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

(10) **DE 600 01 743 T2** 2004.02.05

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 192 532 B1**
(21) Deutsches Aktenzeichen: **600 01 743.5**
(86) PCT-Aktenzeichen: **PCT/US00/14624**
(96) Europäisches Aktenzeichen: **00 936 351.6**
(87) PCT-Veröffentlichungs-Nr.: **WO 00/72140**
(86) PCT-Anmeldetag: **25.05.2000**
(87) Veröffentlichungstag der PCT-Anmeldung: **30.11.2000**
(97) Erstveröffentlichung durch das EPA: **03.04.2002**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **19.03.2003**
(47) Veröffentlichungstag im Patentblatt: **05.02.2004**

(51) Int Cl.[7]: **G06F 9/00**

(30) Unionspriorität:
318518 **25.05.1999** **US**

(84) Benannte Vertragsstaaten:
AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

(73) Patentinhaber:
Oracle Corp., Redwood Shores, Calif., US

(72) Erfinder:
SAXENA, Sanjay, Fremont, US; HARRISON, Christopher, F-31370 Lautignac, FR

(74) Vertreter:
Wächtershäuser und Kollegen, 80333 München

(54) Bezeichnung: **ERWEITERUNG DER ATTRIBUTE EINES ANWENDUNGSPROGRAMMES HERGESTELLT MIT EINEM PROGRAMMIERWERKZEUG DER VIERTEN GENERATION**

Beschreibung

GEBIET DER ERFINDUNG

[0001] Die vorliegende Erfindung betrifft Computersysteme und genauer das Erweitern der Attribute eines Anwendungsprogramms, das unter Verwendung eines Programmierwerkzeugs der vierten Generation erstellt worden ist.

HINTERGRUND DER ERFINDUNG

[0002] Auch wenn Computer viele Aufgaben automatisiert haben, die vorher von Hand ausgeführt worden sind, hat die Arbeit der Computerprogrammierung erst seit ziemlich kurzer Zeit ein gewisses Maß an Automatisierung erfahren. Frühe Computerprogrammiersprachen erforderten, daß der Programmierer den gesamten Code eines Anwendungsprogramms manuell eingab, üblicherweise durch das Eintippen von Quellcode in eine Textdatei. Auch wenn eine solche manuelle Codeeingabe mühsam und fehleranfällig war, hatte sie den Vorteil, daß praktisch alle Aspekte der resultierenden Anwendungsprogramme gänzlich unter Kontrolle des Programmierers standen. Die Programmierer konnten ihre Programme auf jede von ihnen als geeignet erachtete Weise verändern, indem sie einfach den zugrundeliegenden Code veränderten.
[0003] Im Verlauf der Zeit sind Techniken entwickelt worden, um das Erstellen von Anwendungsprogrammen zu automatisieren. Insbesondere sind Werkzeuge entwickelt worden, die es Benutzern ermöglichen, Anwendungsprogramme zu entwerfen und automatisch Code zu erzeugen, der diese Anwendungsprogramme implementiert, indem die Benutzer einfach mit Benutzerschnittstellen-Bedienelementen interagieren. Solche Werkzeuge werden allgemein als Programmierwerkzeuge der vierten Generation (4GL) bezeichnet.
[0004] Beispielsweise kann ein 4GL-Programmierwerkzeug einem Anwendungsentwickler ein Fenster anzeigen, das die Benutzerschnittstelle darstellt, die das zu erzeugende Anwendungsprogramm seinen Benutzern anzeigen wird. Das 4GL-Programmierwerkzeug kann dem Anwendungsentwickler ferner Benutzerschnittstellen-Entwurfswerkzeuge bereitstellen, die es dem Anwendungsentwickler erlauben, übliche Komponenten von Benutzerschnittstellen visuell in das Fenster zu "malen". Beispielsweise können es die Benutzerschnittstellen-Entwurfswerkzeuge dem Anwendungsentwickler erlauben, Ankreuzfelder (check boxes), Menüs, Dialogfelder, Listenfelder und Schaltflächen (buttons) in das Fenster zu setzen, und sie können es dem Anwendungsentwickler erlauben, anzugeben, was geschehen soll, wenn der Benutzer mit diesen Komponenten interagiert.
[0005] Der Benutzer kann für jede Benutzerschnittstellenkomponente, die der Benutzer in dem Fenster angeordnet hat, eine "Eigenschaftspalette" ansehen. Die Eigenschaftspalette führt die vom Benutzer spezifizierbaren Eigenschaften der Benutzerschnittstellenkomponente und den gegenwärtigen Wert dieser Eigenschaften auf. Beispielsweise kann eine Eigenschaft für eine Schaltflächenkomponente die Eigenschaft "Etikett" (label) sein, wobei der der Etikett-Eigenschaft zugeordnete Wert der auf der Schaltfläche angezeigte Text ist.
[0006] Wenn der Anwendungsentwickler die Entwicklung des Anwendungsprogramms abgeschlossen hat, erzeugt das 4GL-Programmierwerkzeug automatisch Code für das Anwendungsprogramm. Die Gestalt des so erzeugten Codes kann von Implementierung zu Implementierung unterschiedlich sein. Beispielsweise kann das 4GL-Programmierwerkzeug maschinenausführbaren Code, Quellcode und/oder Metadaten erzeugen. Zum Zwecke der Erläuterung soll der Begriff "Anwendungscode" hier so verwendet werden, daß er die von einem 4GL-Programmierwerkzeug zur Darstellung der Anwendungsprogramme, die unter Verwendung des 4GL-Programmierwerkzeugs erstellt worden sind, erzeugten Daten bezeichnet, unabhängig von der Gestalt dieser Daten.
[0007] Das 4GL-Programmierwerkzeug wird häufig verwendet, um Anwendungscode zu erzeugen, der nicht unmittelbar durch einen Computer ausführbar ist, aber der, wenn er von einer bestimmten Laufzeitmaschine ausgeführt wird, das Anwendungsprogramm, das unter Verwendung des 4GL-Programmierwerkzeugs erstellt worden ist, implementiert. Zum Beispiel liest die Laufzeitmaschine den von dem 4GL-Programmierwerkzeug erzeugten Anwendungscode und "führt" den Anwendungscode "aus", indem das vorher von dem Anwendungsentwickler entworfene Fenster zusammen mit allen Benutzerschnittstellenkomponenten, die der Anwendungsentwickler dem Fenster hinzugefügt hat, angezeigt wird. Wenn der Benutzer mit diesen Benutzerschnittstellenkomponenten interagiert, führt das Anwendungsprogramm diejenigen Aktionen aus, die der Anwendungsentwickler unter Verwendung des 4GL-Programmierwerkzeugs diesen Komponenten zugeordnet hat.
[0008] 4GL-Programmierwerkzeuge machen die Entwicklung von Anwendungsprogrammen im Vergleich zu manuellen Codierungstechniken bei weitem einfacher und schneller und weniger fehleranfällig. Die durch die 4GL-Programmierung erzielte Leichtigkeit und Geschwindigkeit hat jedoch ihren Preis. Insbesondere geht die praktisch unbegrenzte Flexibilität und Kontrolle, die durch manuelle Codierung bereitgestellt wird, verloren. Stattdessen wird dem Benutzer eines 4GL-Programmierwerkzeugs typischerweise ein "vorgefertigter" Satz von Standard-Benutzerschnittstellenkomponenten bereitgestellt. Der Code, der diese Komponenten tatsächlich implementiert, ist typischerweise in einer kompilierten Bibliothek enthalten, die mit dem 4GL-Programmier-

werkzeug vertrieben wird, und der Benutzer kann daher im allgemeinen nicht auf den Code zugreifen. Demgemäß ist der Benutzer erstens auf den von dem 4GL-Programmierwerkzeug angebotenen Komponentensatz, zweitens auf die von diesen Komponenten bereitgestellten Eigenschaften und drittens auf die Funktionen beschränkt, die von den für diese Komponenten bereitgestellten Methoden ausgeführt werden. Schon geringe Abweichungen von der Norm, wie beispielsweise das Anzeigen einer ovalen Schaltfläche statt einer rechteckigen Schaltfläche, sind nicht möglich, wenn die von dem 4GL-Programmierwerkzeug bereitgestellten Komponenten keine vom Benutzer einstellbaren Eigenschaften aufweisen, die die Abweichungen zulassen. Beispielsweise müßte, damit ein 4GL-Programmierer ovale Schaltflächen benutzen kann, die von dem 4GL-Programmierwerkzeug bereitgestellte Schaltflächenkomponente eine "Form"-Eigenschaft bereitstellen, die von "Rechteck" nach "Oval" geändert werden kann. Wenn die Schaltflächenkomponente keine "Form"-Eigenschaft aufweist, oder wenn "Oval" kein für die "Form"-Eigenschaft unterstützter Wert ist, dann kann der Benutzer keine Anwendungsprogramme erstellen, die ovale Schaltflächen aufweisen.

[0009] Die Attribute und Verhaltensweisen der Benutzerschnittstelle (UI = user interface) und Softwarekomponenten, die ein 4GL-Programmierer in ein Anwendungsprogramm aufnehmen kann, sind in Objektklassen implementiert, die in einer mit dem 4GL-Programmierwerkzeug bereitgestellten Bibliothek geliefert werden. Die Bibliothek wird typischerweise in kompilierter Form bereitgestellt, so daß sie nicht editiert werden kann. Wenn der Benutzer ein Anwendungsprogramm erstellt, das beispielsweise eine Schaltfläche verwendet, erzeugt das 4GL-Programmierwerkzeug Code, der bei der Ausführung durch die Laufzeitmaschine dazu führt, daß Methoden in der "Schaltfläche"-Klasse in der mit dem 4GL-Programmierwerkzeug bereitgestellten Bibliothek aufgerufen werden. Wenn die "anzeigen"-Methode der Schaltflächenklasse so entworfen ist, daß sie eine rechteckige Schaltfläche anzeigt, dann wird eine rechteckige Schaltfläche auch dann angezeigt, wenn der Entwickler eine ovale Schaltfläche bevorzugen würde.

[0010] Aufgrund der obigen Ausführungen ist es offensichtlich wünschenswert, ein Verfahren und ein System bereitzustellen, die eine leichte und schnelle Entwicklung von Anwendungsprogrammen unter Verwendung von 4GL-Programmierwerkzeugen gestatten, aber größere Flexibilität als gegenwärtige 4GL-Programmierwerkzeuge zulassen. Insbesondere ist es wünschenswert, ein 4GL-Programmierwerkzeug bereitzustellen, das Benutzer nicht auf nur diejenigen Implementierungen von Komponenten beschränkt, die mit dem 4GL-Programmierwerkzeug geliefert werden.

ZUSAMMENFASSUNG DER ERFINDUNG

[0011] Gemäß einem Aspekt der Erfindung wird ein 4GL-Programmierwerkzeug bereitgestellt, das es Anwendungsentwicklern erlaubt, die Implementierungsklassen der in den Anwendungsprogrammen, welche die Anwendungsentwickler unter Verwendung des 4GL-Programmierwerkzeugs erzeugen, verwendeten Komponenten anzugeben. Die so angegebenen Implementierungsklassen können Attribute und Verhaltensweisen definieren, die von den entsprechenden Komponentenimplementierungen, die mit dem 4GL-Programmierwerkzeug bereitgestellt werden, nicht unterstützt werden. Während der Laufzeit werden Methoden in den angegebenen Implementierungsklassen aufgerufen. Um die Methoden von benutzerspezifizierten Implementierungsklassen korrekt aufzurufen, müssen die Methodenschnittstellen der Instanz, die die Aufrufe ausführt, bekannt sein. Vorzugsweise implementieren daher alle Komponentenimplementierungen, die von dem Anwendungsprogramm verwendet werden sollen, einschließlich sowohl der "Standard"-Komponentenimplementierungen, die mit dem 4GL-Programmierwerkzeug bereitgestellt werden, als auch der "Spezial"-Komponentenimplementierungen, die von dem Anwendungsentwickler angegeben werden, eine gemeinsame Schnittstelle.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0012] Die vorliegende Erfindung wird anhand eines Beispiels, und nicht auf beschränkende Weise, in den Figuren der beigefügten Zeichnungen veranschaulicht, in denen gleiche Bezugszeichen auf ähnliche Elemente verweisen, und in denen sind:

[0013] **Fig.** 1 ein Blockdiagramm eines 4GL-Anwendungsentwicklungssystems, das die Verwendung von benutzerspezifizierten Komponentenimplementierungen gemäß einem Ausführungsbeispiel der Erfindung gestattet; und

[0014] **Fig.** 2 ein Blockdiagramm, das ein Computersystem veranschaulicht, in dem Ausführungsbeispiele der vorliegenden Erfindung implementiert werden können.

GENAUE BESCHREIBUNG DER BEVORZUGTEN AUSFÜHRUNGSFORM

[0015] Eine Verfahren und eine Vorrichtung zum Erweitern der Attribute eines unter Verwendung eines 4GL-Programmierwerkzeugs erzeugten Anwendungsprogramms werden beschrieben. In der folgenden Beschreibung werden zum Zwecke der Erläuterung viele genaue Einzelheiten angeführt, um für ein gründliches

Verständnis der vorliegenden Erfindung zu sorgen. Für Fachleute auf dem Gebiet ist es jedoch klar, daß die vorliegende Erfindung ohne diese genauen Einzelheiten ausgeführt werden kann. In anderen Fällen werden gut bekannte Strukturen und Vorrichtungen als Blockdiagramme gezeigt, um eine unnötige Verschleierung der vorliegenden Erfindung zu vermeiden.

FUNKTIONSÜBERSICHT

[0016] Ein 4GL-Programmierwerkzeug wird bereitgestellt, das es Benutzern, ähnlich wie übliche 4GL-Programmierwerkzeuge, erlaubt, Anwendungsprogramme zu erstellen, ohne den Code für die Anwendungsprogramme manuell einzutippen. Gemäß einer Ausführungsform benutzen Benutzer das 4GL-Programmierwerkzeug, um Anwendungsprogramme visuell zu erstellen, indem Benutzerschnittstellenkomponenten ausgewählt werden, die ausgewählten Benutzerschnittstellenkomponenten in einem Fenster angeordnet werden, und Eigenschaftswerte für diese Komponenten angegeben werden. Im Unterschied zu üblichen 4GL-Programmierwerkzeugen ist jedoch eine Komponenteneigenschaft, die sich unter Kontrolle des Anwendungsentwicklers befindet, die Implementierungsklasse der Komponente. Der Wert, auf den der Anwendungsentwickler die Implementierungsklasseneigenschaft der Komponente setzt, bestimmt die Objektklasse, die die Eigenschaften und Methoden definiert, welche zur Laufzeit ausgeführt werden, um die Komponente zu implementieren. Die angegebene Implementierungsklasse für eine Schaltfläche kann beispielsweise die übliche mit dem 4GL-Programmierwerkzeug bereitgestellte Schaltflächenklasse sein, oder eine speziell entworfene Klasse für ovale Schaltflächen, die von dem Anwendungsentwickler oder von einem Komponentenentwickler als dritte Partei geschrieben worden ist.
[0017] Während der Laufzeit werden Methoden in der angegebenen Implementierungsklasse aufgerufen, um die Komponente zu instantiieren und um Eigenschaften der Komponente zu setzen und zu lesen. Um diese Methoden korrekt aufzurufen, müssen die Schnittstellen der Methoden derjenigen Instanz bekannt sein, die die Aufrufe ausführt. Daher implementiert gemäß einem Aspekt der Erfindung jede Komponentenimplementierung, die von dem Anwendungsprogramm verwendet werden soll, einschließlich sowohl der "Standard"-Komponentenimplementierungen in der mit dem 4GL-Programmierwerkzeug bereitgestellten Bibliothek als auch der "Spezial"-Komponentenimplementierungen, die von dem Anwendungsentwickler angegeben werden, eine gemeinsame Schnittstelle.

BEISPIELHAFTES 4GL-SYSTEM

[0018] **Fig.** 1 veranschaulicht ein 4GL-System **100** gemäß einer Ausgestaltung der Erfindung. Das 4GL-System **100** beinhaltet ein 4GL-Programmierwerkzeug **112**, das ein Anwendungsentwickler **110** zur Erzeugung von Anwendungscode **114**, der ein Anwendungsprogramm definiert, verwendet. Wie oben erwähnt, kann der Anwendungscode **114** eine Form aufweisen, die unmittelbar durch eine Maschine ausführbar ist, oder er kann eine Form aufweisen, die durch eine virtuelle Maschine ausführbar ist, oder er kann einfach aus Metadaten bestehen, die das Anwendungsprogramm beschreiben. In dem dargestellten Ausführungsbeispiel weist der Anwendungscode **114** eine Form auf, die von einer Laufzeitmaschine **116** verwendet wird, um die Erfindung zu implementieren.
[0019] Zur Laufzeit wird der Anwendungscode **114** von einer Laufzeitmaschine "ausgeführt". Je nach der Beschaffenheit des Anwendungscodes **114** kann der Vorgang des Ausführens des Anwendungscodes **114** beispielsweise das Interpretieren von Befehlen umfassen, die in dem Anwendungscode **114** enthalten sind, oder das Aufrufen von Routinen umfassen, die in dem Anwendungscode **114** beschriebene Anwendungskomponenten implementieren.
[0020] In dem dargestellten System **100** ist das Anwendungsprogramm gemäß einem Client/Server-Modell implementiert, wobei Teile des Anwendungsprogramms auf einem Server ablaufen und andere Teile des Anwendungsprogramms auf einem Client ablaufen. Typischerweise befinden sich die serverseitigen Komponenten auf einer anderen Maschine als die clientseitigen Komponenten, und die Kommunikation zwischen den beiden Komponentensätzen erfolgt über eine Netzwerkverbindung.
[0021] Die serverseitigen Komponenten des Anwendungsprogramms umfassen den Anwendungscode **114** und die Laufzeitmaschine **116**. Die clientseitigen Komponenten des Anwendungsprogramms umfasssen Komponenten, die die einem Anwendungsbenutzer **124** angezeigte Benutzerschnittstelle des Anwendungsprogramms implementieren. Diese Komponenten umfassen einen Verteiler **140**, Nachrichtenverarbeitungsmodule und Ansichten. Die dargestellte Ausführungsform ist mit drei Nachrichtenverarbeitungsmodulen **128**, **130** und **132** und drei Ansichten **134**, **136** und **138** gezeigt.
[0022] Jede Komponente der Benutzerschnittstelle des Anwendungsprogramms weist eine entsprechende Ansicht und ein Nachrichtenverarbeitungsmodul auf. Beispielsweise beinhalte die Benutzerschnittstelle des Anwendungsprogramms eine Schaltfläche (button) B1 und zwei Ankreuzfelder (check boxes) CB1 und CB2. Die Ansicht **134** und das Nachrichtenverarbeitungsmodul **128** können der Schaltfläche B1 entsprechen, die

Ansicht **136** und das Nachrichtenverarbeitungsmodul **130** können dem ersten Ankreuzfeld CB1 entsprechen, und die Ansicht **138** und das Verarbeitungsmodul **132** können dem zweiten Ankreuzfeld CB2 entsprechen.

[0023] Eine Ansicht ist eine Instanz der Implementierungsklasse einer Komponente. Somit enthält die einer Komponente entsprechende Ansicht denjenigen Code, der die Komponente implementiert. Beispielsweise ist die Ansicht **134** eine Instanz der Implementierungsklasse, die die Schaltfläche B1 implementiert. Ähnlich sind die Ansichten **136** und **138** Instanzen der Implementierungsklassen, die die Ankreuzfelder CB1 beziehungsweise CB2 implementieren. Die Ansichten **136** und **138** für die Ankreuzfelder CB1 und CB2 enthalten nicht den gleichen Code, wenn, zum Beispiel, CB1 unter der Verwendung der voreingestellten (default) Ankreuzfeld-Implementierungsklasse implementiert wird, während CB2 unter Verwendung einer vom Benutzer angegebenen Ankreuzfeld-Implementierungsklasse implementiert wird. Unter diesen Bedingungen ist der Code, der CB1 implementiert, typischerweise in einer mit dem 4GL-Programmierwerkzeug **112** vertriebenen Laufzeitbibliothek enthalten, während der Code, der CB2 implementiert, in einer getrennten Datei implementiert ist.

[0024] Jedem Komponententyp entspricht ein Typ eines Nachrichtenverarbeitungsmoduls. Der Typ des Nachrichtenverarbeitungsmoduls für einen gegebenen Komponententyp ist dazu bestimmt, Nachrichten des Typs, der dem Komponententyp zugeordnet ist, zu verarbeiten. Beispielsweise hört ein Schaltflächen-Nachrichtenverarbeitungsmodul auf den Typ von Nachrichten, den Schaltflächen empfangen, und verarbeitet solche Nachrichten, während ein Ankreuzfeld-Nachrichtenverarbeitungsmodul auf den Typ von Nachrichten hört, den Ankreuzfelder empfangen, und solche Nachrichten verarbeitet.

[0025] Das einer Komponente entsprechende Nachrichtenverarbeitungsmodul sendet Nachrichten, die sich auf mit der Komponente zusammenhängende Ereignisse beziehen, zurück zur Laufzeitmaschine **116**. Beispielsweise kann das Nachrichtenverarbeitungsmodul **128** erkennen, daß der Benutzer auf das Schaltfeld B1 geklickt hat. Ansprechend auf die Erkennung des Clicks sendet das Nachrichtenverarbeitungsmodul **128** eine Nachricht über den Verteiler **140** zur Laufzeitmaschine **116**, um die Laufzeitmaschine **116** darauf aufmerksam zu machen, daß die Schaltfläche B1 angeklickt worden ist.

[0026] Das einer Komponente entsprechende Nachrichtenverarbeitungsmodul ruft in Reaktion auf Nachrichten von der Laufzeitmaschine **116** auch Methoden auf, die von der Ansicht der Komponente bereitgestellt werden. Beispielsweise kann das Nachrichtenverarbeitungsmodul **128** auf eine Nachricht von der Laufzeitmaschine **116** reagieren, indem es eine von der Ansicht **134** bereitgestellte Methode aufruft, die die Farbe der Schaltfläche B1 ändert.

BETRIEBSWEISE DER LAUFZEITUMGEBUNG

[0027] Zur Laufzeit liest die Laufzeitmaschine **116** den vorher von dem 4GL-Programmierwerkzeug 112 erzeugten Anwendungscode **114** und führt diesen aus. Dieser Anwendungscode **114** weist Daten auf, die die Implementierungsklasse der Anwendungskomponenten angeben, für die der Anwendungsentwickler **110** eine Implementierungsklasse angegeben hat. Beispielsweise sei die voreingestellte Implementierungsklasse für Schaltflächen DefButton.class (VoreingestellteSchaltfläche.class). Während der Anwendungsentwicklung hat der Anwendungsentwickler **110** möglicherweise eine andere Implementierungsklasse für die Schaltfläche B1, die in der Benutzerschnittstelle des Anwendungsprogramms angezeigt werden soll, angegeben. Zum Zwecke der Erläuterung werde angenommen, daß der Anwendungsentwickler **110** die Implementierungsklasse der Schaltfläche B1 als MyButton.class (MeineSchaltfläche.class) angegeben hat. Unter diesen Bedingungen weist der Anwendungscode **114** Daten auf, die angeben, daß die Implementierungsklasse von B1 MyButton.class ist.

[0028] Während der Ausführung des Anwendungscodes **114** sendet die Laufzeitmaschine **116** Nachrichten an den Verteiler **140**, um die Erzeugung der in dem Anwendungscode **114** angegebenen Komponenten zu veranlassen. Zumindest für diejenigen Komponenten, die nicht die voreingestellten Klassenimplementierungen nutzen, weisen die von der Laufzeitmaschine **116** gesendeten Nachrichten Informationen auf, die die Implementierungsklasse angeben. Beispielsweise werde angenommen, daß die Laufzeitmaschine **116** eine Nachricht an den Verteiler **140** sendet, die die Erzeugung der Schaltfläche B1 unter Verwendung der Implementierungsklasse MyButton.class anfordert.

[0029] Wenn der Verteiler **140** eine Nachricht von der Laufzeitmaschine **116** empfängt, leitet der Verteiler **140** die Nachricht an dasjenige Nachrichtenverarbeitungsmodul weiter, das der in der Nachricht angegebenen Komponente entspricht. Wenn für die in der Nachricht bezeichnete Komponente noch kein Nachrichtenverarbeitungsmodul angelegt worden ist, dann instantiiert der Verteiler **140** ein Nachrichtenverarbeitungsmodul des geeigneten Komponententyps für die Komponente. Wenn beispielsweise die vom Verteiler **140** empfangene Nachricht die Erzeugung einer Schaltfläche (z. B. Schaltfläche B1) anfordert, und ein Nachrichtenverarbeitungsmodul für diese spezielle Schaltfläche noch nicht instantiiert worden ist, dann instantiiert der Verteiler **140** ein Nachrichtenverarbeitungsmodul vom Schaltflächentyp (z. B. Nachrichtenverarbeitungsmodul **128**) für die Schaltfläche. Nach der Instantiierung des geeigneten Nachrichtenverarbeitungsmoduls sendet der Verteiler **140** die Nachricht zum Nachrichtenverarbeitungsmodul.

[0030] Wenn ein Nachrichtenverarbeitungsmodul eine Nachricht zur Erzeugung einer Komponente empfängt, dann bestimmt das Nachrichtenverarbeitungsmodul auf Grundlage der in der Nachricht enthaltenen Informationen, ob die Komponente unter Verwendung der voreingestellten Klassenimplementierung implementiert wird. Wenn die Komponente unter Verwendung der voreingestellten Klassenimplementierung implementiert wird, dann ruft das Nachrichtenverarbeitungsmodul die von der voreingestellten Klassenimplementierung für diesen Komponententyp bereitgestellte Methode auf, die Komponenten instantiiert (d. h., den "Konstruktor"). Dies wird typischerweise ausgeführt, indem eine Methode in der mit dem 4GL-Programmierwerkzeug **112** bereitgestellten Laufzeitbibliothek aufgerufen wird. Wenn die Komponente unter Verwendung einer benutzerspezifizierten Klassenimplementierung implementiert wird, dann ruft das Nachrichtenverarbeitungsmodul den von der benutzerspezifizierten Klassenimplementierung bereitgestellten Konstruktor auf.

[0031] Beispielsweise ermittle das Nachrichtenverarbeitungsmodul **128**, ansprechend auf eine Nachricht, die die Instantiierung der Schaltfläche B1 anfordert, daß die Schaltfläche B1 unter Verwendung der benutzerspezifizierten Implementierungsklasse MyButton.class implementiert wird. Das Nachrichtenverarbeitungsmodul **128** würde dann den durch die Implementierungsklasse MyButton.class bereitgestellten Konstruktor aufrufen, um die Schaltfläche B1 zu instantiieren. Die Instantiierung der Schaltfläche B1 erzeugt eine Instanz der Klasse MyButton, die als Ansicht **134** gezeigt ist.

[0032] Nachdem eine clientseitige Komponente instantiiert worden ist, müssen die Eigenschaften der Komponente auf die vom Anwendungsentwickler **110** für die Komponente angegebenen Werte gesetzt werden. Die Werte für die Eigenschaften der Komponente spiegeln sich in dem von dem 4GL-Programmierwerkzeug **112** erzeugten Anwendungscode **114** wider. Nachdem die Instantiierung der Komponente veranlaßt worden ist, sendet die Laufzeitmaschine **116** Nachrichten an den Verteiler **140**, die die Werte der Eigenschaften der Komponente angeben. Der Verteiler **140** leitet diese Nachrichten an das der Komponente entsprechende Nachrichtenverarbeitungsmodul weiter, und das Nachrichtenverarbeitungsmodul ruft die geeignete Methode der Ansicht der Komponente auf, um die angegebene Eigenschaft auf den angegebenen Wert zu setzen.

[0033] Beispielsweise sei angenommen, daß der Anwendungsentwickler **110** der "Etikett"-Eigenschaft der Schaltfläche B1 den Wert "OK" zuweist. Unter diesen Voraussetzungen würde der von dem 4GL-Programmierwerkzeug 112 erzeugte Anwendungscode **114** angeben, daß die Etikett-Eigenschaft der Schaltfläche B1 auf den Wert "OK" gesetzt worden ist. Zur Laufzeit sendet die Laufzeitmaschine **116**, nachdem sie die Instantiierung der Schaltfläche B1 veranlaßt hat, eine Nachricht an den Verteiler **140**, die angibt, daß die Etikett-Eigenschaft der Schaltfläche B1 auf "OK" gesetzt werden soll. Der Verteiler **140** leitet diese Nachricht an das Nachrichtenverarbeitungsmodul **128** weiter, das die geeignete Routine der Ansicht **134** aufruft, um zu bewirken, daß die Etikett-Eigenschaft der Schaltfläche B1 auf "OK" gesetzt wird. Gemäß einem Ausführungsbeispiel wird dieser Vorgang für jede Eigenschaft der Schaltfläche B1, für die zur Entwicklungszeit ein Wert angegeben worden war, wiederholt.

[0034] Es ist bedeutsam, daß der von der Ansicht **134** in Reaktion auf das Setzen einer Eigenschaft auf einen Wert tatsächlich ausgeführte Vorgang völlig anders sein kann als der Vorgang, der von einer Instanz der voreingestellten Schaltflächenklasse in Reaktion auf das Setzen derselben Eigenschaft auf denselben Wert ausgeführt wird. Beispielsweise kann die voreingestellte Schaltflächenklasse DefButton bewirken, daß, ansprechend auf das Setzen der Etikett-Eigenschaft der Schaltfläche auf den Wert "OK", sich der Text "OK" auf der sichtbaren Fläche einer rechteckigen Schaltfläche befindet. Im Unterschied dazu kann die von einer benutzerspezifizierten Schaltflächenklasse MyButton implementierte Schaltfläche B1 bewirken, daß die Schaltfläche die Gestalt der Buchstaben "OK" aufweist, wenn die Etikett-Eigenschaft auf "OK" gesetzt wird.

[0035] Während der Ausführung des Anwendungsprogramms kann die Laufzeitmaschine **116** den gegenwärtigen Wert einer Eigenschaft einer Benutzerschnittstellenkomponente (UI-Komponente) benötigen. Um den Wert zu bestimmen, sendet die Laufzeitmaschine **116** eine Nachricht an den Verteiler **140**, die die Komponente bezeichnet und die Eigenschaft angibt, deren gegenwärtigen Wert die Laufzeitmaschine **116** benötigt. Der Verteiler **140** leitet die Nachricht an das geeignete Nachrichtenverarbeitungsmodul weiter, und das Nachrichtenverarbeitungsmodul reagiert, indem es die geeignete Methode derjenigen Ansicht aufruft, die der angegebenen Komponente entspricht, um den gegenwärtigen Wert der angegebenen Eigenschaft zu erhalten. Das Nachrichtenverarbeitungsmodul gibt den erhaltenen Wert zurück zum Verteiler **140**, der den Wert in einer Nachricht an die Laufzeitmaschine **116** zurücksendet.

GEMEINSAME SCHNITTSTELLE

[0036] Ein 4GL-Programmierwerkzeug stellt typischerweise für jeden von dem Anwendungsprogrammierer zur Verfügung gestellten Komponententyp erstens eine Klassenimplementierung für den Komponententyp und zweitens ein Nachrichtenverarbeitungsmodul, das mit der Klassenimplementierung zusammenarbeitet, bereit. Hersteller von 4GL-Programmierwerkzeugen können Nachrichtenverarbeitungsmodule, die mit ihren speziellen Klassenimplementierungen zusammenarbeiten, erstellen, weil sie die Lieferanten dieser Klassenimplementierungen sind und daher die Schnittstellen hierfür kennen und kontrollieren.

[0037] Wie oben beschrieben, erlauben es Ausgestaltungen der vorliegenden Erfindung Anwendungsprogrammierern, Klassenimplementierungen anzugeben, die anders als die von den 4GL-Programmierwerkzeugherstellern bereitgestellten Klassenimplementierungen sind. Gemäß einer Ausführungsform kann, auch wenn ein Benutzer eine nicht-voreingestellte Klassenimplementierung für eine Komponente angegeben hat, das von dem 4GL-Programmierwerkzeug für diesen Komponententyp bereitgestellte Nachrichtenverarbeitungsmodul mit der angegebenen Klassenimplementierung zusammenarbeiten.

[0038] Um mit benutzerspezifizierten Implementierungen von Komponentenklassen korrekt zu funktionieren, müssen Nachrichtenverarbeitungsmodule nicht nur die Klassendatei kennen, die die Klassenimplementierung der Komponentenklasse enthält, sondern auch die Schnittstelle der von den Komponentenklassen implementierten Methoden. Beispielsweise muß das Nachrichtenverarbeitungsmodul **128**, um die Etikett-Eigenschaft der Schaltfläche B1 zu setzen, die Schnittstelle der Methode in der Ansicht **134** kennen, die den Wert der Etikett-Eigenschaft setzt.

[0039] Gemäß einer Ausführungsform kennen die Nachrichtenverarbeitungsmodule die Schnittstelle der von benutzerspezifizierten Komponentenklassen implementierten Methoden, weil alle Implementierungen von Komponentenklassen, einschließlich sowohl der voreingestellten Klassenimplementierungen als auch der benutzerspezifizierten Klassenimplementierungen, so entworfen sind, daß sie eine gemeinsame Schnittstelle implementieren, die hier als die IView-Schnittstelle (View = Ansicht) bezeichnet wird. Die Nachrichtenverarbeitungsmodule sind ihrerseits so entworfen, daß sie Aufrufe von Ansichten über die IView-Schnittstelle ausführen. Somit sind die Nachrichtenverarbeitungsmodule, auch wenn es zu der Zeit, zu der die Nachrichtenverarbeitungsmodule entworfen werden, nicht bekannt ist, mit welchen Klassenimplementierungen die Nachrichtenverarbeitungsmodule zusammenarbeiten werden müssen, dennoch in der Lage, mit jedweden vom Anwendungsentwickler **110** angegebenen Klassenimplementierungen zusammenzuarbeiten.

[0040] Gemäß einer Ausführungsform sind alle benutzerspezifizierten Klassenimplementierungen so entworfen, daß sie die folgende Schnittstelle (IView) unterstützen:

```
public void init (IHandler handler);
public void destroy();
public Object getProperty(PropertyID id);
public boolean setProperty(PropertyID id, Object value);
public void addListener(Class type, EventListener listener);
public void removeListener(Class type, EventListener listener);
public void paint(Graphics g);
public void repaint(Rectangle r);
public void add (Object child, int index);
public void remove(Object child);
public void removeAll();
```

[0041] In der obigen Schnittstelle weisen einige Methoden Parameter des Typs IHandler (handler = Verarbeitungsmodul) auf. IHandler ist ein Datentyp zum Speichern eines Wertes, der eine bestimmte Nachrichtenverarbeitungsmodulinstanz angibt. Wie oben erwähnt, ist eine Instanz, die Aufrufe der Methoden in der IView-Schnittstelle ausführt, eine Nachrichtenverarbeitungsmodulinstanz. Somit übergibt für diejenigen Methoden, die erfordern, daß ein IHandler als Eingabevariable übergeben wird, die Nachrichtenverarbeitungsmodulinstanz, die den Aufruf ausführt, typischerweise Daten, die die Instanz selber angeben. Gemäß einer Ausführungsform werden, für jede Komponente, die von der Klassenimplementierung für diese Komponente bereitgestellten Methoden der IView-Schnittstelle von den Nachrichtenverarbeitungsmodul für diese Komponente wie folgt aufgerufen:

[0042] Die "init"-Methode (init = initialisere) wird unmittelbar nach der Erzeugung der Komponente aufgerufen. Die "init"-Methode übergibt der Komponente einen Verweis auf ihr Nachrichtenverarbeitungsmodul und gibt der Komponente Gelegenheit, jedwede benötigte Initialisierung auszuführen.

[0043] Die "destroy"-Methode (destroy = vernichte) wird aufgerufen, wenn eine Komponente nicht weiter benötigt wird. Diese Methode gibt der Komponente Gelegenheit, alle Systemressourcen freizugeben, die die Komponente hält.

[0044] Die "getProperty"-Methode (getProperty = erhalteEigenschaft) gibt den Wert der angeforderten Eigenschaft zurück. Gemäß einer Ausführungsform sind alle Komponentenimplementierungen für einen bestimmten Komponententyp so entworfen, daß sie bestimmte Eigenschaften unterstützen. Wenn die angegebene Eigenschaft von der Komponente nicht unterstützt wird, gibt diese getProperty-Methode Property.NOT_SUPPORTED (Eigenschaft.NICHT_UNTERSTÜTZT) zurück.

[0045] Die "setProperty"-Methode (setProperty = setzeEigenschaft) setzt den Wert (value) der angegebenen Eigenschaft. Wenn die angegebene Eigenschaft von der Komponente nicht unterstützt wird, gibt diese Methode den Wert Falsch zurück, andernfalls gibt sie den Wert Wahr zurück.

[0046] Die "addListener"-Methode (addListener = fügeZuhörerHinzu) fügt einen Zuhörer (listener) des angegebenen Typs (type) hinzu. Gemäß einer Ausgestaltung sind alle Komponentenimplementierungen für einen

bestimmten Komponententyp so entworfen, daß sie bestimmte Zuhörertypen unterstützen.

[0047] Die "removeListener"-Methode (removeListener = entferneZuhörer) entfernt einen Zuhörer (listener) des angegebenen Typs (type).

[0048] In der "paint"-Methode (paint = male, stelle dar) muß sich die Ansicht (view) unter Verwendung des in dem Parameter angegebenen "Graphics"-Objekts (graphics = Grafik) selbst darstellen. Gemäß einer Ausführungsform können Komponenten andere Komponenten umfassen. Eine Komponente, die andere Komponenten enthält, wird als "Behälter" (container) bezeichnet. Für Komponenten, die Teil des Behälters sind, wird die "paint"-Methode von dem Behälter der Komponente aufgerufen. Für andere Komponenten wird die "paint"-Methode von dem Nachrichtenverarbeitungsmodul der Komponente aufgerufen.

[0049] In der "repaint"-Methode (repaint = maleNeu) setzt die Ansicht das bereitgestellte Rechteck (rectangle) auf ungültig. Wenn das Rechteck den Wert Null hat, wird das gesamte Objekt auf ungültig gesetzt.

[0050] Die "add"-Methode (add = fügeHinzu) fügt eine Kind-Komponente (child) einer anderen Komponente hinzu. Für Behälter hat das Kind den Typ IView, wobei IView die Klasse ist, von der alle Ansichten abgeleitet sind. Für Listen hat das Kind den Typ String (string = Kette). Wenn der Index –1 beträgt, wird das Kind am Ende der Liste hinzugefügt. Wenn der Index größer als die gegenwärtige Anzahl von Kindern oder kleiner als –1 ist, schlägt die Methode fehl.

[0051] Die "remove"-Methode (remove = entferne) entfernt ein Kind (child) von dem angegebenen IView. Für Behälter weist das Kind den Typ IView auf. Für Listen weist das Kind den Typ String auf.

[0052] Die "removeAll"-Methode (removeAll = alle entfernen) entfernt alle Kinder aus dem angegebenen IView.

[0053] Wenn ein Anwendungsentwickler wünscht, eine Klasse zu verwenden, die die IView-Schnittstelle nicht unterstützt, kann der Anwendungsentwickler einfach eine Unterklasse dieser Klasse definieren, wobei die Unterklasse die IView-Schnittstelle unterstützt. Beispielsweise werde angenommen, daß der Anwendungsentwickler ein Anwendungsprogramm zu erzeugen wünscht, das die Schaltfläche B1 als eine Instanz einer von dritter Seite bereitgestellten Klasse CoolButton (cooleSchaltfläche) implementiert, wobei die CoolButton-Klasse die IView-Schnittstelle nicht implementiert. Unter diesen Umständen deklariert der Anwendungsentwickler erstens eine Unterklasse von CoolButton, die die IView-Schnittstelle implementiert, und gibt zweitens die Implementierungen jeder Methode der IView-Schnittstelle an. Das Folgende ist ein Beispiel, wie eine Schaltflächenklasse MyButton, die eine die IView-Schnittstelle implementierende Unterklasse ist, deklariert werden kann:

```
public class MyButton                    (öffentliche Klasse MyButton)
        extends CoolButton               (erweitert CoolButton)
        implements IView                 (implementiert IView)
{
[Implementierungen von IView-Methoden]
}
```

[0054] Die "Implementierungen der IView-Methoden" wandeln allgemein Aufrufe, die über die IView-Schnittstelle erfolgen, in Aufrufe um, die über die von CoolButton unterstützte Schnittstelle erfolgen. Beispielsweise wird in einer Ausführungsform die Methode setProperty(id, value) der IView-Schnittstelle zum Setzen aller Eigenschaften einer Ansicht benutzt. Somit hätte ein Aufruf, um das Etikett einer Schaltfläche über die IView-Schnittstelle zu setzen, die Gestalt setProperty(Etikett, Wert). Auf ähnliche Weise hätte ein Aufruf, um die Farbe einer Schaltfläche über die IView-Schnittstelle zu setzen, die Gestalt setProperty(Farbe, Wert). Die CoolButton-Klasse kann jedoch eine Methode "setText(value)" (setText = setzeText) implementieren, um das Etikett einer Schaltfläche zu setzen, und eine Methode "setColor(value)" (setColor = setzeFarbe), um die Farbe einer Schaltfläche zu setzen. Unter diesen Umständen kann die von der MyButton-Klasse für die setProperty-Methode der IView-Schnittstelle bereitgestellte Implementierung beispielsweise wie folgt lauten:

```
setProperty(id, value)
{
switch(id)
case Label:
        {
        MyButton.setText(value);
        break;
        }
case Color:
        {
        MyButton.setColor(value);
        break;
        }
}
```

[0055] Somit wird ein Aufruf von setProperty über die IView-Schnittstelle in einen Aufruf von setText umgewandelt, wenn der Wert der an setProperty übergebenen Variablen id "label" (Etikett) lautet, und der Aufruf wird in einen Aufruf von setColor umgesetzt, wenn der an setProperty übergebene Wert der Variablen id "color" (Farbe) lautet.

AUSFÜHRUNGSALTERNATIVEN

[0056] Zum Zwecke der Erläuterung sind ausführungsspezifische Einzelheiten in die obige Beschreibung aufgenommen worden. Diese Einzelheiten sind jedoch nicht erforderlich, um die Erfindung auszuführen, und sie sollen die Erfindung in keiner Weise beschränken. Beispielweise kann die Erfindung unter Verwendung von Ausführungsalternativen ausgeführt werden, von denen einige nun beschrieben werden sollen.

[0057] In der oben beschriebenen Ausführungsform stellen alle Komponentenimplementierungen den Nachrichtenverarbeitungsmodulen die gleiche Schnittstelle bereit. In Ausführungsalternativen unterstützen jedoch alle benutzerspezifizierten Klassenimplementierungen für jeden gegebenen Komponententyp die gleiche Schnittstelle wie die voreingestellte Klassenimplementierung des gegebenen Komponententyps, aber diese Schnittstellen können sich für unterschiedliche Komponententypen unterscheiden. Beispielsweise würde eine benutzerspezifizierte Klassenimplementierung für eine Schaltfläche die gleiche Schnittstelle wie die voreingestellte Klassenimplementierung für Schaltflächen implementieren (und daher in der Lage sein, mit dem Nachrichtenverarbeitungsmodul für Schaltflächen korrekt zu funktionieren), und eine benutzerspezifizierte Klassenimplementierung für ein Ankreuzfeld würde dieselbe Schnittstelle wie die voreingestellte Klassenimplementierung für ein Ankreuzfeld implementieren (und daher in der Lage sein, mit dem Nachrichtenverarbeitungsmodul für Ankreuzfelder korrekt zu funktionieren). Die Schnittstelle für die Schaltflächenimplementierungen könnte sich jedoch von der Schnittstelle für die Ankreuzfeldimplementierungen unterscheiden.

[0058] In den oben beschriebenen Ausführungsformen sind Benutzerschnittstellenkomponenten (z. B. Schaltflächen und Ankreuzfelder) als Beispiele für die Komponententypen gegeben worden, die einem 4GL-Programmierer zur Verfügung gestellt werden und für die der 4GL-Programmierer nun nicht-voreingestellte Implementierungen angeben kann. Benutzerschnittstellenkomponenten sind jedoch lediglich ein Typ der von 4GL-Programmierwerkzeugen bereitgestellten Komponenten. Andere Komponententypen umfassen beispielsweise Komponenten, die Datenquellen darstellen. Unter Verwendung der hier beschriebenen Techniken kann ein 4GL-Programmierwerkzeug bereitgestellt werden, das es Benutzern gestattet, nicht-voreingestellte Implementierungen für irgendeinen Typ von Anwendungskomponenten anzugeben, einschließlich solcher, die kein sichtbarer Bestandteil der Benutzerschnittstelle des Anwendungsprogramms sind. Auf ähnliche Weise können die Komponenten, für die 4GL-Programmierer nicht-voreingestellte Implementierungsklassen angeben können, sowohl serverseitig ausgeführte als auch clientseitig ausgeführte Komponenten sein.

HARDWAREÜBERBLICK

[0059] **Fig.** 2 ist ein Blockschaltbild, das ein Computersystem **200** veranschaulicht, in dem eine Ausführungsform der Erfindung implementiert werden kann. Das Computersystem **200** weist einen Bus **202** oder einen anderen Kommunikationsmechanismus zum Übermitteln von Informationen und einem mit dem Bus **202** gekoppelten Prozessor **204** zum Verarbeiten von Informationen auf. Das Computersystem **200** umfaßt auch einen Hauptspeicher **206**, wie beispielsweise einen Speicher mit wahlfreiem Zugriff (RAM) oder eine andere dynamische Speichereinrichtung, der mit dem Bus **202** gekoppelt ist, um Informationen und Befehle, die vom Prozessor **204** ausgeführt werden sollen, zu speichern. Der Hauptspeicher **206** kann auch verwendet werden, um während der Ausführung von Befehlen, die von dem Prozessor **204** ausgeführt werden sollen, temporäre Variablen und andere Zwischeninformationen zu speichern. Das Computersystem **200** weist ferner einen Nur-Lese-Speicher (ROM) **208** oder eine andere statische Speichereinrichtung auf, der/die mit dem Bus **202** gekoppelt ist, um statische Instruktionen und Befehle für den Prozessor **204** zu speichern. Eine Speichereinrichtung **210**, wie beispielsweise eine magnetische Platte oder optische Platte, ist vorgesehen und mit dem Bus **202** gekoppelt, um Informationen und Befehle zu speichern.

[0060] Das Computersystem **200** kann über den Bus **202** mit einer Anzeige **212** gekoppelt sein, beispielsweise einer Elektronenstrahlröhre (CRT = cathode ray tube), um Informationen einem Computerbenutzer anzuzeigen. Eine Eingabeeinrichtung **214**, die alphanumerische und andere Tasten aufweist, ist mit dem Bus **202** gekoppelt, um dem Prozessor **204** Informationen und Befehlsauswahlen zu übermitteln. Eine andere Art von Benutzereingabeeinrichtung ist eine Cursorsteuerung **216**, wie beispielsweise eine Maus, eine Rollkugel oder Cursorrichtungstasten, um Richtungsinformationen und Befehlsauswahlen an den Prozessor **204** zu übermitteln und um die Cursorbewegung auf der Anzeige **212** zu steuern. Diese Eingabeeinrichtung weist typischerweise zwei Freiheitsgrade in zwei Achsen auf, nämlich einer ersten Achse (z. B. x) und einer zweiten Achse (z. B. y), was es der Einrichtung ermöglicht, Positionen in einer Ebene anzugeben.

[0061] Die Erfindung betrifft die Verwendung des Computersystems **200** zum Entwurf von 4GL-Anwendungsprogrammen. Gemäß einer Ausführungsform der Erfindung wird durch das Computersystem **200** ein 4GL-Programmierwerkzeug bereitgestellt, das es Anwendungsprogrammierern erlaubt, nicht-voreingestellte Implementierungen von Komponentenklassen anzugeben, und zwar ansprechend darauf, daß der Prozessor **204** eine Sequenz oder mehrere Sequenzen mit einem Befehl oder mehreren Befehlen ausführt, die im Hauptspeicher **206** enthalten sind. Solche Befehle können von einem anderen computerlesbaren Medium, wie beispielsweise der Speichereinrichtung **210**, in den Hauptspeicher **206** eingelesen werden. Die Ausführung der im Hauptspeicher **206** enthaltenen Befehlssequenzen veranlaßt den Prozessor **204**, die hier beschriebenen Verfahrensschritte auszuführen. In Ausführungsalternativen kann eine festverdrahtete Schaltung statt oder in Kombination mit Softwarebefehlen verwendet werden, um die Erfindung zu implementieren. Daher sind Ausgestaltungen der Erfindung nicht auf irgendeine bestimmte Kombination von Hardwareschaltungen und Software beschränkt.

[0062] Der hier verwendete Begriff "computerlesbares Medium" bezieht sich auf jedwedes Medium, das am Bereitstellen von Befehlen an den Prozessor **204** zur Ausführung beteiligt ist. Ein solches Medium kann viele Formen haben, die nicht-flüchtige Medien, flüchtige Medien und Übertragungsmedien einschließen, aber nicht darauf beschränkt sind. Nicht-flüchtige Medien umfassen beispielsweise optische oder magnetische Platten, wie beispielsweise die Speichereinrichtung **210**. Flüchtige Medien umfassen dynamischen Speicher, wie beispielsweise den Hauptspeicher **206**. Übertragungsmedien umfassen Koaxialkabel, Kupferkabel und Faseroptiken, einschließlich der Leitungen, die den Bus **202** enthalten. Übertragungsmedien können auch die Gestalt von akustischen Wellen oder Lichtwellen, wie beispielsweise den bei Funk- und Infrarot-Datenübertragungsvorgängen erzeugten, einnehmen.

[0063] Übliche Formen computerlesbarer Medien umfassen beispielsweise eine Diskette, eine flexible Platte, Festplatte, Magnetband oder irgendein anderes magnetisches Medium, eine CD-ROM, irgendein anderes optisches Medium, Lochkarten, Lochstreifen, irgendein anderes physisches Medium mit Lochmustern, ein RAM, ein PROM, ein EPROM, ein FLASH-EPROM, irgendeinen anderen Speicherchip oder ein anderes Speichermodul, eine Trägerwelle wie unten beschrieben, oder irgendein anderes Medium, von dem ein Computer lesen kann.

[0064] Unterschiedliche Formen computerlesbarer Medien können bei der Übermittlung einer Sequenz oder mehrerer Sequenzen mit einem Befehl oder mehreren Befehlen an den Prozessor **204** zur Ausführung beteiligt sein. Beispielsweise können die Befehle anfangs auf einer Magnetplatte eines entfernten Computers enthalten sein. Der entfernte Computer kann die Befehle in seinen dynamischen Speicher laden und die Befehle unter Verwendung eines Modems über eine Telefonleitung senden. Ein lokal beim Computersystem **200** angeordnetes Modem kann die Daten über die Telefonleitung empfangen und einen Infrarot-Sender benutzen, um die Daten in ein Infrarot-Signal umzuwandeln. Ein Infrarot-Detektor kann die in dem Infrarot-Signal übertragenen Daten empfangen, und eine geeignete Schaltung kann die Daten auf den Bus **202** geben. Der Bus **202** überträgt die Daten in den Hauptspeicher **206**, von wo der Prozessor **204** auf die Befehle zugreift und sie ausführt. Die

Befehle, die der Hauptspeicher **206** erhält, können optional in der Speichereinrichtung **210** gespeichert werden, und zwar entweder vor oder nach der Ausführung durch den Prozessor **204**.

[0065] Das Computersystem **200** weist auch eine mit dem Bus **202** gekoppelte Kommunikationsschnittstelle **218** auf. Die Kommunikationsschnittstelle **218** stellt eine Datenkommunikationsverbindung in beide Richtungen mit einer Netzwerkverbindung **220** bereit, die mit einem lokalen Netzwerk **222** verbunden ist. Beispielsweise kann die Kommunikationsschnittstelle **218** eine Karte für ein Digitalnetzwerk mit integrierten Diensten (ISDN) oder ein Modem sein, um eine Datenübertragungsverbindung für eine Telefonleitung entsprechenden Typs bereitzustellen. Als weiteres Beispiel kann die Kommunikationsschnittstelle **218** eine Karte für ein lokales Netzwerk (LAN) sein, um eine Datenkommunikationsverbindung zu einem passenden LAN bereitzustellen. Auch drahtlose Verbindungen können implementiert werden. In jeder solchen Implementierung sendet und empfängt die Kommunikationsschnittstelle **218** elektrische, elektromagnetische oder optische Signale, die digitale Datenströme übertragen, welche unterschiedliche Arten von Informationen darstellen.

[0066] Die Netzwerkverbindung **220** sorgt typischerweise für eine Datenübermittlung über ein Netzwerk oder mehrere Netzwerke zu anderen Datenvorrichtungen. Beispielsweise kann die Netzwerkverbindung **220** eine Verbindung über das lokale Netzwerk **222** mit einem Hintergrundcomputer **224** oder einer von einem Internet-Dienstleister (ISP = internet service provider) **226** betriebenen Datenausrüstung bereitstellen. Der ISP **226** stellt seinerseits Datenübertragungsdienste über das weltweite Datenpaket-Kommunikationsnetzwerk bereit, das nun allgemein als das "Internet" **228** bezeichnet wird. Sowohl das lokale Netzwerk **222** als auch das Internet **228** benutzen elektrische, elektromagnetische oder optische Signale, die digitale Datenströme übertragen. Die Signale über die unterschiedlichen Netzwerke und die Signale auf der Netzwerkverbindung **220** und durch die Kommunikationsschnittstelle **218**, die die digitalen Daten an und von dem Computersystem **200** übertragen, sind beispielhafte Formen von Trägerwellen, die die Informationen übertragen.

[0067] Das Computersystem **200** kann Nachrichten senden und Daten, einschließlich Progammcode, empfangen, und zwar über das Netzwerk oder die Netzwerke, die Netzwerkverbindung **220** und die Kommunikationsschnittstelle **218**. Bei dem Internet-Beispiel kann ein Server **230** einen angeforderten Code für ein Anwendungsprogramm über das Internet **228**, den ISP **226**, das lokale Netzwerk **222** und die Kommunikationsschnittstelle **218** übertragen. Gemäß der Erfindung stellt ein solches heruntergeladenes Anwendungsprogramm ein 4GL-Programmierwerkzeug wie hier beschrieben bereit.

[0068] Der empfangene Code kann von dem Prozessor **204** beim Empfang ausgeführt werden, und/oder er kann in der Speichereinrichtung **210** oder in einem anderen nicht-flüchtigen Speicher zur späteren Ausführung gespeichert werden. Auf diese Weise kann das Computersystem **200** Anwendungscode in Gestalt einer Trägerwelle erhalten.

[0069] In der obigen Beschreibung ist die Erfindung unter Hinweis auf spezifische Ausführungsformen der Erfindung beschrieben worden. Es ist jedoch offensichtlich, daß unterschiedliche Abwandlungen und Änderungen vorgenommen werden können, ohne das Gebiet der Erfindung, wie es in den beigefügten Ansprüchen definiert ist, zu verlassen. Die Beschreibung und die Zeichnungen sollen demgemäß im Sinne einer Veranschaulichung und nicht im Sinne einer Einschränkung angesehen werden.

Patentansprüche

1. Verfahren zum Ausführen eines von einem 4GL-Programmierwerkzeug (**112**) erstellten Anwendungsprogramms, wobei das Verfahren die Schritte aufweist:

Einlesen, von Anwendungscode (**114**), der von dem 4GL-Programmierwerkzeug (**112**) erstellt worden ist, eines Abschnitts des Anwendungscodes (**114**), der eine Komponente für das Anwendungsprogramm angibt, wobei die Komponente einen bestimmten Komponententyp hat, für den das 4GL-Programmierwerkzeug (**112**) eine erste Implementierung bereitstellt; und

Ausführen des Abschnitts des Anwendungscodes, indem ein Konstruktor aufgerufen wird, um die Komponente in dem Anwendungsprogramm zu erzeugen,

dadurch gekennzeichnet, daß der Konstruktor in einer zweiten Implementierung für die durch den Abschnitt des Anwendungscodes angegebenen Komponente implementiert ist, wobei sich die zweite Implementierung von der ersten Implementierung unterscheidet und von dem 4GL-Programmierwerkzeug nicht bereitgestellt wird.

2. Verfahren nach Anspruch 1, bei dem:

der Schritt des Einlesens von einer Laufzeitmaschine (**116**) ausgeführt wird, die sich auf einem Server befindet;

der Schritt des Aufrufens des Konstruktors von einem Client in Reaktion auf den Eingang einer Nachricht bei dem Client ausgeführt wird, wobei die Nachricht von der Laufzeitmaschine in Reaktion auf die Ausführung des Abschnitts des Anwendungscodes (**114**) erzeugt wird.

3. Verfahren nach Anspruch 2, bei dem:

sowohl die erste Implementierung als auch die zweite Implementierung eine gemeinsame Schnittstelle unterstützen; und
der Schritt des Aufrufens des Konstruktors über die gemeinsame Schnittstelle von einem Nachrichtenverarbeitungsmodul (**128**, **130**, **132**) für den bestimmten Komponententyp ausgeführt wird.

4. Verfahren nach Anspruch 1, bei dem:
das 4GL-Programmierwerkzeug (**112**) den Abschnitt des Anwendungscodes (**114**) erzeugt, und zwar in Reaktion darauf, daß
ein Anwendungsprogrammierer (**110**) eine Eingabe durchführt, die angibt, daß die Anwendung eine Komponente des bestimmten Komponententyps aufweisen soll; und
der Anwendungsprogrammierer (**110**) eine Eingabe durchführt, die angibt, daß die Komponente unter Verwendung der zweiten Implementierung implementiert werden soll.

5. Verfahren nach Anspruch 1, bei dem:
die Laufzeitmaschine (**116**) den Abschnitt des Anwendungscodes ausführt, indem sie eine Nachricht an einen Verteiler (**140**) sendet;
der Verteiler (**140**) auf die Nachricht antwortet, indem er ein Nachrichtenverarbeitungsmodul (**128**, **130**, **132**) für den bestimmten Komponententyp instantiiert; und
das Nachrichtenverarbeitungsmodul den Konstruktor aufruft.

6. Verfahren nach Anspruch 5, bei dem:
sich die Laufzeitmaschine (**116**) auf einem Server befindet;
der Verteiler (**140**) sich auf einem über ein Netzwerk operativ mit dem Server gekoppelten Client befindet; und
die Laufzeitmaschine die Nachricht über das Netzwerk an den Verteiler sendet.

7. Verfahren nach Anspruch 1, ferner mit den Schritten:
Einlesen, von dem von dem 4GL-Programmierwerkzeug (**112**) erzeugten Anwendungscode (**114**), eines zweiten Abschnitts des Anwendungscodes, der
a) eine zweite Komponente für das Anwendungsprogramm angibt, wobei die zweite Komponente einen zweiten bestimmten Komponententyp aufweist, für den das 4GL-Programmierwerkzeug eine dritte Implementierung bereitstellt, wobei der zweite Komponententyp sich von dem ersten bestimmten Komponententyp unterscheidet; und
b) eine vierte Implementierung für die zweite Komponente angibt, die sich von der dritten Implementierung unterscheidet;
Ausführen des zweiten Abschnitts des Anwendungscodes, indem ein in der vierten Implementierung implementierter Konstruktor aufgerufen wird, um die zweite Komponente in dem Anwendungsprogramm zu erzeugen.

8. Verfahren nach Anspruch 7, bei dem die erste, zweite, dritte und vierte Implementierung alle eine gemeinsame Schnittstelle implementieren.

9. Verfahren zur Verwendung in einem System, das ein 4GL-Programmierwerkzeug (**112**) aufweist, welches es Anwendungsprogrammierern (**110**) erlaubt, durch Interaktion mit Benutzerschnittstellenobjekten und Steuerelementen, die von dem 4GL-Programmierwerkzeug bereitgestellt werden, Anwendungsprogramme zu erzeugen, die Komponenten aufweisen, welche einem Satz von Komponententypen angehören, für die Implementierungen von dem 4GL-Programmierwerkzeug bereitgestellt werden, wobei das Verfahren die Schritte aufweist, daß
das 4GL-Programmierwerkzeug (**112**) ein Steuerelement zum Erhalten von Daten bereitstellt, die eine Implementierung für eine Komponente eines Anwendungsprogramms, das unter Verwendung des 4GL-Programmierwerkzeugs entworfen wird, angeben, wobei die Komponente eine Instanz eines Komponententyps ist, der dem Satz von Komponententypen zugehört; und
das 4GL-Programmierwerkzeug (**112**) Anwendungscode (**114**) erzeugt, der, wenn er ausgeführt wird, einen Konstruktor zum Erzeugen der Komponente in dem Anwendungsprogramm aufruft,
dadurch gekennzeichnet, daß die Implementierung eine durch den Benutzer angegebene Implementierung, und zwar nicht eine der durch das 4GL-Programmierwerkzeug bereitgestellten Implementierungen, ist, und daß der Konstruktor in der durch den Benutzer angegebenen Implementierung für die von einem Abschnitt des Anwendungscodes angegebene Komponente implementiert ist.

10. Verfahren nach Anspruch 9, bei dem das Steuerelement eines einer Mehrzahl von Steuerelementen auf einer Eigenschaftspalette ist, um durch den Benutzer angegebene Werte für eine Mehrzahl von Eigen-

schaften der Komponente zu erhalten.

11. Verfahren nach Anspruch 9, bei dem der Schritt des Erzeugens des Anwendungscodes umfaßt, daß Anwendungscode erzeugt wird, der von einer Laufzeitmaschine (**116**) interpretiert wird, um das Anwendungsprogramm auszuführen.

12. Verfahren nach Anspruch 11, bei dem:
die von dem 4GL-Programmierwerkzeug (**112**) für den Satz von Komponententypen bereitgestellten Implementierungen alle eine gemeinsame Schnittstelle unterstützen;
die von dem Benutzer angegebene Implementierung die gemeinsame Schnittstelle unterstützt; und
während der Ausführung des Anwendungsprogramms die Komponente und alle Komponenten, welche unter Verwendung der von dem 4GL-Programmierwerkzeug bereitgestellten Implementierungen implementiert werden, mittels Aufrufen instantiiert werden, die über die gemeinsame Schnittstelle durchgeführt werden.

13. Computerlesbares Medium mit einer Befehlssequenz oder mehreren Befehlssequenzen zum Ausführen eines von einem 4GL-Programmierwerkzeug (**112**) erstellten Anwendungsprogramms, wobei die Ausführung der einen Befehlssequenz oder der mehreren Befehlssequenzen durch einen Prozessor oder mehrere Prozessoren den einen Prozessor oder die mehreren Prozessoren veranlaßt, die Schritte auszuführen:
Einlesen, von Anwendungscode (**114**), der von dem 4GL-Programmierwerkzeug (**112**) erstellt worden ist, eines Abschnitts des Anwendungscodes (**114**), der eine Komponente für das Anwendungsprogramm angibt, wobei die Komponente einen bestimmten Komponententyp hat, für den das 4GL-Programmierwerkzeug (**112**) eine erste Implementierung bereitstellt; und
Ausführen des Abschnitts des Anwendungscodes, indem ein Konstruktor aufgerufen wird, um die Komponente in dem Anwendungsprogramm zu erzeugen,
dadurch gekennzeichnet, daß der Konstruktor in einer zweiten Implementierung für die durch den Abschnitt des Anwendungscodes angegebenen Komponente implementiert ist, wobei sich die zweite Implementierung von der ersten Implementierung unterscheidet und von dem 4GL-Programmierwerkzeug nicht bereitgestellt wird.

14. Computerlesbares Medium nach Anspruch 13, bei dem:
der Schritt des Einlesens von einer Laufzeitmaschine (**116**) ausgeführt wird, die sich auf einem Server befindet;
der Schritt des Aufrufens des Konstruktors von einem Client in Reaktion auf den Eingang einer Nachricht bei dem Client ausgeführt wird, wobei die Nachricht von der Laufzeitmaschine in Reaktion auf die Ausführung des Abschnitts des Anwendungscodes (**114**) erzeugt wird.

15. Computerlesbares Medium nach Anspruch 14, bei dem:
sowohl die erste Implementierung als auch die zweite Implementierung eine gemeinsame Schnittstelle unterstützen; und
der Schritt des Aufrufens des Konstruktors über die gemeinsame Schnittstelle von einem Nachrichtenverarbeitungsmodul (**128**, **130**, **132**) für den bestimmten Komponententyp ausgeführt wird.

16. Computerlesbares Medium nach Anspruch 13, bei dem:
das 4GL-Programmierwerkzeug (**112**) den Abschnitt des Anwendungscodes (**114**) erzeugt, und zwar in Reaktion darauf, daß
ein Anwendungsprogrammierer (**110**) eine Eingabe durchführt, die angibt, daß die Anwendung eine Komponente des bestimmten Komponententyps aufweisen soll; und
der Anwendungsprogrammierer (**110**) eine Eingabe durchführt, die angibt, daß die Komponente unter Verwendung der zweiten Implementierung implementiert werden soll.

17. Computerlesbares Medium nach Anspruch 13, bei dem:
die Laufzeitmaschine (**116**) den Abschnitt des Anwendungscodes ausführt, indem sie eine Nachricht an einen Verteiler (**140**) sendet;
der Verteiler (**140**) auf die Nachricht antwortet, indem er ein Nachrichtenverarbeitungsmodul (**128**, **130**, **132**) für den bestimmten Komponententyp instantiiert; und
das Nachrichtenverarbeitungsmodul den Konstruktor aufruft.

18. Computerlesbares Medium nach Anspruch 17, bei dem:
sich die Laufzeitmaschine (**116**) auf einem Server befindet;
der Verteiler (**140**) sich auf einem über ein Netzwerk operativ mit dem Server gekoppelten Client befindet; und
die Laufzeitmaschine die Nachricht über das Netzwerk an den Verteiler sendet.

19. Computerlesbares Medium nach Anspruch 13, ferner mit Befehlen zum Ausführen der Schritte:
Einlesen, von dem von dem 4GL-Programmierwerkzeug (**112**) erzeugten Anwendungscode (**114**), eines zweiten Abschnitts des Anwendungscodes, der
a) eine zweite Komponente für das Anwendungsprogramm angibt, wobei die zweite Komponente einen zweiten bestimmten Komponententyp hat, für den das 4GL-Programmierwerkzeug eine dritte Implementierung bereitstellt, wobei der zweite Komponententyp sich von dem ersten bestimmten Komponententyp unterscheidet; und
b) eine vierte Implementierung für die zweite Komponente angibt, die sich von der dritten Implementierung unterscheidet;
Ausführen des zweiten Abschnitts des Anwendungscodes, indem ein in der vierten Implementierung implementierter Konstruktor aufgerufen wird, um die zweite Komponente in dem Anwendungsprogramm zu erzeugen.

20. Computerlesbares Medium nach Anspruch 19, bei dem die erste, zweite, dritte und vierte Implementierung alle eine gemeinsame Schnittstelle implementieren.

21. Computerlesbares Medium mit Befehlen zum Implementieren eines 4GL-Programmierwerkzeugs (**112**), welches es Anwendungsprogrammierern (**110**) erlaubt, durch Interaktion mit Benutzerschnittstellenobjekten und Steuerelementen, die von dem 4GL-Programmierwerkzeug bereitgestellt werden, Anwendungsprogramme zu erzeugen, die Komponenten aufweisen, welche einem Satz von Komponententypen angehören, für die Implementierungen von dem 4GL-Programmierwerkzeug bereitgestellt werden, wobei das computerlesbare Medium Befehle zum Ausführen der Schritte aufweist, daß:
das 4GL-Programmierwerkzeug (**112**) ein Steuerelement zum Erhalten von Daten bereitstellt, die eine Implementierung für eine Komponente eines Anwendungsprogramms, das unter Verwendung des 4GL-Programmierwerkzeugs entworfen wird, angeben, wobei die Komponente eine Instanz eines Komponententyps ist, der dem Satz von Komponententypen zugehört; und
das 4GL-Programmierwerkzeug (**112**) Anwendungscode (**114**) erzeugt, der, wenn er ausgeführt wird, einen Konstruktor zum Erzeugen der Komponente in dem Anwendungsprogramm aufruft,
dadurch gekennzeichnet, daß die Implementierung eine durch den Benutzer angegebene Implementierung, und zwar nicht eine der durch das 4GL-Programmierwerkzeug bereitgestellten Implementierungen, ist, und daß der Konstruktor in der durch den Benutzer angegebenen Implementierung für die von einem Abschnitt des Anwendungscodes angegebene Komponente implementiert ist.

22. Computerlesbares Medium nach Anspruch 21, bei dem das Steuerelement eines einer Mehrzahl von Steuerelementen auf einer Eigenschaftspalette ist, um durch den Benutzer angegebene Werte für eine Mehrzahl von Eigenschaften der Komponente zu erhalten.

23. Computerlesbares Medium nach Anspruch 21, bei dem der Schritt des Erzeugens des Anwendungscodes umfaßt, daß Anwendungscode erzeugt wird, der von einer Laufzeitmaschine (**116**) interpretiert wird, um das Anwendungsprogramm auszuführen.

24. Computerlesbares Medium nach Anspruch 23, bei dem:
die von dem 4GL-Programmierwerkzeug (**112**) für den Satz von Komponententypen bereitgestellten Implementierungen alle eine gemeinsame Schnittstelle unterstützen;
die von dem Benutzer angegebene Implementierung die gemeinsame Schnittstelle unterstützt; und
während der Ausführung des Anwendungsprogramms die Komponente und alle Komponenten, welche unter Verwendung der von dem 4GL-Programmierwerkzeug bereitgestellten Implementierungen implementiert werden, mittels Aufrufen instantiiert werden, die über die gemeinsame Schnittstelle durchgeführt werden.

Es folgen 2 Blatt Zeichnungen

Anhängende Zeichnungen

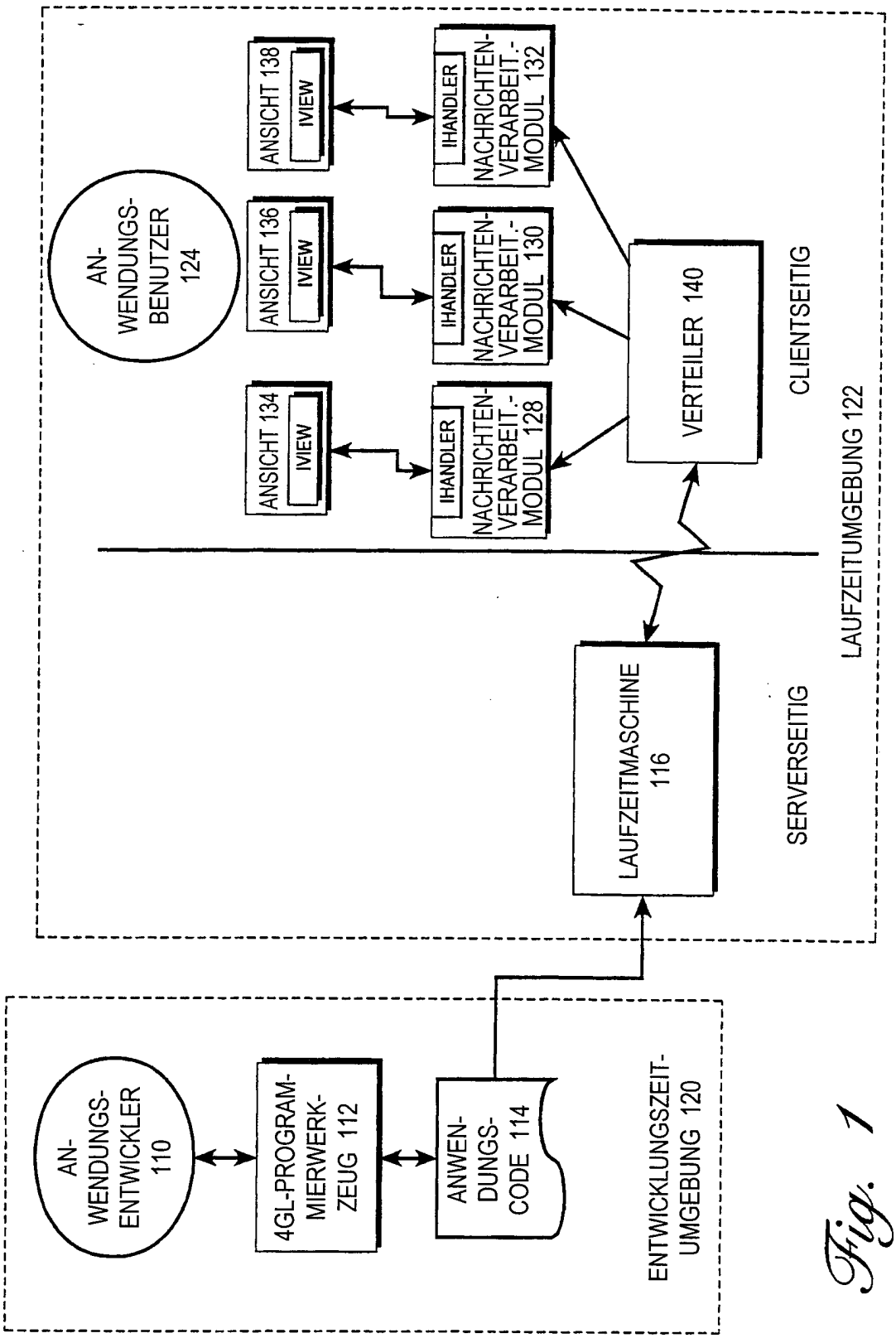

Fig. 1

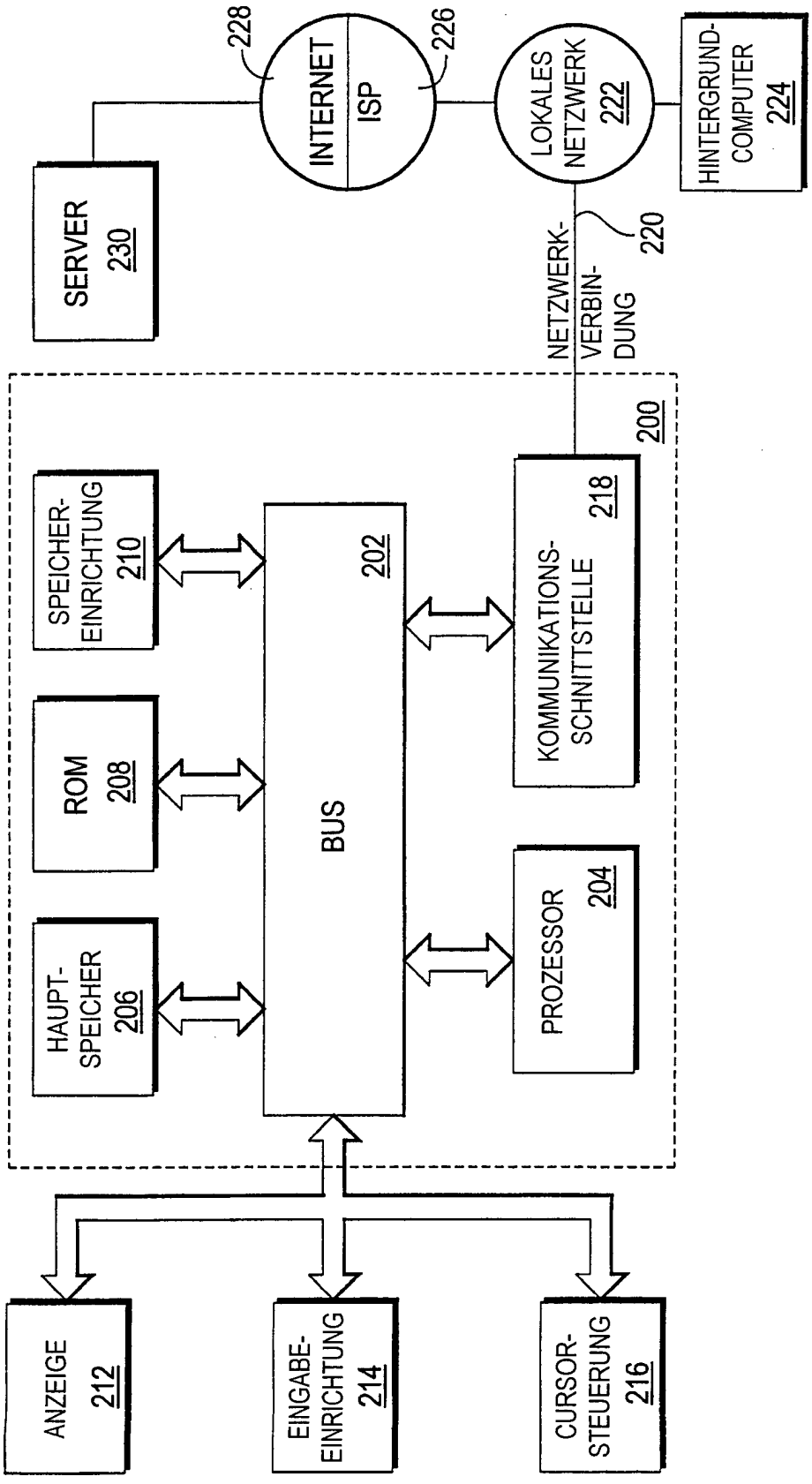
228
INTERNET
ISP
226
LOKALES NETZWERK 222
HINTERGRUND-COMPUTER 224
SERVER 230
NETZWERK-VERBIN-DUNG
220
200
SPEICHER-EINRICHTUNG 210
ROM 208
HAUPT-SPEICHER 206
BUS 202
KOMMUNIKATIONS-SCHNITTSTELLE 218
PROZESSOR 204
ANZEIGE 212
EINGABE-EINRICHTUNG 214
CURSOR-STEUERUNG 216

Fig. 2