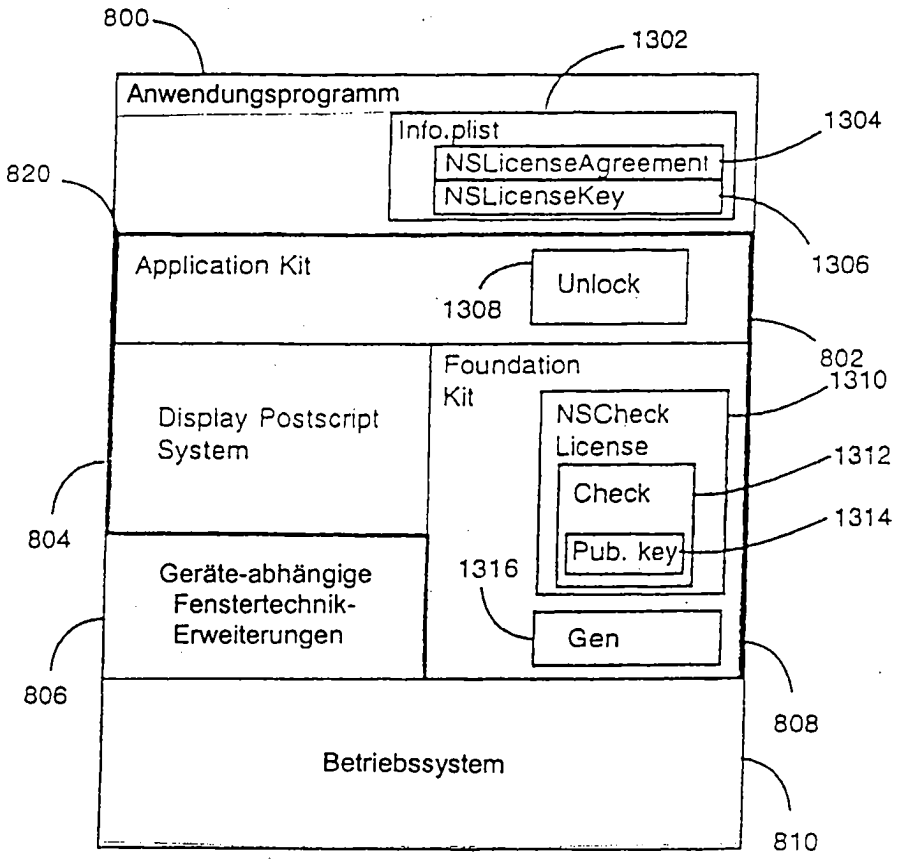
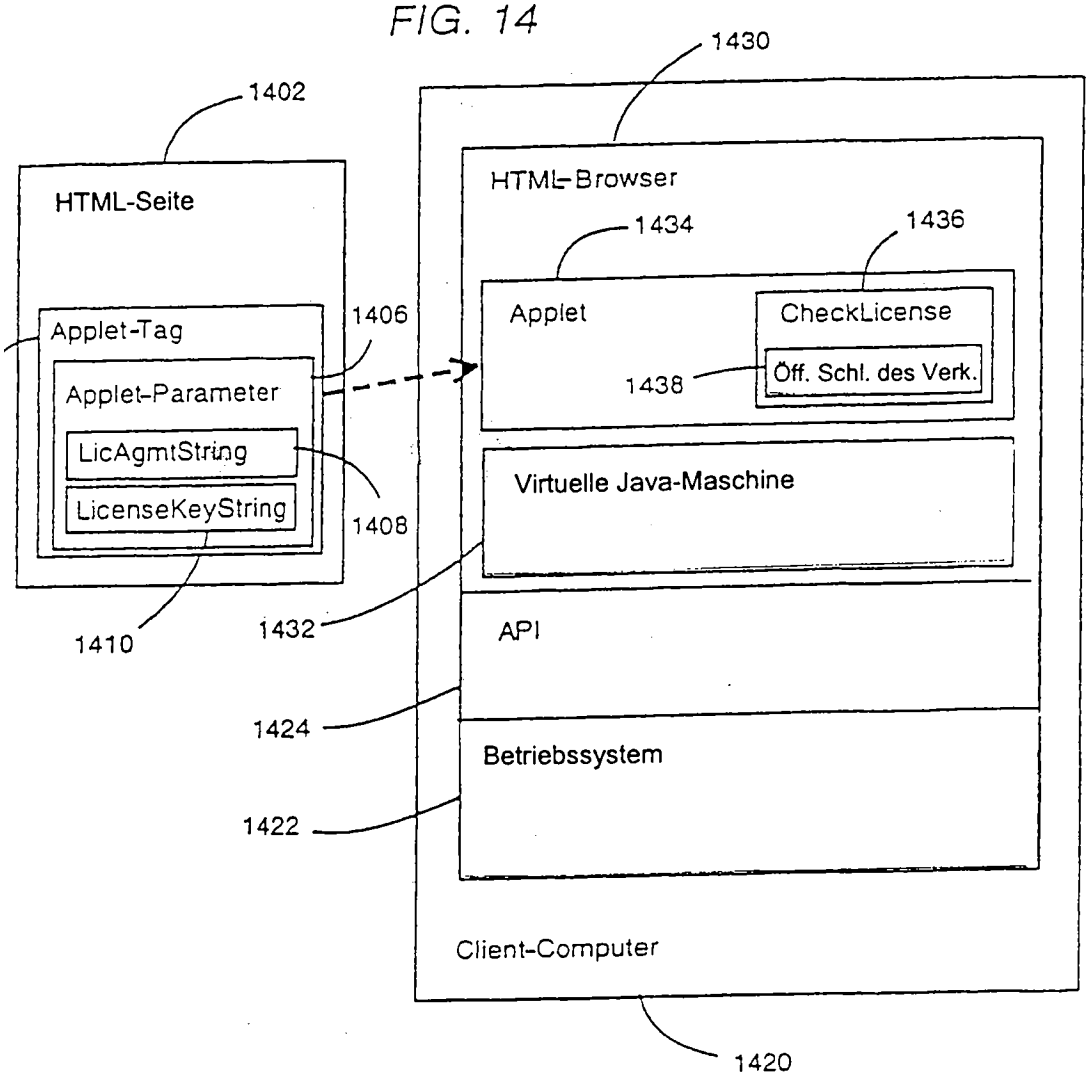

800
1302
Anwendungsprogramm
Info.plist
1304
NSLicenseAgreement
NSLicenseKey
820
1306
Application Kit
1308
Unlock
802
Foundation Kit
1310
NSCheck License
Display Postscript System
1312
Check
1314
Pub. key
804
1316
Geräte-abhängige Fenstertechnik-Erweiterungen
Gen
806
808
Betriebssystem
810

FIG. 14


1430
1402
HTML-Seite
HTML-Browser
1434
1436
Applet-Tag
1406
Applet
CheckLicense
Applet-Parameter
1438
Öff. Schl. des Verk.
LicAgmtString
Virtuelle Java-Maschine
LicenseKeyString
1408
1410
1432
API
1424
Betriebssystem
1422
Client-Computer
1420