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

(10) **DE 695 33 349 T2** 2005.08.04

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 720 329 B1**
(21) Deutsches Aktenzeichen: **695 33 349.6**
(96) Europäisches Aktenzeichen: **95 117 827.6**
(96) Europäischer Anmeldetag: **10.11.1995**
(97) Erstveröffentlichung durch das EPA: **03.07.1996**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **11.08.2004**
(47) Veröffentlichungstag im Patentblatt: **04.08.2005**

(51) Int Cl.[7]: **H04L 12/24**

(30) Unionspriorität:
348024 **01.12.1994** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB, SE

(73) Patentinhaber:
Hewlett-Packard Development Co., L.P., Houston, Tex., US

(74) Vertreter:
Schoppe, Zimmermann, Stöckeler & Zinkler, 82049 Pullach

(72) Erfinder:
Schettler, Robert D., Fort Collins, Colorado 80521, US; McCollom, William G., Fort Collins, Colorado 80525, US; Haimson, David M., Fort Collins, Colorado 80521, US

(54) Bezeichnung: **Persistenzspezifizierungssystem und Verfahren für Hochleistungssubkarten auf Anfrage**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung

Gebiet der Erfindung

[0001] Die vorliegende Erfindung bezieht sich allgemein auf Datenkommunikationsnetzwerke und insbesondere auf ein Persistenzspezifikationssystem und -verfahren zum Ermöglichen einer Erzeugung mit hoher Leistungsfähigkeit von Auf-Anforderung-Teilabbildungen in einer Verwaltungsstation für ein Datenkommunikationsnetzwerk.

Hintergrund der Erfindung

[0002] Ein Datenkommunikationsnetzwerk umfasst allgemein eine Gruppe von Geräten, z. B. Computer, Repeater, Brücken, Router, etc., die bei Netzwerkknoten gelegen sind, und eine Sammlung von Kommunikationskanälen zum Verbinden der verschiedenen Knoten. Eine Hardware und eine Software, die dem Netzwerk und insbesondere den Geräten zugeordnet sind, ermöglichen, dass die Geräte Daten elektronisch über die Kommunikationskanäle austauschen.

[0003] Die Größe von Netzwerken variiert. Ein lokales Netz (LAN = Local Area Network) ist ein Netzwerk von Geräten in naher Nähe, typischerweise weniger als eine Meile, und gewöhnlich durch ein einziges Kabel verbunden, z. B. ein Koaxialkabel. Ein weites Netz (WAN = Wide Area Network) ist ein Netzwerk von Geräten, die durch längere Abstände getrennt sind und oft z. B. durch Telefonleitungen oder Satellitenverbindungen verbunden sind. Tatsächlich überspannen einige WANs die USA sowie die Welt. Ferner sind viele dieser Netzwerke bereits verfügbar für eine Verwendung durch die Öffentlichkeit, häufig einschließlich Universitäten und kommerzieller Industrien.

[0004] Ein sehr beliebtes Industrienorm-Protokoll für eine Datenkommunikation entlang der Netzwerke ist das Internetprotokoll (IP). Dieses Protokoll wurde ursprünglich durch das Verteidigungsministerium der US-Regierung entwickelt und wurde durch die US-Regierung einer öffentlichen Verwendung zugänglich gemacht. Mit der Zeit wurden das Tramsmission-Control-Protocol (TCP) und das Unreliable-Datagram-Protocol (UDP) für eine Verwendung mit dem IP entwickelt. Das erstere Protokoll (TCP/IP) ist ein Protokoll, das eine Übertragung von Daten ohne Fehler garantiert, da dasselbe eine bestimmte Prüffunktionalität implementiert, und das letztere Protokoll (UDP/IP) ist ein Protokoll, das eine Übertragung von Daten nicht garantiert, aber viel weniger Mehraufwand erfordert, als die TCP/IP-Plattform. Um die verschiedenen Geräte, die in einem Netzwerk gelegen sind, zu verfolgen und zu verwalten, wurde ferner schließlich das Simple-Network-Management-Protocol (SNMP) für eine Verwendung bei einer UDP/IP-Plattform entwickelt. Die Verwendung der vorhergehenden Protokolle wurde in der Industrie extensiv und zahlreiche Verkäufer stellen nun viele Typen von Netzwerkgeräten her, die diese Protokolle einsetzen können.

[0005] Viele Verwaltungssoftwarepakete („Verwaltungsplattformen") sind gegenwärtig zum Implementieren von „Verwaltungsstationen" in einem Netzwerk und insbesondere in Verbindung mit dem SNMP erhältlich. Beispiele von im Handel erhältlichen SNMP-Verwaltungssoftwarepaketen umfassen OpenView von der Hewlett-Packard Company (hierin die Anmelderin), NetView/6000 von IBM Corp., Spectrum von Cabletron Systems, Inc., NetLabs/Manager von NetLabs, Inc. und SunNet Manager von SunConnect, Inc. Die Knoten in einem Netzwerk und die Verbindungen derselben, oft als die Netzwerk-„Topologie" bezeichnet, werden am besten in einem graphischen Format angezeigt und die meisten, wenn auch nicht alle, der verfügbaren Verwaltungssoftwarepakete stellen dieses Merkmal bereit. Mit diesen Paketen kann ein Netzwerk typischerweise von unterschiedlichen Betrachtungspunkten betrachtet werden, abhängig von dem Bereich der erwünschten Ansicht. Eine Ansicht des Netzwerks könnte z. B. eine sehr breite umfassende Ansicht aller Knoten des gesamten Netzwerks sein. Eine zweite Ansicht könnte eine Ansicht dieser Abschnitte eines Netzwerks innerhalb eines lokalen Bereichs sein, z. B. innerhalb eines bestimmten Standorts oder Gebäudes. Eine dritte Ansicht eines Netzwerks, oft ein Segment genannt, könnte eine Ansicht der Knoten sein, die an einem speziellen LAN-Kabel angeschlossen sind.

[0006] Hewlett-Packards sehr erfolgreiches OpenView war der Gegenstand mehrerer Patente, einschließlich z. B. dem US-Patent Nr. 5,185,860, erteilt an J. C. Wu am 09. Februar 1993, und dem US-Patent Nr. 5,276,789, erteilt an Besaw et al. am 04. Januar 1994. Das US-Patent Nr. 5,185,860 beschreibt ein automatisches Entdeckungssystem für eine Verwaltungsstation zum Bestimmen der Netzwerkgeräte und -verbindungen eines Netzwerks oder der Topologie. Das US-Patent Nr. 5,276,789 beschreibt ein Graphikanzeigesystem für eine Verwaltungsstation zum graphischen Anzeigen der Topologie eines Netzwerks und stellt verschiedene Ansichten (einschließlich Internet-, Segment-, und Knotenansichten) bereit, die durch einen Benutzer angefordert werden können.

[0007] Obwohl die gegenwärtig verfügbaren SNMP-Verwaltungsstationen zu einem Ausmaß verdienstvoll sind, befindet sich die Technik von SNMP-Verwaltungsstationen immer noch in den Anfängen und die Leistungsfähigkeit dieser Verwaltungsstationen kann noch verbessert und optimiert werden. Ein spezifischer Bereich, in dem man sich eine Optimierung vorstellt, betrifft die Integrationsan-

wendungen, die den Verwaltungsstationen zugeordnet sind und die zusätzliche Informationen über Netzwerkgeräte, z. B. Router, liefern können. Die zusätzlichen Informationen könnten z. B. Gerätekonfigurationsinformationen, einen Gerätestatus, Leistungsfähigkeitsparameter, zusätzliche Topologieinformationen, etc. umfassen. Bei Verwaltungsstationen des Stands der Technik jedoch werden diese zusätzlichen Informationen dem Benutzer oft nicht verfügbar gemacht. Ein Grund, dass diese zusätzlichen Informationen nicht verfügbar gemacht werden, ist, dass diese Systeme unter einer unangemessenen Schnittstelle zwischen den Integrationsanwendungen und dem Anzeigeabbildungsgenerator innerhalb dieser Systeme leiden.

[0008] Genauer gesagt weisen viele Verwaltungsstationen Auf-Anforderung-Teilabbildung-Fähigkeiten auf. Ein derartiges Beispiel ist Hewlett-Packards OpenView. Bei Auf-Anforderung-Teilabbildung-Systemen entspricht eine Teilabbildung jeder Ansicht des Netzwerks, die angezeigt werden soll. Die Systemabbildung ist die Sammlung aller Teilabbildungen. Bei diesen Auf-Anforderung-Teilabbildung-Systemen und insbesondere dem OpenView-System spezifiziert der Benutzer, welche Teilabbildungen der Benutzer verfügbar zu haben wünscht, und spezifiziert daher die Teilabbildungen, die innerhalb der Abbildung resident sind. Außerdem kann der Benutzer ferner eine Teilabbildung während eines Betriebs öffnen oder „explodieren lassen", obwohl dieselbe nicht als innerhalb der Abbildung resident spezifiziert ist. In diesem Fall wird die Teilabbildung unmittelbar aus Topologiedaten erzeugt, wenn der Benutzer die Verwaltungsstation auffordert, die Teilabbildung zu öffnen, daher der Name Auf-Anforderung. Der vorhergehende Entwurf ist jedoch problematisch. Erstens, falls eine Teilabbildung nicht in der Abbildung resident ist und der Benutzer die Teilabbildung öffnet, dann kann eine Integrationsanwendung auf Grund von Zeitbeschränkungen beim Öffnen der erwünschten Teilabbildung nicht angemessen bestimmen, welche zusätzlichen Informationen zu der Verwaltungsstation zu liefern sind, und der Benutzer wird deshalb nicht über diese zusätzlichen Informationen benachrichtigt. Mit anderen Worten benötigen die Integrationsanwendungen Zeit, um die Topologiedaten zu analysieren, und Zeit, um die zusätzlichen Informationen für eine Anzeige für den Benutzer zu erzeugen.

[0009] Ein anderes Problem bei gegenwärtigen Entwürfen ist, dass Integrationsanwendungen für nicht-residente Teilabbildungen nicht in der Lage sind, den Benutzer vor kritischen Informationen zu warnen, bis der Benutzer die Teilabbildung explodieren lässt. Es ist erwünscht, den Benutzer zu warnen, sobald eine Anomalie erfasst wird.

[0010] Bei Verwaltungsstationen des Stands der Technik bleibt, wenn eine Teilabbildung einmal erzeugt wurde, dieselbe ferner von da an in der Abbildung des Systems, selbst falls die Integrationsanwendungen die Teilabbildung nicht länger benötigen. Diese missliche Lage verwendet unerwünschterweise wertvollen Speicherplatz. Diese Situation verbraucht ferner unnötigerweise Prozessorzeit, wenn gewisse Ereignisse oder Veränderungen bei einer Netzwerktopologie unnötigerweise zum Aktualisieren einiger der unnötigen residenten Teilabbildungen verarbeitet werden. Somit besteht ein Bedarf in der Industrie nach einem System und einem Verfahren zum besseren schnittstellenmäßigen Verbinden von Integrationsanwendungen mit einem Auf-Anforderung-Teilabbildung-System zum Minimieren von Speichererfordernissen desselben und zum Optimieren einer Leistungsfähigkeit (einschließlich einer Geschwindigkeit) desselben.

[0011] Es ist die Aufgabe der vorliegenden Erfindung, ein System und ein Verfahren zum Verbessern der Leistungsfähigkeit eines Auf-Anforderung-Teilabbildung-Prozesses in einer Verwaltungsstation zu schaffen.

[0012] Diese Aufgabe wird durch ein System gemäß Anspruch 1 und ein Verfahren gemäß Anspruch 9 gelöst.

[0013] Ein weiterer Vorteil der vorliegenden Erfindung ist, ein verbessertes schnittstellenmäßiges Verbinden von Integrationsanwendungen mit Teilabbildungsinformationen zu liefern.

[0014] Ein weiterer Vorteil der vorliegenden Erfindung ist, ein System und ein Verfahren zum Minimieren von Speichererfordernissen für einen Auf-Anforderung-Teilabbildung-Prozess in einer Verwaltungsstation bereitzustellen.

[0015] Ein weiterer Vorteil der vorliegenden Erfindung ist, ein System und ein Verfahren zum Minimieren einer erforderlichen Verarbeitungszeit für einen Auf-Anforderung-Teilabbildung-Prozess in einer Verwaltungsstation bereitzustellen.

[0016] Kurz beschrieben ist die vorliegende Erfindung ein Persistenzspezifikationssystem und -verfahren für eine Verwaltungsstation oder ein anderes Gerät eines Netzwerks zum Verbessern einer Schnittstelle zwischen einer Integrationsanwendung und einer graphischen Benutzerschnittstelle, so dass mehr Informationen ein Netzwerk betreffend zu einem Benutzer geliefert werden, während Speichererfordernisse und eine Verarbeitungszeit minimiert sind. Das System weist einen Prozessor, der die verschiedenen Softwareelemente des Systems ausführt, einen Entdeckungsmechanismus zum Bestimmen der Netzwerktopologiedaten und einen Layout-Mechanismus zum Umwandeln der Netzwerktopologiedaten in Abbildungsdaten und zum Treiben ei-

ner Anzeige mit den Abbildungsdaten auf.

[0017] Genauer gesagt weist der Entdeckungsmechanismus einen Netzwerkmonitor, der konfiguriert ist, um die Topologie des Netzwerks zu überwachen und zu bestimmen, eine Topologiedatenbank zum Speichern der Topologiedaten und einen Topologieverwalter bzw. eine Topologieverwaltungseinrichtung zum Verwalten der Topologiedatenbank auf. Der Netzwerkmonitor ist ferner konfiguriert, um Topologieveränderungsereignisse zu erzeugen, wenn eine Veränderung bei einer Netzwerktopologie auftritt, und ferner derartige Ereignisse von anderen Geräten zu empfangen, die mit dem Netzwerk verbunden sind.

[0018] Der Layout-Mechanismus ist in Kommunikation mit dem Entdeckungsmechanismus und einer oder mehreren Integrationsanwendungen. Der Layout-Mechanismus weist einen Topologie-zu-Abbildung-Übersetzer, eine graphische Benutzerschnittstelle (GUI = graphical user interface) und eine Abbildungsdatenbank auf. Der Übersetzer ist konfiguriert, um die Topologiedaten in die Abbildungsdaten umzuwandeln, die Ereignisse zu empfangen und die Abbildungsdaten basierend auf den Ereignissen zu verändern. Die GUI ist konfiguriert, um die Abbildungsdaten von dem Übersetzer zu empfangen, die Abbildungsdaten in der Abbildungsdatenbank zu verwalten und die Anzeige basierend auf den Abbildungsdaten zu treiben. Die GUI liefert ferner Informationen zu und empfängt Informationen von den Integrationsanwendungen, die die Abbildungsdaten betreffen.

[0019] Bei dem bevorzugten Ausführungsbeispiel ist der Übersetzer konfiguriert, um einen Satz von hierarchisch angeordneten Teilabbildungen entsprechend verschiedener Ansichten des Netzwerks zu erzeugen. Eine Abbildung ist hierin als die Sammlung aller Teilabbildungen definiert. Die hierarchisch angeordneten Teilabbildungen umfassen eine Internet-Teilabbildung, die zumindest ein Netzwerkobjekt aufweist, zumindest eine Netzwerk-Teilabbildung, die dem zumindest einem Netzwerkobjekt zugeordnet ist und zumindest ein Segmentobjekt aufweist, zumindest eine Segment-Teilabbildung, die dem zumindest einem Segmentobjekt zugeordnet ist und zumindest ein Knotenobjekt aufweist, und zumindest eine Knoten-Teilabbildung, die dem zumindest einem Knotenobjekt zugeordnet ist und zumindest ein Schnittstellenobjekt aufweist.

[0020] Gemäß einem erheblichen Merkmal der vorliegenden Erfindung weist der Übersetzer einen Persistenzspezifikationsmechanismus auf. Der Persistenzspezifikationsmechanismus und eine zugeordnete Methodologie spezifizieren basierend auf Eingangssignalen, die dem Objekt entsprechen, wann ein Objekt, das angezeigt werden soll, „persistent" ist und wann ein Objekt, das angezeigt werden soll, „transient" ist. Die Eingangssignale werden durch den Benutzer und durch die Integrationsanwendungen erzeugt, die mit der Entdeckungs-/Layout-Software verbunden sind. Der Übersetzer definiert Teilabbildungen als persistent, wenn die Teilabbildung ein persistentes Objekt aufweist, und als transient, wenn die Teilabbildung ohne persistente Objekte ist.

[0021] Persistente Teilabbildungen werden erzeugt und kontinuierlich in dem Übersetzer beibehalten, so dass Integrationsanwendungen wirksam einen kontinuierlichen Zugriff aufweisen und die persistenten Objektinformationen verwenden können, um zusätzliche Informationen für den Benutzer zu erzeugen und zu liefern. Transiente Teilabbildungen werden lediglich für eine temporäre Zeitperiode erzeugt, wenn ein Benutzer das System auffordert, und der Übersetzer ist konfiguriert, um die transienten Teilabbildungen zu löschen (nicht zu speichern), nachdem der Benutzer mit einem Betrachten der transienten Teilabbildungen fertig ist. Folglich verbessert das Persistenzspezifikationssystem die Schnittstelle zwischen den Integrationsanwendungen und der graphischen Benutzerschnittstelle, so dass mehr Informationen ein Netzwerk betreffend zu dem Benutzer geliefert werden. Das System minimiert ferner Speichererfordernisse und eine Verarbeitungszeit durch ein Entfernen unnötiger Teilabbildungen, die basierend auf eingehenden Ereignissen der Gegenstand einer Veränderung sein könnten.

[0022] Zusätzlich zu einem Erreichen aller zuvor erwähnter Aufgaben, weist die vorliegende Erfindung zahlreiche weitere Vorteile auf, von denen einige hierin im Folgenden geschildert sind.

[0023] Ein Vorteil der vorliegenden Erfindung ist, dass dieselbe einfach in einem Entwurf ist und ohne weiteres auf einer kommerziellen Massenskala zu implementieren ist.

[0024] Ein weiterer Vorteil der vorliegenden Erfindung ist, dass dieselbe effizient sowie in einem Betrieb zuverlässig ist.

[0025] Ein weiterer Vorteil der vorliegenden Erfindung ist, dass die Grundlagen des Persistenzspezifikatiossystems und -verfahrens nicht nur auf Verwaltungsstationen sondern auch auf andere Geräte angewendet werden können, einschließlich Geräten, die das SNMP nicht praktizieren.

[0026] Andere Aufgaben, Merkmale und Vorteile der vorliegenden Erfindung werden einem Fachmann auf dem Gebiet auf eine Untersuchung der folgenden Zeichnungen und der detaillierten Beschreibung hin ersichtlich. Alle derartigen zusätzlichen Aufgaben, Merkmale und Vorteile sollen hierin innerhalb des Schutzbereichs der vorliegenden Erfindung eingeschlossen sein.

Kurze Beschreibung der Zeichnungen

[0027] Die vorliegende Erfindung wird mit Bezug auf die folgenden Zeichnungen in dem Kontext des Texts ersichtlicher.

[0028] Fig. 1 ist ein Blockdiagramm einer Verwaltungsstation, die eine Entdeckungs-/Layout-Software aufweist, die das Persistenzspezifikationssystem und -verfahren der vorliegenden Erfindung einsetzt;

[0029] Fig. 2 ist ein schematisches Diagramm, das eine Anzeigeabbildung darstellt, die eine Sammlung von Teilabbildungen aufweist, von denen jegliche auf der Anzeige der Verwaltungsstation durch die Entdeckungs-/Layout-Software von Fig. 1 angezeigt werden können;

[0030] Fig. 3 ist ein Blockdiagramm der Entdeckungs-/Layout-Software von Fig. 1 und einer Integrationsanwendung, die mit derselben schnittstellenmäßig verbunden ist, gemäß der vorliegenden Erfindung;

[0031] Fig. 4 ist ein Flussdiagramm, das die Architektur des Topologie-zu-Abbildung-Übersetzers von Fig. 3 darstellt;

[0032] Fig. 5 ist ein Flussdiagramm, das die Architektur eines Stapel-Lesen-Blocks von Fig. 4 darstellt;

[0033] Fig. 6 ist ein Flussdiagramm, das die Architektur eines Objekte-Wiedererlangen-Blocks von Fig. 4 darstellt;

[0034] Fig. 7 ist ein Flussdiagramm, das die Architektur des Abbildungsveränderungen-Berechnen-Blocks von Fig. 4 darstellt;

[0035] Fig. 8 ist ein Flussdiagramm, das die Architektur eines Teilabbildungsveränderungen-Berechnen-Blocks von Fig. 7 darstellt;

[0036] Fig. 9 ist ein Flussdiagramm, das die Architektur eines Netzwerkveränderungsblocks von Fig. 8 darstellt;

[0037] Fig. 10 ist ein Flussdiagramm, das die Architektur eines Segmentveränderungsblocks von Fig. 8 darstellt;

[0038] Fig. 11 ist ein Flussdiagramm, das die Architektur eines Knotenveränderungsblocks von Fig. 8 darstellt;

[0039] Fig. 12 ist ein Flussdiagramm, das die Architektur eines Schnittstellenveränderungsblocks von Fig. 8 darstellt;

[0040] Fig. 13 ist ein Flussdiagramm, das die Architektur eines Abbildung-Aktualisieren-Blocks von Fig. 4 darstellt;

[0041] Fig. 14 ist ein Flussdiagramm, das die Architektur eines Auf-Anforderung-Teilabbildung-Blocks innerhalb der graphischen Benutzerschnittstelle (GUI) von Fig. 2 darstellt;

[0042] Fig. 15 ist ein Zustandsdiagramm, das einer Teilabbildung von Fig. 2 entspricht, gemäß dem neuartigen Persistenzspezifikationssystem und -verfahren von Fig. 1;

[0043] Fig. 16 ist ein schematisches Diagramm eines Persistenzvektors, der einer Teilabbildung entspricht, zum Implementieren des Persistenzspezifikationssystems und -verfahrens von Fig. 1;

[0044] Fig. 17 ist ein Flussdiagramm, das die Architektur eines Teilabbildung-Hinzufügen-Entscheidungsblocks von Fig. 7 zum Implementieren des Persistenzspezifikationssystems und -verfahrens von Fig. 1 darstellt; und

[0045] Fig. 18 ist ein Flussdiagramm, das die Architektur eines Persistenzspezifikation-Entscheidungsblocks von sowohl Fig. 7 als auch Fig. 17 zum Implementieren des Persistenzspezifikationssystems und -verfahrens von Fig. 1 darstellt.

Detaillierte Beschreibung des bevorzugten Ausführungsbeispiels

[0046] Die folgende Beschreibung ist über den gegenwärtig betrachteten besten Modus zum Ausführen der vorliegenden Erfindung. Diese Beschreibung soll nicht in einem begrenzenden Sinn aufgefasst werden, sondern wird lediglich zum Zweck eines Beschreibens der allgemeinen Grundlagen der Erfindung vorgenommen. Der Schutzbereich der Erfindung sollte durch ein Referenzieren der beigefügten Ansprüche bestimmt sein.

[0047] Fig. 1 zeigt ein Blockdiagramm einer objektorientierten Verwaltungsstation **100**, die mit einem Universalcomputersystem implementiert ist, das eine neuartige Entdeckungs-/Layout-Software **101** enthält, die einen Persistenzspezifikationsmechanismus **103** und eine zugeordnete Methodologie der vorliegenden Erfindung einsetzt. Mit Bezug auf Fig. 1 enthält die Verwaltungsstation **100** einen herkömmlichen Prozessor **102**. Der Prozessor **102** kommuniziert mit anderen Elementen innerhalb der Verwaltungsstation **100** über einen Systembus **104**. Ein Eingabegerät **106**, z. B. eine Tastatur oder eine Maus, wird verwendet, um Daten von einem Benutzer des Systems **100** einzugeben, und eine Anzeige **108** wird verwendet, um Daten zu dem Benutzer auszugeben. Eine Netzwerkschnittstelle **112** wird verwendet, um die Verwaltungsstation **100** schnittstellenmäßig mit

einem Netzwerk **118** zu verbinden, um zu ermöglichen, dass die Verwaltungsstation **100** als ein Knoten in einem Netzwerk **118** handelt. Eine Platte **114** kann verwendet werden, um die Software der Entdeckungs-/Layout-Software **101** der vorliegenden Erfindung zu speichern, sowie die Datenbanken (Topologie- und Abbildungs-) zu speichern, die durch die Entdeckungs-/Layout-Software **101** erzeugt werden. Ein Drucker **116** kann verwendet werden, um eine Druckkopieausgabe der Knoten des Netzwerks **118** bereitzustellen, die durch die Entdeckungs-/Layout-Software **101** entdeckt werden. Ein Hauptspeicher **110** innerhalb der Verwaltungsstation **100** enthält die Entdeckungs-/Layout-Software **101**. Die Entdeckungs-/Layout-Software **101** kommuniziert mit einem herkömmlichen Betriebssystem **122** und einer herkömmlichen Netzwerk-Software **124**, um die Knoten in dem Netzwerk **118** zu entdecken. Die Netzwerk-Software **124** dient als die Intelligenz, einschließlich einer Validierung, für die Datenkommunikationsprotokolle. Wie es in Fig. 1 gezeigt ist, implementiert bei dem bevorzugten Ausführungsbeispiel die Netzwerk-Software das IP, das TCP und UDP über das IP, and das SNMP über das UDP. Alle der vorhergehenden Protokolle sind auf dem Gebiet gut bekannt.

[0048] Die Entdeckungs-/Layout-Software **101** implementiert eine objektorientierte Funktionalität. In dem Kontext von SNMP-Verwaltern bedeutet objektorientiert, dass die meisten der Verwaltungssystemhandlungen und -prozesse, die der Benutzer aufrufen kann, zu einer Klasse von Geräten anstelle einzeln verwalteter Netzwerkknoten hin orientiert sind.

[0049] Allgemein ist die Entdeckungs-/Layout-Software **101** von Fig. 1 konfiguriert, um die Netzwerktopologie zu entdecken, d. h. alle Netzwerkknoten und Netzwerkverbindungen, die in dem Netzwerk **118** existieren, und um eine Abbildung aufzubauen, die verschiedene Teilabbildungen aufweist, von denen jegliche zum Anzeigen der Netzwerktopologie auf der Anzeige **108** verwendet werden können. Fig. 2 zeigt eine **Abb.** 200, die durch die Entdeckungs-/Layout-Software **101** aus Topologiedaten erzeugt wird, die von dem Netzwerk **118** entdeckt werden. Die Entdeckungs-/Layout-Software **101** kann jegliche der verschiedenen Teilabbildungen zu der Anzeige **108** (Fig. 1) zum Betrachten durch den Benutzer treiben.

[0050] Die Teilabbildungen in der **Abb.** 200 von Fig. 2 sind in einer Hierarchie angeordnet. Eine Wurzel-Teilabbildung bzw. Root-Teilabbildung **202** ist auf einer Wurzel-Ebene bzw. Root-Ebene definiert. Die Wurzel-Teilabbildung **202** stellt die Teilabbildung der höchsten logischen Ebene in der Hierarchie dar und zeigt Objekte **203**, die als Ankerpunkte für unterschiedliche Teilabbildungshierarchien handeln. Jede Hierarchie ist ein getrennter Verwaltungsbereich. Dies könnte z. B. ein Netzwerk, eine logische Gruppierung von Knoten oder ein gewisser anderer Bereich sein. Eine Internet-Teilabbildung **204** ist auf einer Internet-Ebene definiert und wird durch ein „Explodieren Lassen" eines Objekts **203** innerhalb der Wurzel-Teilabbildung **202** erzeugt. In dem Kontext dieses Dokuments bedeutet „Explodieren Lassen", dass der Benutzer die Verwaltungsstation **100** mit dem Eingabegerät **106** auffordert, zu unterbrechen und mehr Daten zu liefern, die zu dem fraglichen Objekt **203** gehören. Ferner stellt die Internet-Teilabbildung **204** Objekte **203** in der Form von Netzwerken und Routern dar. Irgendeine einer Anzahl von Netzwerk-Teilabbildungen **206** kann aus der Internet-Teilabbildung **204** explodieren gelassen werden. Jede Netzwerk-Teilabbildung **206** zeigt Objekte **203** in der Form von Segmenten und Verbindern. Irgendeine einer Anzahl von Segment-Teilabbildungen **208** kann aus einem Objekt **203** innerhalb einer Netzwerk-Teilabbildung **206** explodieren gelassen werden. Jede Segment-Teilabbildung **208** zeigt Objekte in der Form von Netzwerkknoten. Schließlich kann irgendeine einer Anzahl von Knoten-Teilabbildungen **210** aus einem Objekt innerhalb einer Segment-Teilabbildung **208** explodieren gelassen werden. Jede Knoten-Teilabbildung **210** zeigt Objekte **203** in der Form von Schnittstellen innerhalb dieses Knotens.

[0051] Bei dem bevorzugten Ausführungsbeispiel implementiert, obwohl es nicht notwendig ist, um die vorliegende Erfindung zu praktizieren, die Entdeckungs-/Layout-Software **101** Auf-Anforderung-Teilabbildungen, um einen Speicher und eine Verarbeitungszeit zu bewahren. Das Konzept von Auf-Anforderung-Teilabbildungen ist, lediglich diese Teilabbildungen in der **Abb.** 200 von Fig. 2 zu platzieren, die der Benutzer sehen will. Das Nettoergebnis ist, dass lediglich ein Abschnitt der Teilabbildungs-Hierarchie zu einer gegebenen Zeit in der **Abb.** 200 ist. In Fig. 2 sind Teilabbildungen (nicht-resident), die nicht vorhanden sind, aber auf ein Auffordern durch den Benutzer hin erzeugt würden, durch eine Schraffur angegeben. Der residente Teilabbildungsteilsatz der Hierarchie ändert sich über die Zeit, wenn der Benutzer die Teilabbildungshierarchie durchläuft und bewirkt, dass nicht-residente Teilabbildungen erzeugt werden.

[0052] Ein Blockdiagramm auf hoher Ebene der Entdeckungs-/Layout-Software **101** (Fig. 1) ist in Fig. 3 dargelegt. Mit der Ausnahme des Persistenzspezifikationsmechanismus **103** ist die Architektur der Entdeckungs-/Layout-Software **101** in Fig. 3 im Wesentlichen die gleiche wie die oder ähnlich der Architektur von Hewlett-Packards gut bekanntem und im Handel erhältlichen Verwaltungssoftwarepaket, das OpenView genannt wird. Wie es in Fig. 3 gezeigt ist, weist auf der allgemeinen Architekturebene die Entdeckungs-/Layout-Software **101** einen Entdeckungsmechanismus **302** zum Entdecken von Knoten und Verbindungen des Netzwerks **118** und einen

Layoutmechanismus **304** zum Empfangen von Topologiedaten von dem Entdeckungsmechanismus **302** und zum Erzeugen der **Abb.** 200 (Fig. 2) zum Treiben der Anzeige **108** auf. Außerdem können eine oder mehrere Integrationsanwendungen **332** Anzeige- und Abbildungsinformationen mit dem Layoutmechanismus **304** kommunizieren.

[0053] Der Entdeckungsmechanismus **302** weist einen Netzwerkmonitor **306**, der mit dem Netzwerk **118** verbunden ist, wie es durch Verbindungen **308a**, **308b** angegeben ist, einen Topologieverwalter bzw. eine Topologieverwaltungseinrichtung **310**, die mit dem Netzwerkmonitor **306** verbunden ist, wie es durch Pfeile **312a**, **312b** angegeben ist, und eine Topologiedatenbank in Kommunikation mit dem Topologieverwalter **310** auf, wie es durch einen Pfeil **316** angegeben ist.

[0054] Der Netzwerkmonitor **306** sendet und empfängt Datenpakete zu und von dem Netzwerk **118**. Der Netzwerkmonitor **306** entdeckt und überwacht eine Netzwerktopologie, wie es durch die Pfeile **308a**, **308b** angegeben ist. Wenn sich eine Netzwerktopologie in dem Netzwerk verändert, erzeugt der Netzwerkmonitor **306** Ereignisse oder Programmunterbrechungen (traps) (SNMP-Fachsprache), die einen Objektidentifizierer und Objektveränderungsinformationen umfassen. Der Netzwerkmonitor **306** kann ferner Ereignisse von anderen Geräten in dem Netzwerk **118** empfangen, wie beispielsweise einem Router. Der Netzwerkmonitor **306** steht mit dem Netzwerk **118** durch die Netzwerksoftware **126** in Wechselwirkung (Fig. 1), die im Wesentlichen Protokollstapel aufweist, die bei dem bevorzugten Ausführungsbeispiel IP, TCP, UDP und SNMP entsprechen, und die diese Protokolle allgemein implementiert und Validierungsfunktionen durchführt. Ferner bestückt der Netzwerkmonitor **306** die Topologiedatenbank **314** durch den Topologieverwalter **310** und benachrichtigt den Topologieverwalter **310** über Ereignisse (Topologieveränderungen). Schließlich ist anzumerken, dass das US-Patent Nr. 5,185,860 an Wu ein Knotenentdeckungssystem beschreibt, das eingesetzt werden könnte, um den Netzwerkmonitor **306** hierin zu implementieren.

[0055] Der Topologieverwalter **310** verwaltet die Topologiedatenbank **314**, wie es durch einen Zweirichtungspfeil **316** angegeben ist. Der Topologieverwalter **310** fordert den Netzwerkmonitor **306** auf, Topologiedaten zu aktualisieren, die auf spezielle Ereignisse bezogen sind, wie es durch den Pfeil **312a** angegeben ist, und empfängt Topologieaktualisierungen, wie es durch den Pfeil **312b** angegeben ist.

[0056] Die Topologiedatenbank **314** speichert Topologiedaten basierend auf Objekten, die verwendet werden, um das Netzwerk aus logischen Gründen zu partitionieren. Objekte umfassen zum Beispiel, aber nicht begrenzt darauf, ein Netzwerk, ein Segment, einen Computer, einen Router, einen Repeater, eine Brücke, etc. Außerdem umfassen die Topologiedaten, die mit Bezug auf die Objekte gespeichert sind, zum Beispiel, aber nicht begrenzt darauf, eine Schnittstellen- oder Geräteadresse, einen Schnittstellen- oder Gerätetyp, einen Schnittstellen- oder Gerätehersteller und ob eine Schnittstelle oder ein Gerät das SNMP unterstützt.

[0057] Der Layoutmechanismus **304** weist einen Topologie-zu-Abbildung-Übersetzer **318** in Kommunikation mit dem Topologieverwalter **310**, wie es durch Pfeile **320a**, **320b** angegeben ist, eine graphische Benutzerschnittstelle (GUI = graphical user interface) **322** in Kommunikation mit dem Topologie-zu-Abbildung-Übersetzer **318**, wie es durch Pfeile **324a**, **324b** angegeben ist, und eine Abbildungsdatenbank **326** in Kommunikation mit der GUI **322** auf, wie es durch einen Zweirichtungspfeil **328** angegeben ist. Die Integrationsanwendung **332** kommuniziert Informationen mit der GUI **322**, wie es durch Pfeile **333a**, **333b** angegeben ist.

[0058] Es ist anzumerken, dass der Netzwerkmonitor **306**, der Topologieverwalter **310**, der Übersetzer **318** und die GUI **322** sich bei einem Verwenden der Kombination des Betriebssystems **122** (Fig. 1) und des Prozessors **102** (Fig. 1) abwechseln, um die jeweiligen Funktionen derselben zu erzielen. Ein „Kontextschalter", wie derselbe hierin verwendet wird, bezieht sich auf eine Veränderung bei einer Steuerung des Systems **122** und/oder des Prozessors **102** durch die vorhergehenden Softwareelemente.

[0059] Der Übersetzer **318** wandelt Topologiedaten aus der Topologiedatenbank **314** in Abbildungsdaten um und baut die verschiedenen Teilabbildungen **202** – **210** in der **Abb.** 200 von Fig. 2 auf. Der Übersetzer **318** kann eine Anforderung zu dem Topologieverwalter **310** weiterleiten, wie es durch einen Pfeil **320a** angegeben ist, um Topologiedaten hinsichtlich spezieller Objekte zu erhalten. Zusätzlich zu einem Weiterleiten von Topologiedaten zu dem Übersetzer **318** auf eine Anforderung hin benachrichtigt der Topologieverwalter **310** den Übersetzer **318**, wie es durch den Pfeil **320b** angegeben ist, wenn sich Topologiedaten basierend auf einem Ereignis verändert haben, so dass der Übersetzer **318** jegliche geeignete Veränderungen bei den Teilabbildungen vornehmen kann.

[0060] Die GUI **322** verwaltet die Abbildungsdatenbank **326**, wie es durch den Zweirichtungspfeil **328** angegeben ist, und verwaltet die Anzeige **108** und das Eingabegerät **106**, wie es durch die Pfeile **330a**, **330b** angegeben ist. Die GUI **322** empfängt Abbildungsaktualisierungen von dem Übersetzer **318**, wie es durch einen Pfeil **324b** angegeben ist, und legt dem Übersetzer **318** benutzerausgelöste Ereignisse vor, wie es durch einen Pfeil **324a** angegeben ist. Ein

benutzerausgelöstes Ereignis umfasst eine Aufforderung **330a** von einem Benutzer, ein Objekt explodieren zu lassen, wie es mit Bezug auf Fig. 2 beschrieben ist. Schließlich ist anzumerken, dass das US-Patent Nr. 5,276,789 an Besaw et al. eine graphische Benutzerschnittstelle beschreibt, die eingesetzt werden könnte, um die GUI **322** hierin zu implementieren.

[0061] Fig. 4 zeigt ein Flussdiagramm **400**, das die Architektur und Funktionalität des bevorzugten Ausführungsbeispiels des Topologie-zu-Abbildung-Übersetzers **318** (Fig. 3) angibt. Der Übersetzer setzt den Persistenzspezifikationsmechanismus **103** und eine zugeordnete Methodologie gemäß der vorliegenden Erfindung ein, die Kontextschalter (Veränderungen bei der Steuerung des Betriebssystems **122** und/oder des Prozessors **102**) minimieren und die Geschwindigkeit und die Leistungsfähigkeit der Entdeckungs-/Layout-Software **101** erheblich verbessern.

[0062] Mit Bezug auf Fig. 4 werden anfänglich Ereignisse in einer Warteschlange (nicht gezeigt) oder einem Akkumulator, der dem Topologieverwalter **310** zugeordnet ist, in Warteschlange gestellt und akkumuliert und warten auf eine Wiedererlangung durch den Übersetzer **318**. Der Übersetzer **318** liest während jedes Zugriffs einen Stapel von Ereignissen von dem Topologieverwalter **310**. Jedes Ereignis enthält einen Objektidentifizierer und eine Objektveränderung. Außerdem ist der Stapel um irgendeine Anzahl von Ereignissen größer als Null. Bei dem getesteten Ausführungsbeispiel war der Stapel auf eine Anzahl von nicht mehr als **500** Ereignissen begrenzt, aber andere Einstellungen, entweder kleiner als oder größer (vielleicht erheblich) als diese Anzahl könnten verwendet werden, abhängig von der Konfiguration des Systems.

[0063] Als nächstes, wie es bei einem Block **404** angegeben ist, ruft der Übersetzer **318** den Topologieverwalter **310** für eine Liste von Topologiedaten hinsichtlich aller Objekte, die in den Ereignissen identifiziert wurden. Nach einem Empfangen der Topologiedaten, tritt der Block **404** zu einem Block **406** über.

[0064] Bei dem Block **406** berechnet der Übersetzer **318** die Veränderungen, die an den Abbildungsdaten vorzunehmen sind, insbesondere der **Abb.** 200 (Fig. 2), basierend auf den Topologiedatenveränderungen, die in den Ereignissen angegeben sind. Der Block **406** tritt zu einem Block **408** über.

[0065] Bei dem Block **408** aktualisiert der Übersetzer **318** die **Abb.** 200 (Fig. 2) durch ein Rufen der GUI **322** und Benachrichtigen der GUI **322** über alle Teilabbildungsveränderungen (SYMCHANGELIST und NEWSYMLIST, die hierin im Folgenden beschrieben sind), die zu allen Objektveränderungen gehören. Diese Transaktion ist vorzugsweise, obwohl nicht notwendigerweise, eine Stapelübertragung. Während dieser Stapelübertragungstransaktion identifiziert der Übersetzer **318** jede Teilabbildung, die verändert werden soll, jedes Objekt, das innerhalb einer Teilabbildung verändert werden soll, und die spezielle Veränderung, die an dem Objekt bewirkt werden soll. Eine Objektveränderung kann zum Beispiel, aber nicht begrenzt darauf, eine Farb-, Positions- oder Verbindungsveränderung umfassen. Der Block **408** tritt zu einem Block **410** über.

[0066] Bei dem Block **410** bestimmt der Übersetzer **318**, ob es einen weiteren Stapel von Ereignissen gibt, der von dem Topologieverwalter **310** gelesen werden soll. Falls dem so ist, dann tritt der Block **410** zu dem Block **402** über und der vorhergehend beschriebene Prozess wird wiederholt. Falls nicht, dann wartet die Software bei dem Block **410** auf einen anderen Stapel von Ereignissen.

[0067] Wegen des bevorzugten Ausführungsbeispiels des Übersetzers **318**, das in Fig. 4 dargelegt ist, werden Topologiedaten, die zu verschiedenen Objekten gehören, in einem Stapel von dem Topologieverwalter **310** wiedererlangt und werden ferner Abbildungsdaten, die zu verschiedenen Teilabbildungen gehören, in einem Stapel von dem Übersetzer **318** zu der GUI **322** übertragen. Die vorhergehende Implementierung minimiert Kontextschalter, d. h. minimiert die Anzahl von Malen, die eine Steuerung des Prozessors **102** (Fig. 1) und/oder des Betriebssystems **122** (Fig. 1) von einem Softwaremodul zu einem anderen geleitet wird.

[0068] Fig. 5 zeigt ein Flussdiagramm der Architektur und Funktionalität zum Implementieren eines bevorzugten Ausführungsbeispiels des Stapel-Lesen-Blocks **402** (Fig. 4). Dieses Flussdiagramm stellt dar, wie der Übersetzer **318** einen Stapel von Ereignissen von dem Topologieverwalter **310** liest. Wie es bei einem Block **502** angegeben ist, werden anfänglich Ereignisse von dem Topologieverwalter **310**, die Veränderungen bei Topologiedaten angeben, akkumuliert (in Warteschlange gestellt). Ein Zähler bei einem Block **504** wird in Verbindung mit einer Schleife verwendet, um jedes Ereignis von dem Topologieverwalter **310** zu dem Übersetzer **318** zu führen bzw. zu leiten. Bei einem Block **506** wird ein Ereignis durch den Übersetzer **318** von dem Verwalter **310** gelesen. Der Block **506** tritt zu einem Block **508** über, der das Ereignis decodiert. Das Ereignis wird decodiert, um den Ereignistyp und zugeordnete Daten zu identifizieren. Es gibt zahlreiche Typen von Ereignissen und unterschiedliche Typen von Ereignissen weisen unterschiedliche Typen von zugeordneten Daten auf. Genauer gesagt kann ein Ereignis zum Beispiel, aber nicht darauf begrenzt, einen neuen Knoten oder eine Knotenstatusveränderung (z. B. verbunden/zugreifbar oder verbunden/unzugreifbar) betreffen. Ein Ereignis weist einen Ereignisidentifizierer, gewöhnlich

bei dem Kopfblock, zum Identifizieren des Ereignistyps auf. In dem Fall eines neuen Knotens enthält das Ereignis außerdem einen Objektidentifizierer und eine Adresse. In dem Fall einer Knotenstatusveränderung enthält das Ereignis einen Objektidentifizierer, den alten Status und den neuen Status.

[0069] Der Block **508** tritt zu einem Block **510** über. Bei dem Block **510** werden die decodierten Ereignisdaten (d. h. eine Aufzeichnung) zu einer TLIST hinzufügt. Bei einem Block **512** wird der Zähler inkrementiert, so dass ein anderes Ereignis bedient wird. Der Block **512** tritt zu einem Block **514** über, der bestimmt, ob es noch Ereignisse gibt, die bedient werden sollen. Falls dem so ist, dann tritt der Block **514** zurück zu dem Block **506** über und der zuvor erwähnte Prozess wird wiederholt. Falls nicht, dann kehrt der Block **514** zu dem Block **404** (Fig. 4) zurück.

[0070] Fig. 6 zeigt ein Flussdiagramm der Architektur und Funktionalität eines bevorzugten Ausführungsbeispiels zum Implementieren des Objektinformationen-Wiedererlangen-Blocks **404** (Fig. 4). Wie es in Fig. 6 angegeben ist, werden in diesem Flussdiagramm Objektinformationen (OBJINFO) aus den decodierten Ereignisdaten decodiert, die in der TLIST enthalten sind. Bei einem Block **602** wird die TLIST gelesen. Der Block **602** tritt zu einem Block **604** über, der einen Ereigniszähler einleitet. Der Zähler bewirkt in Verbindung mit einer Schleife, dass alle der Ereignisse innerhalb der TLIST bedient werden. In der Schleife wird bei einem Block **606** eine einzige Aufzeichnung gelesen. Aus der Aufzeichnung werden (a) ein Objektidentifizierer und (b) eine Objektveränderung bestimmt. Die vorhergehenden Daten werden in einer Objektliste (OBJLIST) platziert. Als nächstes wird bei einem Block **608** der Zähler inkrementiert, so dass eine andere Aufzeichnung der TLIST bedient wird, falls jegliche verbleiben. Der Block **608** tritt zu einem Block **610** über. Bei dem Block **610** wird durch ein Vergleichen des Aufzeichnungszählwerts des Aufzeichnungszählers mit der Gesamtanzahl von bereits verarbeiteten Aufzeichnungen bestimmt, ob jegliche Ereignisse übrig sind, die bedient werden sollen. Falls dem so ist, dann tritt der Block **610** zurück zu dem Block **606** über, der beginnt, eine weitere Aufzeichnung zu bedienen. Falls nicht, dann tritt der Block **610** zu einem Block **612** über, der eine Anforderung nach einer Stapelübertragung von Objektinformationen, die zu allen der Objekte innerhalb des Stapels gehören, zu dem Topologieverwalter **310** sendet. Fig. 7 zeigt ein Flussdiagramm der Architektur und Funktionalität eines bevorzugten Ausführungsbeispiels des Abbildungsveränderungen-Berechnen-Blocks **406** (Fig. 4). Bei diesem Flussdiagramm bestimmt der Übersetzer, welche Teilabbildungen (Fig. 2) verändert sind und die Veränderung, die bewirkt werden soll, basierend auf den Objektidentifizierungen und den Objektveränderungen, die vorhergehend basierend auf den Ereignissen bestimmt wurden. Mit Bezug auf Fig. 7 leitet ein Block **701** einen Objektveränderungszähler ein, so dass alle Objektveränderungen betrachtet werden. Der Block **701** tritt zu einem Block **702** über. Der Block **702** bestimmt einen Teilabbildungsidentifizierer basierend darauf, welche der Teilabbildungen (Fig. 2) durch die Objektveränderung beeinflusst sind, um die es sich gegenwärtig handelt. Der Block **702** tritt zu einem Block **704** über, der bestimmt, ob die beeinflusste Teilabbildung existiert. Falls die Teilabbildung existiert, dann tritt der Block **704** zu dem Block **710** über. Falls die Teilabbildung nicht existiert, dann tritt der Block **704** zu dem Block **705** über. Bei dem Block **705** wird eine Bestimmung vorgenommen, ob die fragliche Teilabbildung basierend auf der Persistenzspezifikation der vorliegenden Erfindung erzeugt werden sollte, d. h. ob die Teilabbildung persistente Objekte enthält. Die Persistenzspezifikation wird mit Bezug auf Fig. 15 – Fig. 18 hierin im Folgenden ausführlicher beschrieben (insbesondere Fig. 17 ist ein Flussdiagramm, das die spezifische Architektur des Blocks **705** darstellt). Falls die Teilabbildung nicht hinzugefügt werden soll, dann tritt der Block **705** zu einem Block **716** über. Falls die Teilabbildung hinzugefügt werden soll, dann tritt der Block **705** zu einem Block **706** über, der die beeinflusste Teilabbildung in der **Abb.** 200 (Fig. 2) erzeugt. Der Block **706** tritt zu einem Block **708** über.

[0071] Bei dem Block **708** bestückt der Übersetzer **318** die neuerzeugte Teilabbildung mit Abbildungsdaten, die aus der Topologiedatenbank **326** wiedererlangt werden. Bei dem Block **710** werden als nächstes Teilabbildungsveränderungen basierend auf dem aktuellen Ereignis, insbesondere dem Objektidentifizierer und der Objektveränderung, berechnet. Diese Berechnungen des Blocks **710** werden hierin im Folgenden mit Bezug auf Fig. 8 beschrieben. Der Block **710** tritt zu einem Block **712** über, der eine Bestimmung vornimmt, ob das fragliche Objekt die neuartige Persistenzspezifikation einhält. Die Persistenzspezifikation wird wiederum hierin im Folgenden mit Bezug auf Fig. 15 – Fig. 18 ausführlicher beschrieben (insbesondere ist Fig. 18 ein Flussdiagramm, das die spezifische Architektur des Blocks **712** darstellt). Falls das Objekt die Persistenzspezifikation einhält, dann tritt der Block **712** zu einem Block **714** über, der die fragliche Teilabbildung als persistent identifiziert, und dann tritt der Block **714** zu einem Block **716** über. Falls das Objekt die Persistenzspezifikation nicht einhält, dann tritt der Block **712** zu dem Block **716** über.

[0072] Bei dem Block **716** wird der Objektveränderungszähler inkrementiert, so dass eine weitere Objektveränderung mit Bezug auf die Teilabbildungen betrachtet wird. Der Block **716** tritt zu einem Block **718** über, der eine Bestimmung vornimmt, ob jegliche Objektveränderungen verbleiben, um bedient zu werden. Falls dem so ist, dann tritt der Block **718** zurück zu dem Block **702** über. Falls nicht, dann endet das

Flussdiagramm nach dem Block **718**.

[0073] Daher wurde bei dem Abschluss der Operation der Logik in Fig. 7 ein Stapel von Teilabbildungsidentifizierern mit zugeordneten Teilabbildungsveränderungen aus einem Stapel von Objektidentifizierern mit zugeordneten Objektveränderungen erzeugt.

[0074] Mit Bezug auf Fig. 8 erlangt bezüglich der Teilabbildungsveränderungsberechnungen von Block **710** (Fig. 7) ein Block **804** Daten wieder, die ein einziges Objekt aus der OBJLIST betreffen. Der Block **804** geht zu einem Block **806** über, der bestimmt, ob der Objekttyp ein Netzwerk ist. Falls dem so ist, dann tritt der Block **806** zu einem Block **808** über (Flussdiagramm in Fig. 9), der die Teilabbildungsveränderungen berechnet und dann endet das Flussdiagramm. Falls nicht, dann tritt der Block **806** zu dem Block **810** über.

[0075] Bei dem Block **810** wird eine Bestimmung vorgenommen, ob der Objekttyp ein Segment ist. Falls dem so ist, dann tritt der Block **810** zu dem Block **812** über (Flussdiagramm von Fig. 10), der die Segmentveränderungen an den Teilabbildungen berechnet, und dann endet das Flussdiagramm. Falls nicht, dann tritt der Block **810** zu dem Block **814** über.

[0076] Bei dem Block **814** wird eine Bestimmung vorgenommen, ob der Objekttyp ein Knoten ist. Falls dem so ist, dann tritt der Block **814** zu dem Block **816** über (Flussdiagramm von Fig. 11), der die Knotenveränderungen für die Teilabbildungen berechnet, und dann endet das Flussdiagramm. Falls nicht, dann tritt der Block **816** zu dem Block **818** über.

[0077] Bei dem Block **818** wird eine Bestimmung vorgenommen, ob der Objekttyp eine Schnittstelle ist. Falls dem so ist, dann tritt der Block **818** zu dem Block **820** über (Flussdiagramm von Fig. 12), der die Schnittstellenveränderungen an der Teilabbildung berechnet, und dann endet das Flussdiagramm. Falls nicht, dann endet das Flussdiagramm.

[0078] Fig. 9 zeigt ein Flussdiagramm der Architektur und Funktionalität eines bevorzugten Ausführungsbeispiels zum Implementieren des Netzwerkänderungsblocks **808** (Fig. 8). Dieses Flussdiagramm berechnet Veränderungen an der Internet-Teilabbildung **204** (Fig. 2), die die Netzwerke anzeigt. Außerdem ist anzumerken, dass es bei dem bevorzugten Ausführungsbeispiel lediglich eine einzige Teilabbildung (mehrere Teilabbildungen sind möglich) auf der Internet-Ebene gibt. Mit Bezug auf Fig. 9 wird bei einem Block **902** eine Variable INET gesetzt, um die Inhalte der Internet-Teilabbildung **204** (Fig. 2) anzunehmen. Die Inhalte umfassen eine Liste von Netzwerkobjekten und Router-Objekten und eine Liste von Verbindungen zwischen dem Netzwerk und Router-Objekten. Der Block **902** tritt zu einem Block **904** über. Bei dem Block **904** wird eine Variable NETOBJ gesetzt, um den Wert des Objektidentifizierers (OBJID) anzunehmen. Der OBJID wird aus der OBJINFO wiedererlangt. Der Block **904** tritt zu einem Block **906** über. Bei dem Block **906** wird eine Bestimmung vorgenommen, ob NETOBJ in INET ist, d. h. ob das Objekt, das verändert werden soll, innerhalb der Internet-Teilabbildung **204** (Fig. 2) liegt. Falls dem so ist, dann tritt der Block **906** zu dem Block **908** über, der das Netzwerk, das zu dem NETOBJ gehört, zu einer Liste SYMCHANGELIST hinzufügt. Falls nicht, dann tritt der Block **906** zu dem Block **910** über, der das Netzwerk, das zu der NETOBJ gehört, zu einer Liste NEWSYMLIST hinzufügt. Die Listen SYMCHANGELLIST und NEWSYMLIST werden schließlich während der Stapelübertragung zwischen denselben durch den Übersetzer **318** zu der GUI **322** weitergeleitet.

[0079] Fig. 10 zeigt ein Flussdiagramm der Architektur und Funktionalität eines bevorzugten Ausführungsbeispiels zum Implementieren des Segmentänderungsblocks **812** (Fig. 8). Bei diesem Flussdiagramm werden Segmentveränderungen bestimmt und berechnet. Mit Bezug auf Fig. 10 setzt ein Block **1002** eine Variable INET, um die Inhalte der Internet-Teilabbildung **204** (Fig. 2) anzunehmen. Die Inhalte umfassen eine Liste von Netzwerk- und Router-Objekten und eine Liste von Verbindungen zwischen den Netzwerk- und den Router-Elementen. Der Block **1002** tritt zu einem Block **1004** über. Bei dem Block **1004** wird eine Variable SEGOBJ gesetzt, um den aktuellen Objektidentifizierer (OBJID) anzunehmen, der aus den Objektinformationen (OBJINFO) wiedererlangt wird. Der Block **1004** tritt zu einem Block **1006** über. Bei dem Block **1006** wird eine Variable NETOBJ zu dem identifizierten Netzwerk (NETID) gesetzt, das aus den OBJINFO bestimmt wird. Der Block **1006** tritt zu einem Block **1008** über. Bei dem Block **1008** wird eine Bestimmung vorgenommen, ob NETOBJ in INET ist, d. h. ob das aktuelle Netzwerk innerhalb der aktuellen Internet-Teilabbildung **204** (Fig. 2) ist. Falls nicht, dann endet das Flussdiagramm von Fig. 10. Falls dem so ist, dann tritt der Block **1008** zu einem Block **1010** über. Bei dem Block **1010** wird eine Variable NET gesetzt, um die Inhalte der Netzwerk-Teilabbildung **206** (Fig. 2) anzunehmen, die zu NETOBJ gehört. Die Inhalte umfassen zum Beispiel, aber nicht darauf begrenzt, eine Liste von Segment- und Verbinderobjekten und Verbindungen zwischen einem Segment und Verbindern. Der Block **1010** tritt zu einem Block **1012** über. Bei dem Block **1012** wird eine Bestimmung vorgenommen, ob SEGOBJ in der NET ist (d. h. ist das Segment in der Netzwerk-Teilabbildung?). Falls dem so ist, dann tritt der Block **1012** zu dem Block **1014** über, der das Segment, das zu SEGOBJ gehört, zu der SYMCHANGELIST hinzufügt. Falls nicht, tritt andernfalls der Block **1012** zu dem Block **1016** über, der das Segment, das zu SEGOBJ gehört, zu NEWSYM-

LIST hinzufügt. Schließlich endet das Flussdiagramm von Fig. 10 nach den Blöcken **1014**, **1016** und eine Operation tritt zurück zu Fig. 8 über.

[0080] Fig. 11 zeigt ein Flussdiagramm der Architektur und Funktionalität eines bevorzugten Ausführungsbeispiels zum Implementieren des Knotenveränderungsblocks **816** (Fig. 8). Bei dem Flussdiagramm von Fig. 11 werden Knotenveränderungen durch den Übersetzer **318** bestimmt und berechnet. Wie es in Fig. 11 gezeigt ist, setzt ein Block **1102** eine Variable INET, um die Inhalte der Internet-Teilabbildung **202** (Fig. 2) anzunehmen. Die Inhalte umfassen eine Liste von Netzwerk- und Router-Objekten und eine Liste von Verbindungen zwischen den Netzwerk- und Router-Elementen. Der Block **1102** tritt zu einem Block **1104** über. Bei dem Block **1104** wird eine Variable NODEOBJ gesetzt, um den Objektidentifizierer (OBJID) anzunehmen, der in den Objektinformationen (OBJINFO) enthalten ist. Der Block **1104** tritt zu einem Block **1106** über. Bei dem Block **1106** wird eine Variable SEGOBJ gesetzt, um den Segmentidentifizierer (SEGID) anzunehmen, der innerhalb der OBJINFO enthalten ist. Der Block **1106** tritt zu einem Block **1108** über. Bei dem Block **1108** wird eine Variable NETOBJ gesetzt, um den Netzwerkidentifizierer (NETID) anzunehmen, der innerhalb der OBJINFO enthalten ist. Der Block **1108** tritt zu einem Block **1110** über. Bei dem Block **1110** wird eine Bestimmung vorgenommen, ob NETOBJ in INET ist (d. h. ist das Netzwerk in der Internet-Teilabbildung?). Falls nicht, dann endet das Flussdiagramm. Falls dem so ist, dann tritt der Block **1110** zu dem Block **1112** über. Bei dem Block **1112** wird die Variable NET gesetzt, um die Inhalte der Netzwerk-Teilabbildung **206** (Fig. 2) anzunehmen, die zu NETOBJ gehört. Die Inhalte umfassen zum Beispiel, aber nicht darauf begrenzt, eine Liste von Segmenten, Verbindern und Verbindungen zwischen Segmenten und Verbindern. Der Block **1112** tritt zu einem Block **1114** über. Bei dem Block **1114** wird eine Bestimmung vorgenommen, ob SEGOBJ in NET ist. Falls nicht, dann endet das Flussdiagramm. Falls dem so ist, dann tritt der Block **1114** zu dem Block **1116** über. Bei dem Block **1116** wird die Variable SEG gesetzt, um die Inhalte der Segment-Teilabbildung **208** (Fig. 2) anzunehmen, die zu SEGOBJ gehört. Die Inhalte umfassen zum Beispiel, aber nicht darauf begrenzt, eine Liste von Knoten und Verbindungen zwischen den Knoten und dem Netzwerk. Der Block **1116** geht zu einem Block **1118** über. Bei dem Block **1118** wird eine Bestimmung vorgenommen, ob NODEOBJ in SEG ist, d. h. ob das Knotenobjekt in dem vorliegenden fraglichen Segment ist. Falls dem so ist, dann tritt der Block **1118** zu dem Block **1120** über, der den Knoten, der zu NODEOBJ gehört, zu SYMCHANGELIST hinzufügt, und dann endet das Flussdiagramm. Falls nicht, tritt andernfalls der Block **1118** zu dem Block **1122** über, der den Knoten, der zu NODEOBJ gehört, zu NEWSYMLIST hinzufügt, und dann endet das Flussdiagramm.

[0081] Fig. 12A bis Fig. 12B zeigen gemeinsam ein Flussdiagramm der Architektur und Funktionalität des bevorzugten Ausführungsbeispiels zum Implementieren des Schnittstellenveränderungsblocks **820** (Fig. 8). Bei diesem Flussdiagramm werden Schnittstellenveränderungen in den Teilabbildungen durch den Übersetzer **318** (Fig. 3) bestimmt und berechnet. Mit Bezug auf Fig. 12A setzt ein Block **1202** eine Variable INET, um die Inhalte der Internet-Teilabbildung **204** (Fig. 2) anzunehmen, um die es sich gegenwärtig handelt. Die Inhalte umfassen eine Liste von Netzen, Routern und Verbindungsobjekten. Der Block **1202** tritt zu einem Block **1204** über. Bei dem Block **1204** wird eine Variable IFOBJ gesetzt, um den OBJID anzunehmen, der innerhalb der OBJINFO enthalten ist. Der Block **1204** tritt zu dem Block **1206** über. Bei dem Block **1206** wird die Variable NODEOBJ gesetzt, um den NODEID anzunehmen, der innerhalb der OBJINFO enthalten ist. Der Block **1206** tritt zu einem Block **1208** über. Bei dem Block **1208** wird die Variable SEGOBJ gesetzt, um den SEGID anzunehmen, der innerhalb der OBJINFO enthalten ist. Der Block **1208** tritt zu einem Block **1210** über. Bei dem Block **1210** wird eine Variable NETOBJ gesetzt, um den NETID anzunehmen, der innerhalb der OBJINFO enthalten ist. Nach dem Block **1210** wurde der Initialisierungsprozess abgeschlossen und der Block **1210** tritt zu einem Block **1212** über.

[0082] Bei dem Block **1212** wird eine Bestimmung vorgenommen, ob NETOBJ in INET ist, d. h. ob das Netzwerkobjekt in der aktuellen Internet-Teilabbildung **204** (Fig. 2) ist. Falls nicht, endet das Flussdiagramm, wie es in Fig. 12A gezeigt ist. Falls dem so ist, dann tritt der Block **1212** zu einem Block **1214** über. Bei dem Block **1214** wird eine Bestimmung vorgenommen, ob NODEOBJ in INET ist, d. h. ob das Knotenobjekt in der Internet-Teilabbildung **204** (Fig. 2) ist. Falls nicht, dann tritt der Block **1214** zu dem Block **1222** über. Falls dem so ist, dann tritt der Block **1214** zu dem Block **1216** über.

[0083] Bei dem Block **1216** wird eine Bestimmung vorgenommen, ob IFOBJ in INET ist. Falls dem so ist, dann tritt der Block **1216** zu dem Block **1218** über, der die Schnittstelle, die zu IFOBJ gehört, zu der SYMCHANGELIST hinzufügt. Falls nicht, dann tritt der Block **1216** zu einem Block **1220** über, der die Schnittstelle, die zu IFOBJ (zwischen Knotenobjekt und Netzwerkobjekt) gehört, zu NEWSYMLIST hinzufügt.

[0084] Bei dem Block **1222** wird die Variable NET gesetzt, um die Inhalte der Netzwerk-Teilabbildung **206** (Fig. 2) anzunehmen. Die Inhalte umfassen zum Beispiel, aber nicht darauf begrenzt, Segmente, Verbinder, Verbindungen, etc. Der Block **1222** tritt zu einem Block **1224** von Fig. 12B über.

[0085] Bei dem Block **1224** wird eine Bestimmung vorgenommen, ob SEGOBJ in NET ist, d. h. ob das Segmentobjekt innerhalb der Netzwerk-Teilabbildung **206** (Fig. 2) ist. Falls nicht, dann endet das Flussdiagramm. Falls dem so ist, dann tritt der Block **1224** zu dem Block **1226** über.

[0086] Bei dem Block **1226** wird eine Bestimmung vorgenommen, ob NODEOBJ in NET ist, d. h. ob das Knotenobjekt innerhalb der Netzwerk-Teilabbildung **206** (Fig. 2) ist. Falls nicht, dann tritt das Flussdiagramm zu einem Block **1234** über. Falls dem so ist, dann tritt der Block **1226** zu einem Block **1228** über.

[0087] Bei dem Block **1228** wird eine Bestimmung vorgenommen, ob IFOBJ innerhalb NET ist, d. h. ob das Schnittstellenobjekt innerhalb der Netzwerk-Teilabbildung **206** (Fig. 2) ist. Falls dem so ist, dann tritt der Block **1228** zu dem Block **1230** über, der die Schnittstelle, die zu IFOBJ gehört, zu SYMCHANGELIST hinzufügt. Falls nicht, dann tritt der Block **1228** zu dem Block **1232** über, der die Schnittstelle, die zu IFOBJ (die zwischen einem Knotenobjekt und einem Segmentobjekt ist) gehört, zu NEWSYMLIST hinzufügt. Die Blöcke **1230**, **1232** treten zu dem Block **1234** über.

[0088] Bei dem Block **1234** wird die Variable SEG gesetzt, um die Inhalte der Segment-Teilabbildung **208** (Fig. 2) anzunehmen. Die Inhalte umfassen zum Beispiel, aber nicht begrenzt darauf, Knoten und Verbindungen zu einem Netzwerk (Schnittstellen). Der Block **1234** tritt zu einem Block **1236** über.

[0089] Bei dem Block **1236** wird eine Bestimmung vorgenommen, ob NODEOBJ in SEG ist, d. h. ob das Knotenobjekt innerhalb der Segment-Teilabbildung **208** (Fig. 2) ist. Falls nicht, dann tritt das Flussdiagramm zu einem weiteren Block über. Falls dem so ist, dann tritt der Block **1236** zu dem Block **1238** über.

[0090] Bei dem Block **1238** wird eine Bestimmung vorgenommen, ob IFOBJ innerhalb SEG ist, d. h. ob das Schnittstellenobjekt innerhalb der Segment-Teilabbildung **208** (Fig. 2) ist. Falls dem so ist, dann tritt der Block **1238** zu dem Block **1248** über, der die Schnittstelle, die zu IFOBJ gehört, zu SYMCHANGELIST hinzufügt. Falls nicht, dann tritt der Block **1238** zu dem Block **1244** über, der die Schnittstelle, die zu IFOBJ gehört, zu NEWSYMLIST hinzufügt. Die Blöcke **1242**, **1244** treten zu dem weiteren Block über, bei dem die Variable NODE gesetzt wird, um die Inhalte der Knoten-Teilabbildung **210** (Fig. 2) anzunehmen. Die Inhalte umfassen Schnittstellenobjekte. Dieser Block tritt zu einem zweiten weiteren Block über.

[0091] Es wird eine Bestimmung vorgenommen, ob IFOBJ innerhalb NODE ist, d. h. ob das Schnittstellenobjekt innerhalb der Knoten-Teilabbildung **210** (Fig. 2) ist. Falls dem so ist, dann wird die Schnittstelle, die zu IFOBJ gehört, zu SYMCHANGELIST hinzugefügt. Falls nicht, wird die Schnittstelle, die zu IFOBJ gehört, zu NEWSYMLIST hinzugefügt. Schließlich endet das Flussdiagramm, das gemeinsam in Fig. 12A bis Fig. 12B enthalten ist.

[0092] Fig. 13 zeigt ein Flussdiagramm der Architektur und Funktionalität eines bevorzugten Ausführungsbeispiels zum Implementieren des Abbildung-Aktualisieren-Blocks **408** (Fig. 4). Bei diesem Flussdiagramm wird eine Stapelübertragung von Veränderungen durch den Übersetzer **318** zu der GUI **322** gesendet. Mit Bezug auf Fig. 13 überträgt bei einem Block **1302** der Übersetzer **318** die NEWSYMLIST zu der GUI **322** und bei einem Block **1304** überträgt der Übersetzer **318** die SYMCHANGELIST zu der GUI **322**. Nach dem Block **1304** endet das Flussdiagramm von Fig. 13 und die Operation geht zurück zu dem Block **410** (Fig. 4) über.

[0093] Fig. 14 stellt ein Auf-Anforderung-Teilabbildung-Modul dar, das innerhalb der GUI **322** (Fig. 3) enthalten ist. Dieses Flussdiagramm implementiert die Benutzerschnittstelle zu den verschiedenen Teilabbildungen der **Abb.** 200 (Fig. 2). Mit Bezug auf Fig. 14 überwacht bei einem Block **1402** die GUI **322** die Eingabegeräte, die mit der Verwaltungsstation **100** (Fig. 1) verbunden sind, z. B. die Tastatur **106**. Wenn der Benutzer der Verwaltungsstation **100** die Verwaltungsstation **100** über die Tastatur **106** oder über ein gewisses anderes Eingabegerät auffordert, ein Objekt auf der Anzeige **108** explodieren zu lassen, tritt der Block **1402** von

[0094] Fig. 14 zu dem Block **1404** über, um die Benutzeranforderung zu verarbeiten. Bei dem Block **1404** wird eine Bestimmung vorgenommen, ob die Tochter-Teilabbildung innerhalb der **Abb.** 200 (Fig. 2) enthalten ist. Falls dem so ist, dann tritt der Block **1404** zu dem Block **1408** über. Falls nicht, dann tritt der Block **1404** zu dem Block **1406** über, der die Teilabbildung erzeugt und bestückt. Die GUI **322** bestückt die Teilabbildung durch ein Auffordern des Übersetzers **318**, eine Teilabbildung basierend auf Topologiedaten zu erzeugen und zu bestücken, die von dem Topologieverwalter **310** wiedererlangt werden. Außerdem tritt der Block **1406** zu dem Block **1408** über, der die Tochter-Teilabbildung öffnet und die Tochter-Teilabbildung auf der Anzeige **108** für den Benutzer anzeigt.

Persistenzspezifikation

[0095] Das Konzept von Auf-Anforderung-Teilabbildungen, wie dasselbe mit Bezug auf Fig. 2 dargestellt und beschrieben ist, besteht darin, lediglich diese Teilabbildungen in der **Abb.** 200 (Fig. 2) zu platzieren, die der Benutzer auf der Anzeige **108** (Fig. 1) sehen will. Bei der Implementierung des Persistenz-

spezifikationsmechanismus **103** (Fig. 1) hierin ist dieses Konzept erweitert, um Teilabbildungen zu umfassen, die durch Integrationsanwendungen **332** (Fig. 3) benötigt werden, die sich während einer Operation dynamisch verändern können. Das Nettoergebnis ist, dass lediglich ein Abschnitt der Teilabbildungshierarchie zu einer gegebenen Zeit in der **Abb.** 200 (Fig. 2) ist und die Informationen, die zu dem Benutzer geliefert werden, wesentlich durch ein Ermöglichen verbessert werden, dass Integrationsanwendungen **332** Gerätekonfigurations-Informationen ergänzen, die durch den Entdeckungsmechanismus **302** (Fig. 3) aus dem Netzwerk **118** entdeckt wurden.

[0096] Die Persistenzspezifikation betrifft ein Definieren jeder Teilabbildung der **Abb.** 200 (Fig. 2) als entweder persistent oder transient, wie es in dem Diagramm **1500** von Fig. 15 angegeben ist. Gleichgültig was durch die Persistenzspezifikation spezifiziert wird, hat der Benutzer immer noch einen Zugriff auf die gesamte Topologie durch die **Abb.** 200. Diese Teilabbildungen, die als persistent bezeichnet sind, werden jedoch unmittelbar nach dieser Bestimmung an der **Abb.** 200 platziert, und diese Abbildungen, die als transient bezeichnet sind, werden lediglich auf Anforderung nach einer Benutzeraufforderung erzeugt, und dann, wenn der Benutzer die transiente Teilabbildung verlässt, wird die transiente Teilabbildung von der **Abb.** 200 (Fig. 2) entfernt.

[0097] Die Persistenzspezifikation ist nützlich, wenn es eine Integrationsanwendung **332** (Fig. 3) gibt, die eng mit der Teilabbildungshierarchie integriert ist. Mit anderen Worten ist dieselbe nützlich, wo die Integrationsanwendung **332** davon abhängt, dass der Übersetzer **318** bestimmte Objekte in der **Abb.** 200 platziert, damit die Integrationsanwendung **332** ordnungsgemäß wirksam ist. Zum Beispiel kann eine Firma eine Integrationsanwendung **332** entwickelt haben, die einem Router in einer Knoten-Teilabbildung ein Blob-Symbol hinzufügt und den Status der Blobs benötigt, um sich in der Teilabbildungshierarchie nach oben auszubreiten. Folglich würde die Integrationsanwendung **332** den Router benötigen, der in der Persistenzspezifikation spezifiziert ist.

[0098] Um die Persistenzspezifikation zu implementieren, wird ein Persistenzvektor **1600** von Fig. 16 jeder der Teilabbildungen der **Abb.** 200 (Fig. 2) durch die GUI **322** (Fig. 3) zugeordnet. Der Persistenzvektor **1600** weist zahlreiche Persistenzbits auf. Ein Persistenzbit b_U entspricht dem Benutzer des Computersystems **100** und wird durch die GUI **322** erzeugt. Außerdem ist ein Persistenzbit $b_{A1}...b_{An}$ vorgesehen, das jeder der Integrationsanwendungen **332** (Fig. 3) entspricht, die der Entdeckungs-/Layout-Software **101** (Fig. 1, Fig. 3) zugeordnet ist. Damit eine Teilabbildung als transient betrachtet wird, darf keines der persistenten Bits b_U, $b_{A1}...b_{An}$ in dem Persistenzvektor **1600** aktiviert sein. Wenn umgekehrt ein jegliches der Bits b_U, $b_{A1}...b_{An}$ des Persistenzvektors **1600** aktiviert ist, dann wird die jeweilige Teilabbildung als persistent betrachtet.

[0099] Unter erneuter Bezugnahme auf das Zustandsdiagramm in Fig. 15 geht eine Teilabbildung unter den folgenden Umständen von transient zu persistent über: (a) der Benutzer oder eine Integrationsanwendung **332** nimmt eine Veränderung (z. B. ein Hinzufügen einer Hintergrundgraphik, ein Bewegen eines Symbols, ein Verändern eines Symboletiketts, ein Verändern bei einem automatischen oder manuellen Layout oder ein Verstecken eines Symbols) mit Bezug auf ein Objekt vor und die Veränderung beeinflusst nichts, was in der Topologiedatenbank **314** gespeichert ist; oder (b) eine Integrationsanwendung Ai aktiviert das entsprechende Persistenzbit b_{Ai} derselben. Eine Teilabbildung ändert sich von dem persistenten Zustand zu dem transienten Zustand, wenn eine Integrationsanwendung das entsprechende Persistenzbit derselben deaktiviert, falls dies darin resultiert, dass alle Persistenzbits des Persistenzvektors **1600** deaktiviert werden. Folglich kann somit ein Benutzer eine persistente Teilabbildung erzeugen, während eine Integrationsanwendung entweder eine persistente oder eine transiente Teilabbildung erzeugen kann.

[0100] Man rufe sich in Erinnerung, dass in Fig. 7 der Block **705** eine Anfrage vornimmt, ob die fragliche Teilabbildung zu der **Abb.** 200 (Fig. 2) hinzugefügt werden sollte, basierend auf der Persistenzspezifikation. Fig. 17 ist ein Flussdiagramm, das die Architektur und Funktionalität eines bevorzugten Ausführungsbeispiels des Blocks **705** darstellt. Wie es in Fig. 17 gezeigt ist, wird bei einem Block **1702** eine Variable OBJINFO gesetzt, um jeden der Objektidentifizierer innerhalb der Teilabbildung anzunehmen, und jeder der Objektidentifizierer wird über einen Zähler betrachtet. Der Block **1702** tritt zu einem Block **1704** über.

[0101] Bei dem Block **1704** wird eine Bestimmung vorgenommen, ob das spezielle fragliche Objekt die Persistenzspezifikation einhält. Falls nicht, dann tritt der Block **1704** zu dem Block **712** (Fig. 7) über. Falls dem so ist, dann tritt der Block **1704** zu dem Block **1706** über, der die Teilabbildung, die das Objekt enthält, als eine persistente Teilabbildung durch ein Aktivieren des Persistenzbits (eines von b_U, $b_{A1}...b_{An}$) definiert, das der fraglichen Teilabbildung entspricht. Der Block **1706** tritt dann zu dem Block **706** (Fig. 7) über.

[0102] Fig. 18 zeigt ein Flussdiagramm zum Bestimmen, ob ein Objekt als entweder persistent oder transient klassifiziert werden sollte, wie es in Block **712** (Fig. 7) und ebenfalls in Block **1704** (Fig. 17) bestimmt ist. Unter Bezugnahme auf Fig. 18 setzt ein

Block **1802** eine Variable FILTEREXPR, um eine Liste von Feldern und Werten anzunehmen, die von dem Übersetzer **103** zu der Integrationsanwendung **322** übertragen werden. Die Felder und Werte sind im Wesentlichen Flags für die Integrationsanwendungen **332** und identifizieren ergänzende Informationen, die durch die Integrationsanwendungen **332** geliefert werden können. Ein Feld identifiziert einen gewissen Parameter eines Objekts (zum Beispiel, aber nicht begrenzt auf, einen Verkäufer (Hersteller), einen Gerätetyp, eine Adresse, eine Rate (z. B. Pakete/Sekunde), etc.). Der Wert ist einfach ein Wert für dieses Feld (zum Beispiel, aber nicht begrenzt auf, CISCO, Router, 15, 2.112.337, 55 Pakete/Sekunde, etc.). Also sind für ein Verkäuferfeld mögliche Werte z. B. Hewlett-Packard, CISCO, International Business Machines (IBM), etc. Für einen Gerätetyp sind mögliche Werte z. B. ein Router, ein Drucker, etc.

[0103] Ferner tritt der Block **1802** zu einem Block **1804** über, der einen Zähler zum Zweck eines Betrachtens aller Paarungen von Feldern und Werten mit Bezug auf das fragliche Objekt einleitet. Der Block **1804** tritt in die Schleife über, die bei einem Block **1806** beginnt.

[0104] Bei dem Block **1806** wird eine Variable EXPR gesetzt, um ein Feld und einen Wert anzunehmen. Der Block **1806** tritt zu einem Block **1808** über, der eine Variable EXPRVAL setzt, um den Wert (EXPR.VALUE) innerhalb der Variablen EXPR anzunehmen. Der Block **1808** tritt zu einem Block **1810** über.

[0105] Bei dem Block **1810** wird eine Variable OBJVAL gesetzt, um den Objektwert, der zu dem fraglichen Objekt gehört, innerhalb des speziellen Felds (EXPR. FIELD) von EXPR anzunehmen. Der Block **1810** tritt zu einem Block **1812** über.

[0106] Bei dem Block **1812** wird OBJVAL mit EXPRVAL verglichen, d. h. der Objektwert wird mit dem Integrationsanwendungswert oder Testwert verglichen. Falls der Objektwert nicht mit dem Integrationsanwendungswert übereinstimmt, dann hält das Objekt die persistente Spezifikation nicht ein, wie es bei einem Block **1814** angegeben ist, und das Flussdiagramm endet. Falls der Objektwert jedoch mit allen der Integrationsanwendungswerte übereinstimmt, dann wird das Objekt schließlich bei einem Block **1820** persistent gemacht. Vor einem Erreichen des Blocks **1820** jedoch tritt der Block **1812** zu einem Block **1816** über, der den Zähler inkrementiert, der bei dem Block **1804** eingeleitet wird. Außerdem tritt der Block **1816** zu einem Block **1818** über, der bestimmt, ob alle EXPRs betrachtet wurden. Falls einige verbleiben, dann tritt der Block **1818** zurück zu dem Block **1806** über und der vorhergehende Prozess geht weiter. Falls keine EXPRs mehr verbleiben, dann tritt das Flussdiagramm zu dem Block **1820** über, der das Objekt als die Persistenzspezifikation einhaltend spezifiziert, und dann endet das Flussdiagramm.

[0107] Wenn ein Objekt definitiv mit einem Feld-/Wert-Paar in Übereinstimmung gebracht ist und als persistent bezeichnet ist, benachrichtigt der Übersetzer **103** die GUI **322** über diese Tatsache und dann informiert die GUI **322** die Integrationsanwendung **332** über die Existenz des persistenten Objekts. Die GUI **322** wiederum liefert ergänzende Informationen, wie es durch einen Pfeil **333b** angegeben ist, die zu dem persistenten Objekt gehören und/oder auf demselben basieren, zu der GUI **322** für eine Einbringung in die geeigneten Teilabbildungen und für eine Anzeige.

Patentansprüche

1. Ein Persistenzspezifikationssystem (**100'**), das folgende Merkmale aufweist:
einen Prozessor (**102**);
einen Entdeckungsmechanismus (**302**), der dem Prozessor (**102**) zugeordnet ist, wobei der Entdeckungsmechanismus (**302**) konfiguriert ist, um Topologiedaten zu erzeugen und zu speichern, die Geräte und Verbindungen eines Netzwerks (**118**) angeben; und
einen Layoutmechanismus (**304**), der dem Prozessor (**102**) zugeordnet und in Kommunikation mit dem Entdeckungsmechanismus (**302**) ist, wobei der Layoutmechanismus (**304**) konfiguriert ist, um die Topologiedaten von dem Entdeckungsmechanismus (**302**) zu empfangen, wobei der Layout-Mechanismus (**304**) konfiguriert ist, um eine Anzeige (**108**) basierend auf den Topologiedaten zu treiben, wobei der Layoutmechanismus (**304**) folgende Merkmale aufweist:
einen Übersetzer, der konfiguriert ist, um die Topologiedaten in die Abbildungsdaten umzuwandeln, wobei der Übersetzer eine Persistenzspezifikationseinrichtung (**103**) aufweist, wobei die Persistenzspezifikationseinrichtung (**103**) zum Spezifizieren basierend auf einem Persistenzeingangssignal, wenn ein Objekt (**203**), das angezeigt werden soll, persistent ist, und alternativ, wenn das Objekt (**203**) transient ist, ist, wobei die Persistenzspezifikationseinrichtung (**103**) zum Definieren der Abbildungsdaten als persistent ist, wenn die Abbildungsdaten ein persistentes Objekt (**203**) aufweisen, und zum Definieren der Abbildungsdaten als transient, wenn die Abbildungsdaten ohne ein persistentes Objekt (**203**) sind, wobei der Übersetzer konfiguriert ist, um die persistenten Abbildungsdaten zu erzeugen und kontinuierlich beizubehalten, und konfiguriert ist, um transiente Abbildungsdaten zu erzeugen und während einer Zeitperiode entsprechend einer Anforderung durch einen Benutzer beizubehalten; und
eine Schnittstelle (**322**), die konfiguriert ist, um die Abbildungsdaten von dem Übersetzer zu empfangen und um die Anzeige (**108**) basierend auf den Abbil-

dungsdaten zu treiben.

2. Das Persistenzspezifikationssystem (**100'**) gemäß Anspruch 1 zum Verbessern einer Kommunikation zwischen einer Integrationsanwendung (**322**) und der Schnittstelle (**322**), so daß mehr Informationen, die ein Netzwerk (**118**) betreffen, zu einem Benutzer geliefert werden, während Speichererfordernisse und eine Verarbeitungszeit minimiert sind, wobei:
der Entdeckungsmechanismus (**302**) eine Topologiedatenbank (**314**) zum Speichern der Topologiedaten aufweist;
der Layoutmechanismus (**304**) eine Abbildungsdatenbank (**326**) zum Speichern einer Mehrzahl von Teilabbildungen (**202** – **210**) von Abbildungsdaten für die Schnittstelle (**322**) aufweist, wobei die Teilabbildungen (**202** – **210**) für ein Treiben der Anzeige sind;
der Übersetzer (**318**) konfiguriert ist, um die Topologiedaten von der Topologiedatenbank (**314**) in Abbildungsdaten für die Abbildungsdatenbank (**326**) umzuwandeln, wobei der Übersetzer (**318**) konfiguriert ist, um Teilabbildungen (**202** – **210**) aus den Abbildungsdaten für die Schnittstelle (**322**) zum Treiben der Anzeige (**108**) zu erzeugen;
die Persistenzspezifikationseinrichtung (**103**) konfiguriert ist, um jede Teilabbildung (**202** – **210**) auszuwerten, und konfiguriert ist, um eine Teilabbildung (**202** – **210**) basierend auf der Teilabbildungsauswertung als persistent und alternativ als transient zu spezifizieren, wobei die Teilabbildung (**202** – **210**) als persistent spezifiziert wird, wenn die Teilabbildung (**202** – **210**) ein persistentes Objekt (**203**) aufweist, wobei die Teilabbildung (**202** – **210**) als transient spezifiziert wird, wenn die Teilabbildung (**202** – **210**) kein persistentes Objekt (**203**) aufweist; und
der Übersetzer (**318**) konfiguriert ist, um die persistenten Teilabbildungen (**202** – **210**) zu erzeugen und innerhalb der Abbildungsdatenbank (**326**) kontinuierlich beizubehalten, wobei der Übersetzer (**318**) konfiguriert ist, um die transienten Teilabbildungen (**202** – **210**) auf ein Empfangen einer Benutzeraufforderung hin zu erzeugen und für eine Zeitperiode entsprechend der Benutzeraufforderung beizubehalten.

3. Das System (**100'**) gemäß Anspruch 1 oder 2, bei dem der Persistenzeingang durch eine Integrationsanwendung (**322**) erzeugt wird.

4. Das System (**100'**) gemäß Anspruch 1 oder 2, bei dem der Persistenzeingang durch den Benutzer erzeugt wird.

5. Das System (**100'**) gemäß Anspruch 2, bei dem der Persistenzeingang einen Verkäufer oder einen Gerätetyp definiert.

6. Das System (**100'**) gemäß Anspruch 1 oder 2, bei dem der Übersetzer konfiguriert ist, um eine Mehrzahl von hierarchisch angeordneten Teilabbildungen (**202** – **210**) aus den Topologiedaten zu erzeugen.

7. Das System (**100'**) gemäß Anspruch 1 oder 2, das ferner eine Integrationsanwendung (**322**) in Kommunikation mit dem Persistenzspezifikationsmechanismus (**103**) zum Erzeugen des Persistenzeingangs aufweist, und wobei:
der Persistenzspezifikationsmechanismus konfiguriert ist, um die Integrationsanwendung (**322**) über die persistenten Objekte (**203**) zu benachrichtigen; und
die Integrationsanwendung (**322**) konfiguriert ist, um ergänzende Informationen zu liefern oder der Schnittstelle (**322**) basierend auf den persistenten Objekten (**203**) Informationen anzuzeigen.

8. Das System (**100'**) gemäß Anspruch 6, bei dem die hierarchisch angeordneten Teilabbildungen (**202** – **210**) eine Internet-Teilabbildung (**204**), die zumindest ein Netzwerkobjekt (**203**) aufweist, zumindest eine Netzwerk-Teilabbildung (**206**), die dem zumindest einen Netzwerkobjekt (**203**) zugeordnet ist und zumindest ein Segmentobjekt (**203**) aufweist, zumindest eine Segment-Teilabbildung (**208**), die dem zumindest einen Segmentobjekt (**203**) zugeordnet ist und zumindest ein Knotenobjekt (**203**) aufweist, und zumindest eine Knoten-Teilabbildung (**210**) umfassen, die dem zumindest einem Knotenobjekt (**203**) zugeordnet ist und zumindest ein Schnittstellenobjekt (**203**) aufweist.

9. Ein Persistenzspezifikationsverfahren zum Ermöglichen, daß eine Integrationsanwendungssoftware Informationen zu einem Auf-Anforderung-Teilabbildung-Generator (**304**) liefert, das die folgenden Schritte aufweist:
Erzeugen von Teilabbildungen (**202** – **210**) mit dem Generator (**304**);
Übertragen von Konfigurationsdaten von einer Integrationsanwendung (**322**) zu dem Generator (**304**);
Spezifizieren von Objekten (**203**) in den Teilabbildungen (**202** – **210**) als persistent basierend auf den Konfigurationsdaten;
Spezifizieren einer jeglichen der Teilabbildungen (**202** – **210**), die ein persistentes Objekt (**203**) aufweist, als eine persistente Teilabbildung und Erzeugen und kontinuierliches Beibehalten einer jeglichen der persistenten Teilabbildungen (**202** – **210**) innerhalb des Generators (**304**);
Spezifizieren einer jeglichen der Teilabbildungen (**202** – **210**) als transient, wenn die Teilabbildung ohne ein persistentes Objekt (**203**) ist, und Erzeugen und Beibehalten einer jeglichen der transienten Teilabbildungen während einer Zeitperiode, die einer Anforderung durch einen Benutzer entspricht; und
Benachrichtigen der Integrationsanwendung (**322**) über die persistenten Objekte (**203**).

10. Das Verfahren gemäß Anspruch 9, das ferner folgende Schritte aufweist:

Erzeugen von zusätzlichen Anzeigeinformationen bei der Integrationsanwendung (**322**) basierend auf dem persistenten Objekt (**203**); und
Kommunizieren der zusätzlichen Anzeigeinformationen von der Integrationsanwendung (**322**) zu dem Generator (**304**) für eine Anzeige (**108**).

11. Das Verfahren gemäß Anspruch 9, das ferner folgende Schritte aufweist:
Empfangen einer Aufforderung von einem Benutzer bei dem Generator (**304**) nach einer Anzeige einer nicht-existierenden Teilabbildung (**202** – **210**);
Erzeugen einer transienten Teilabbildung (**202** – **210**) für den Benutzer nach der Aufforderung;
Beibehalten der transienten Teilabbildung (**202** – **210**) für eine Zeitperiode, die nach der Aufforderung beginnt, bis der Benutzer die Teilabbildung (**202** – **210**) schließt; und
Unterlassen, die transiente Teilabbildung (**202** – **210**) zu speichern, so daß die transiente Teilabbildung (**202** – **210**) beendet ist, wenn der Benutzer die Teilabbildung (**202** – **210**) schließt.

12. Das Verfahren gemäß Anspruch 9, bei dem die Konfigurationsdaten einen Gerätehersteller oder einen Gerätetyp oder eine Adresse spezifizieren.

Es folgen 18 Blatt Zeichnungen

Anhängende Zeichnungen

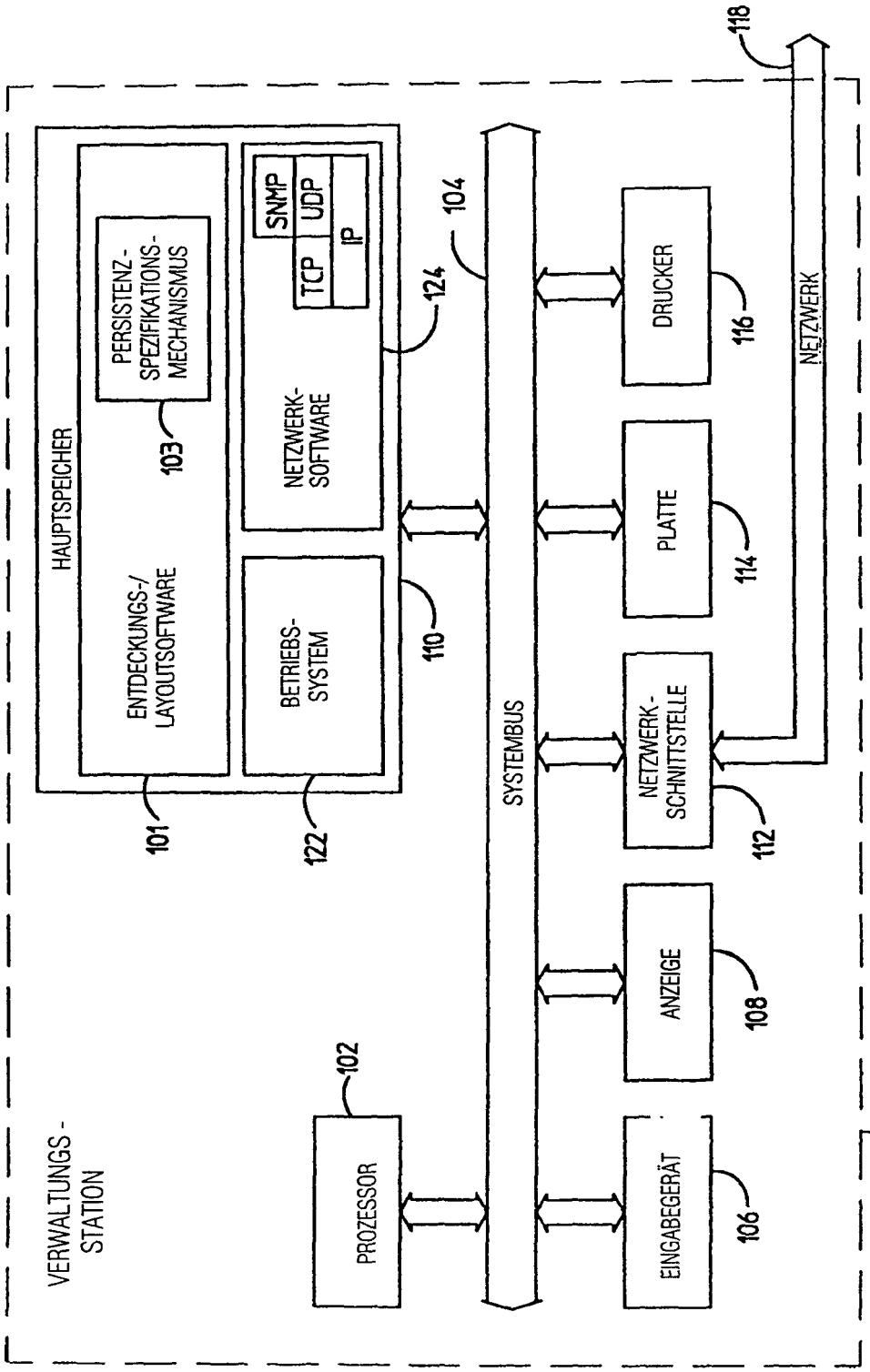

FIG 1

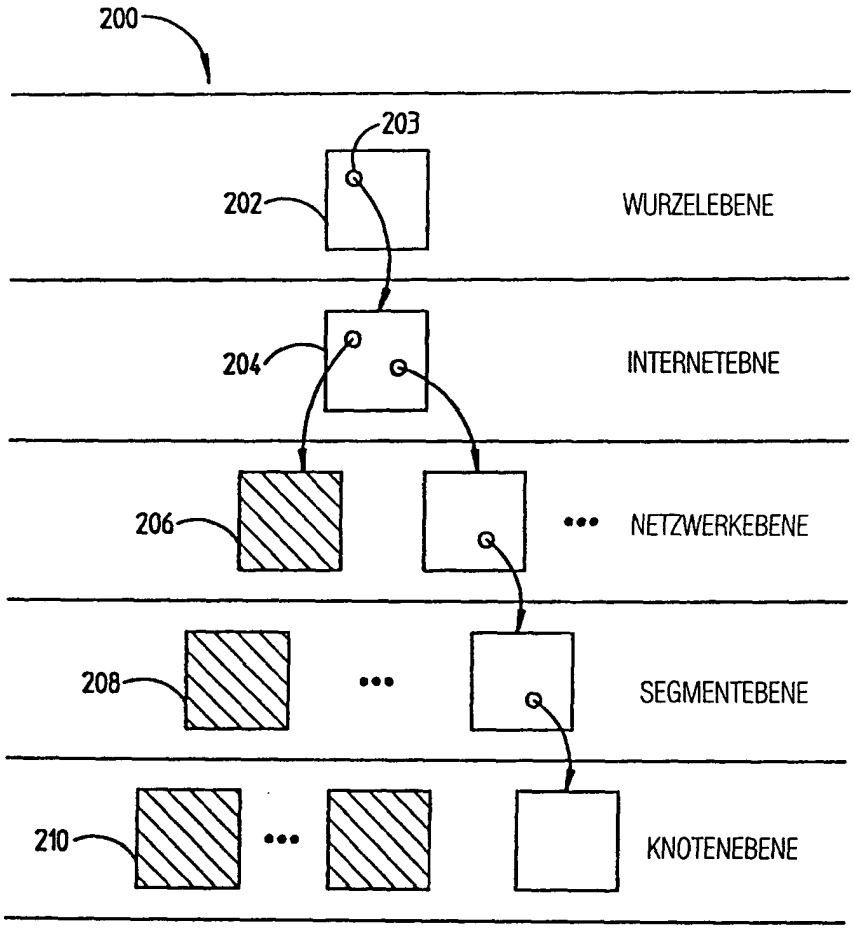
200
203
202
WURZELEBENE
204
INTERNETEBNE
206
NETZWERKEBENE
208
SEGMENTEBENE
210
KNOTENEBENE

FIG 2

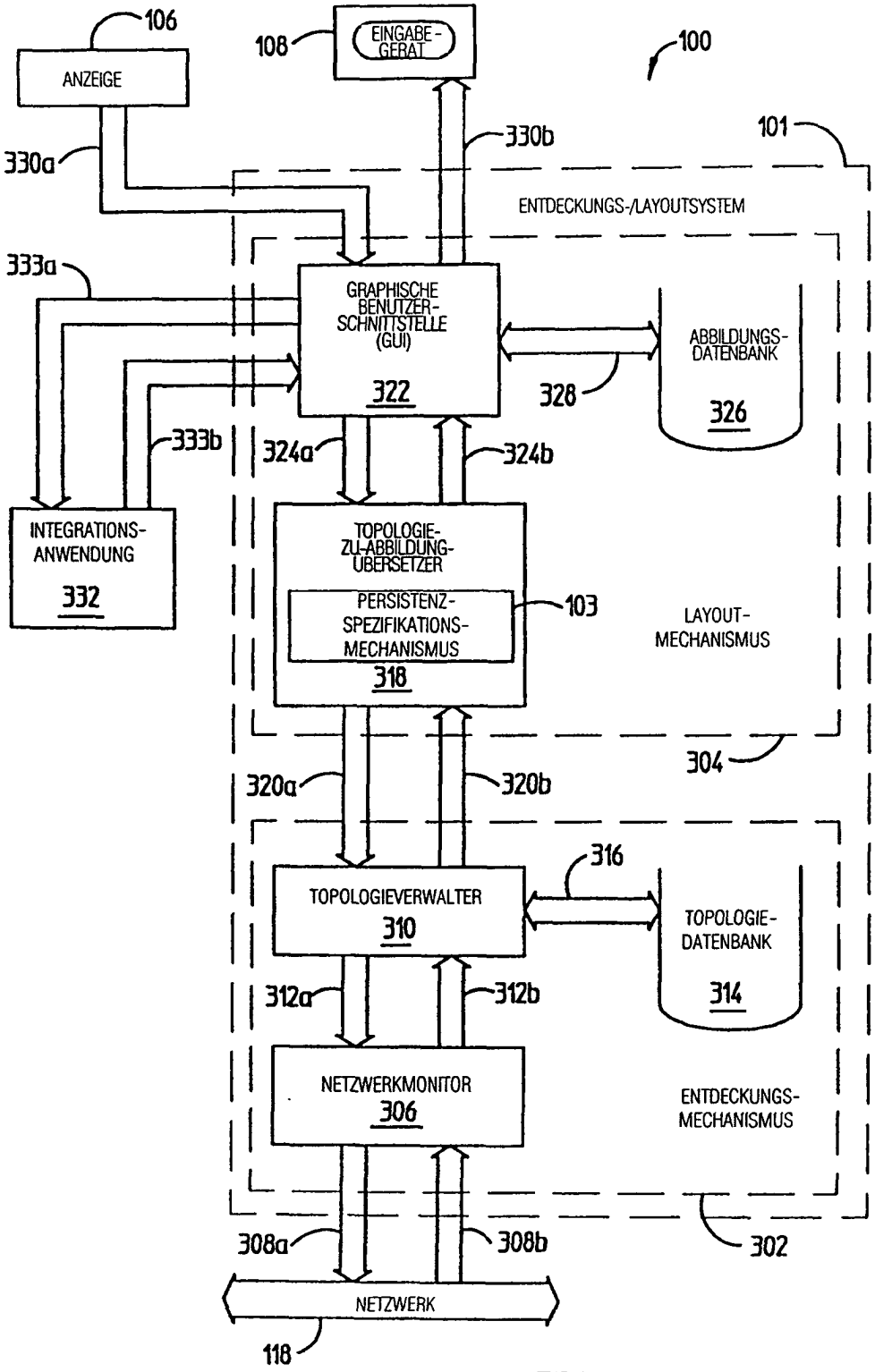
106
ANZEIGE
108
EINGABE-GERÄT
100
101
330a
330b
ENTDECKUNGS-/LAYOUTSYSTEM
333a
GRAPHISCHE BENUTZER-SCHNITTSTELLE (GUI)
322
ABBILDUNGS-DATENBANK
326
328
333b
324a
324b
INTEGRATIONS-ANWENDUNG
332
TOPOLOGIE-ZU-ABBILDUNG-ÜBERSETZER
PERSISTENZ-SPEZIFIKATIONS-MECHANISMUS
318
103
LAYOUT-MECHANISMUS
304
320a
320b
316
TOPOLOGIEVERWALTER
310
TOPOLOGIE-DATENBANK
314
312a
312b
NETZWERKMONITOR
306
ENTDECKUNGS-MECHANISMUS
308a
308b
302
NETZWERK
118

FIG 3

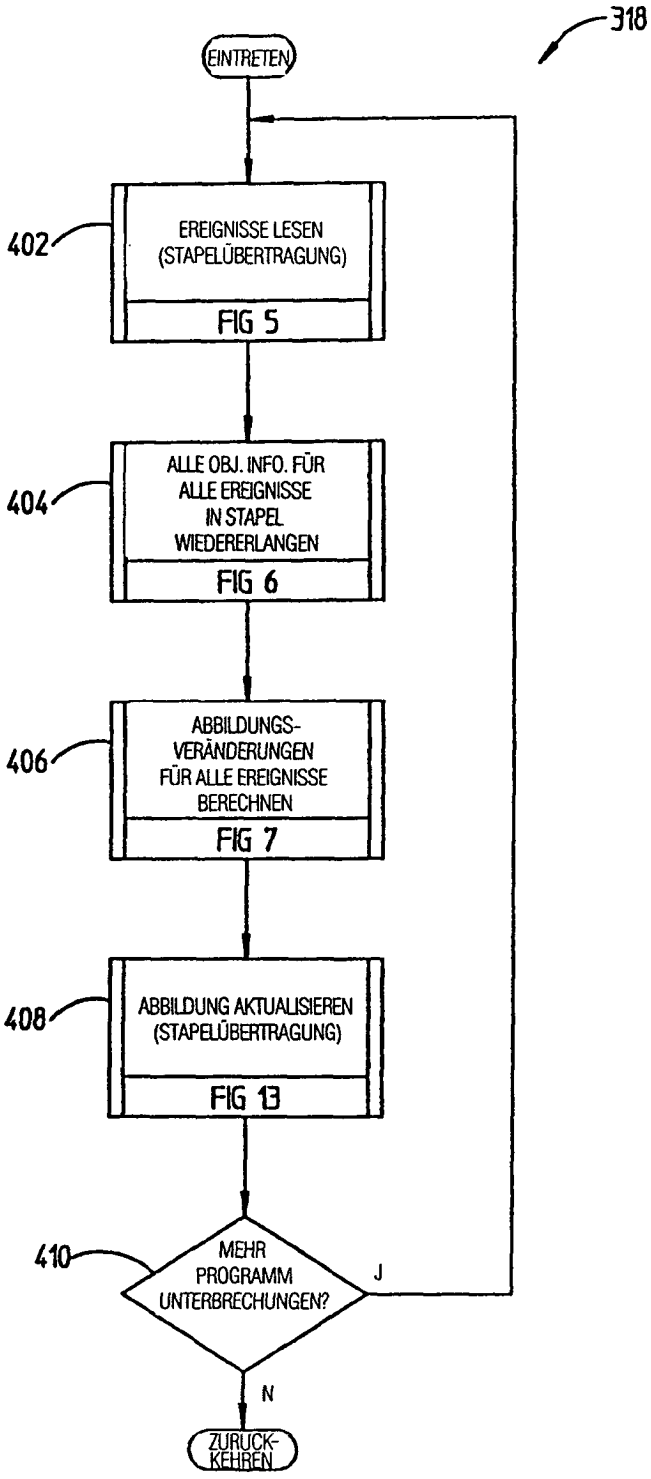
318
EINTRETEN
402
EREIGNISSE LESEN
(STAPELÜBERTRAGUNG)
FIG 5
404
ALLE OBJ. INFO. FÜR
ALLE EREIGNISSE
IN STAPEL
WIEDERERLANGEN
FIG 6
406
ABBILDUNGS-
VERÄNDERUNGEN
FÜR ALLE EREIGNISSE
BERECHNEN
FIG 7
408
ABBILDUNG AKTUALISIEREN
(STAPELÜBERTRAGUNG)
FIG 13
410
MEHR
PROGRAMM
UNTERBRECHUNGEN?
J
N
ZURÜCK-
KEHREN

FIG 4

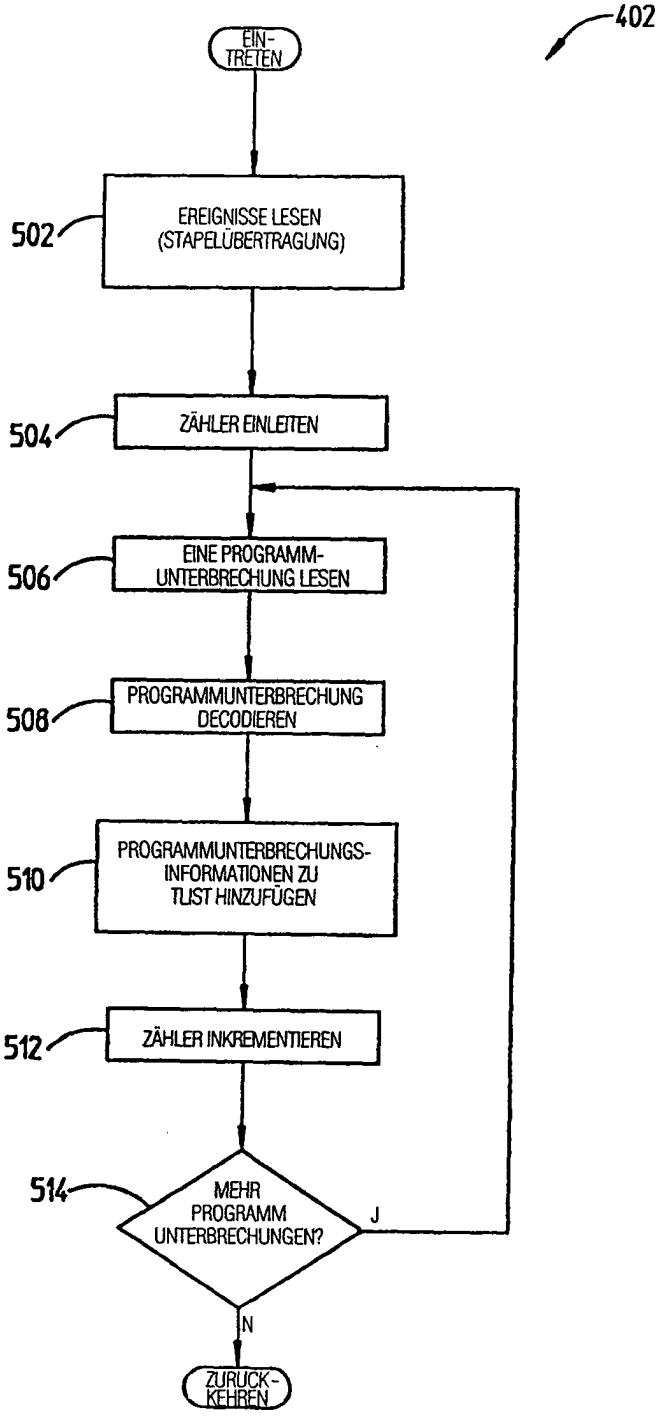
402
EIN-TRETEN
502
EREIGNISSE LESEN (STAPELÜBERTRAGUNG)
504
ZÄHLER EINLEITEN
506
EINE PROGRAMM-UNTERBRECHUNG LESEN
508
PROGRAMMUNTERBRECHUNG DECODIEREN
510
PROGRAMMUNTERBRECHUNGS-INFORMATIONEN ZU TLIST HINZUFÜGEN
512
ZÄHLER INKREMENTIEREN
514
MEHR PROGRAMM UNTERBRECHUNGEN?
J
N
ZURÜCK-KEHREN

FIG 5

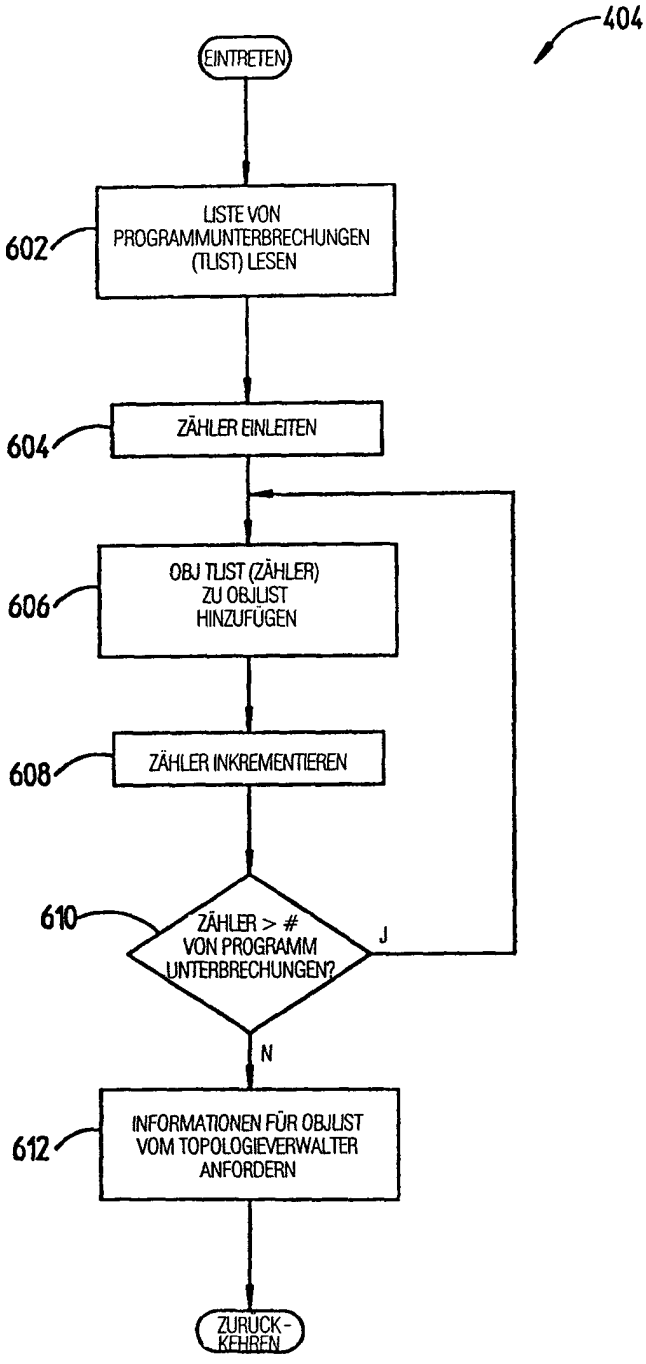
404
EINTRETEN
602
LISTE VON PROGRAMMUNTERBRECHUNGEN (TLIST) LESEN
604
ZÄHLER EINLEITEN
606
OBJ TLIST (ZÄHLER) ZU OBJLIST HINZUFÜGEN
608
ZÄHLER INKREMENTIEREN
610
ZÄHLER > # VON PROGRAMM UNTERBRECHUNGEN?
J
N
612
INFORMATIONEN FÜR OBJLIST VOM TOPOLOGIEVERWALTER ANFORDERN
ZURÜCK-KEHREN

FIG 6

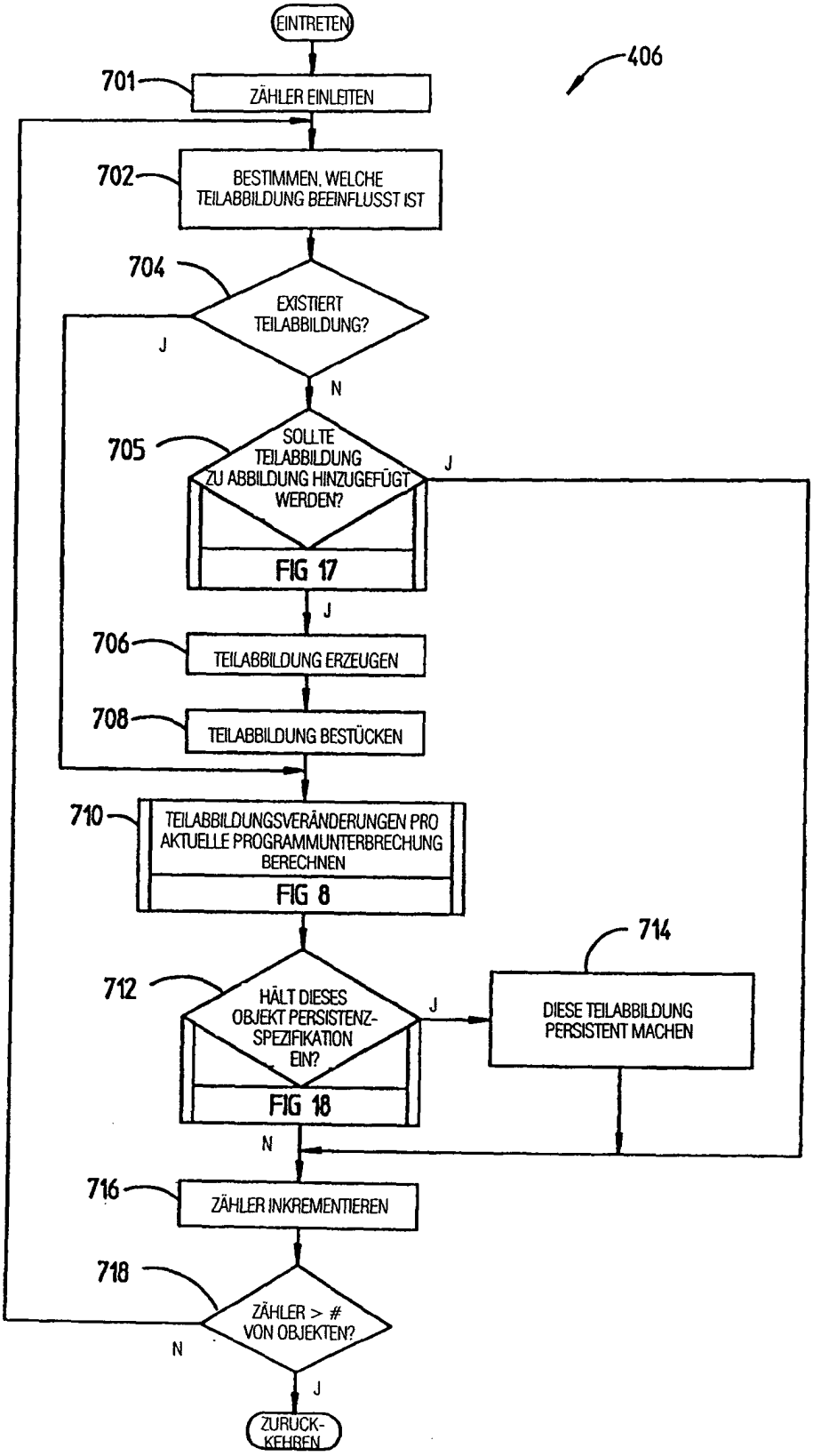
EINTRETEN
406
701
ZÄHLER EINLEITEN
702
BESTIMMEN, WELCHE TEILABBILDUNG BEEINFLUSST IST
704
EXISTIERT TEILABBILDUNG?
J
N
705
SOLLTE TEILABBILDUNG ZU ABBILDUNG HINZUGEFÜGT WERDEN?
J
FIG 17
J
706
TEILABBILDUNG ERZEUGEN
708
TEILABBILDUNG BESTÜCKEN
710
TEILABBILDUNGSVERÄNDERUNGEN PRO AKTUELLE PROGRAMMUNTERBRECHUNG BERECHNEN
FIG 8
714
712
HÄLT DIESES OBJEKT PERSISTENZ-SPEZIFIKATION EIN?
J
DIESE TEILABBILDUNG PERSISTENT MACHEN
FIG 18
N
716
ZÄHLER INKREMENTIEREN
718
ZÄHLER > # VON OBJEKTEN?
N
J
ZURÜCK-KEHREN

FIG 7

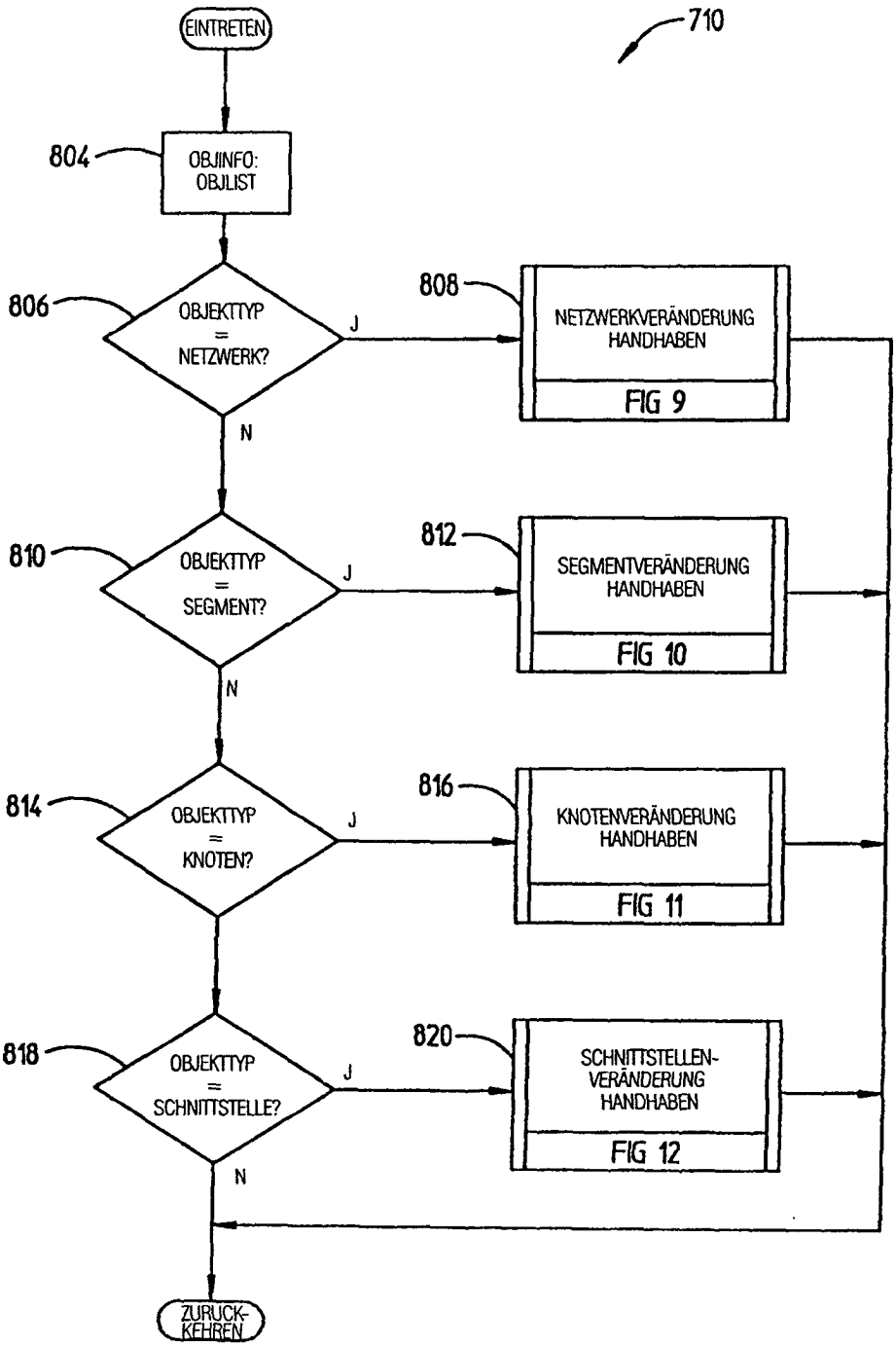
710
EINTRETEN
804
OBJINFO:
OBJLIST
806
OBJEKTTYP
=
NETZWERK?
J
N
808
NETZWERKVERÄNDERUNG
HANDHABEN
FIG 9
810
OBJEKTTYP
=
SEGMENT?
J
N
812
SEGMENTVERÄNDERUNG
HANDHABEN
FIG 10
814
OBJEKTTYP
=
KNOTEN?
J
816
KNOTENVERÄNDERUNG
HANDHABEN
FIG 11
818
OBJEKTTYP
=
SCHNITTSTELLE?
J
N
820
SCHNITTSTELLEN-
VERÄNDERUNG
HANDHABEN
FIG 12
ZURÜCK-
KEHREN

FIG 8

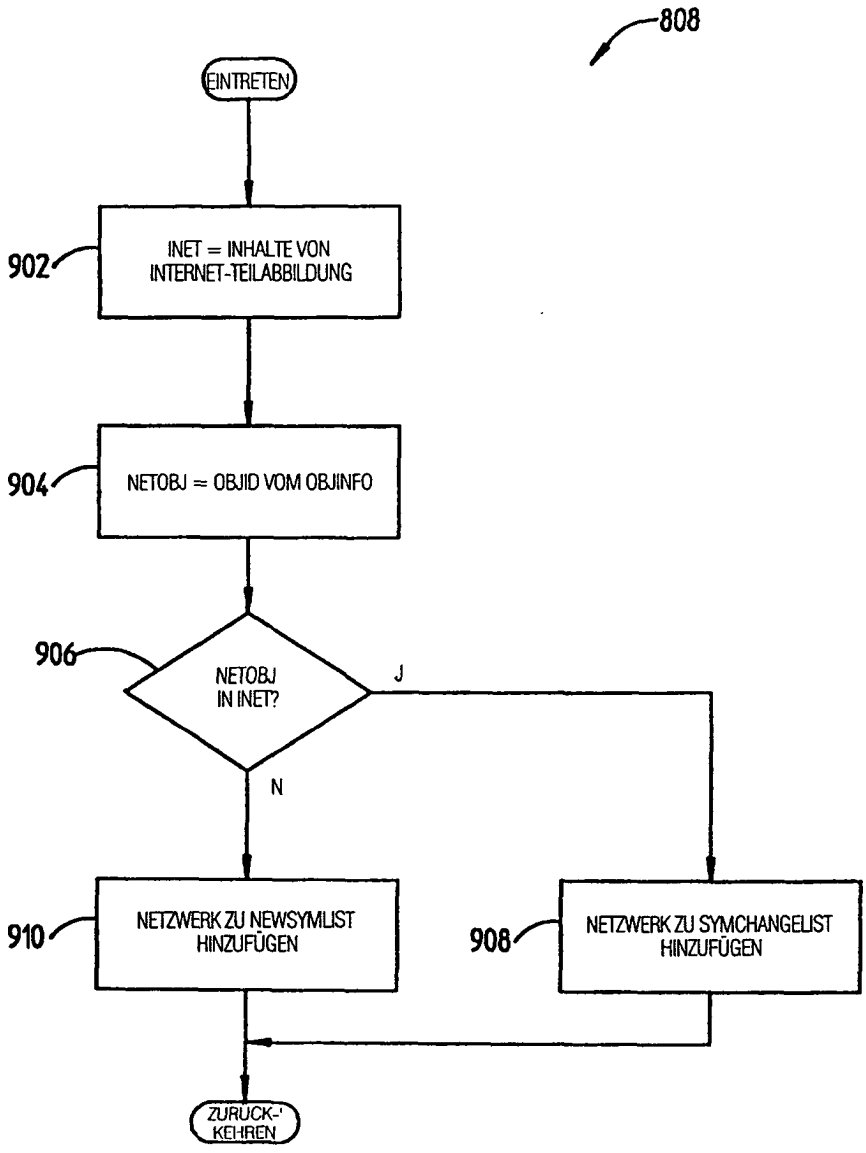
808
EINTRETEN
902
INET = INHALTE VON INTERNET-TEILABBILDUNG
904
NETOBJ = OBJID VOM OBJINFO
906
NETOBJ IN INET?
J
N
910
NETZWERK ZU NEWSYMLIST HINZUFÜGEN
908
NETZWERK ZU SYMCHANGELIST HINZUFÜGEN
ZURÜCK-KEHREN

FIG 9

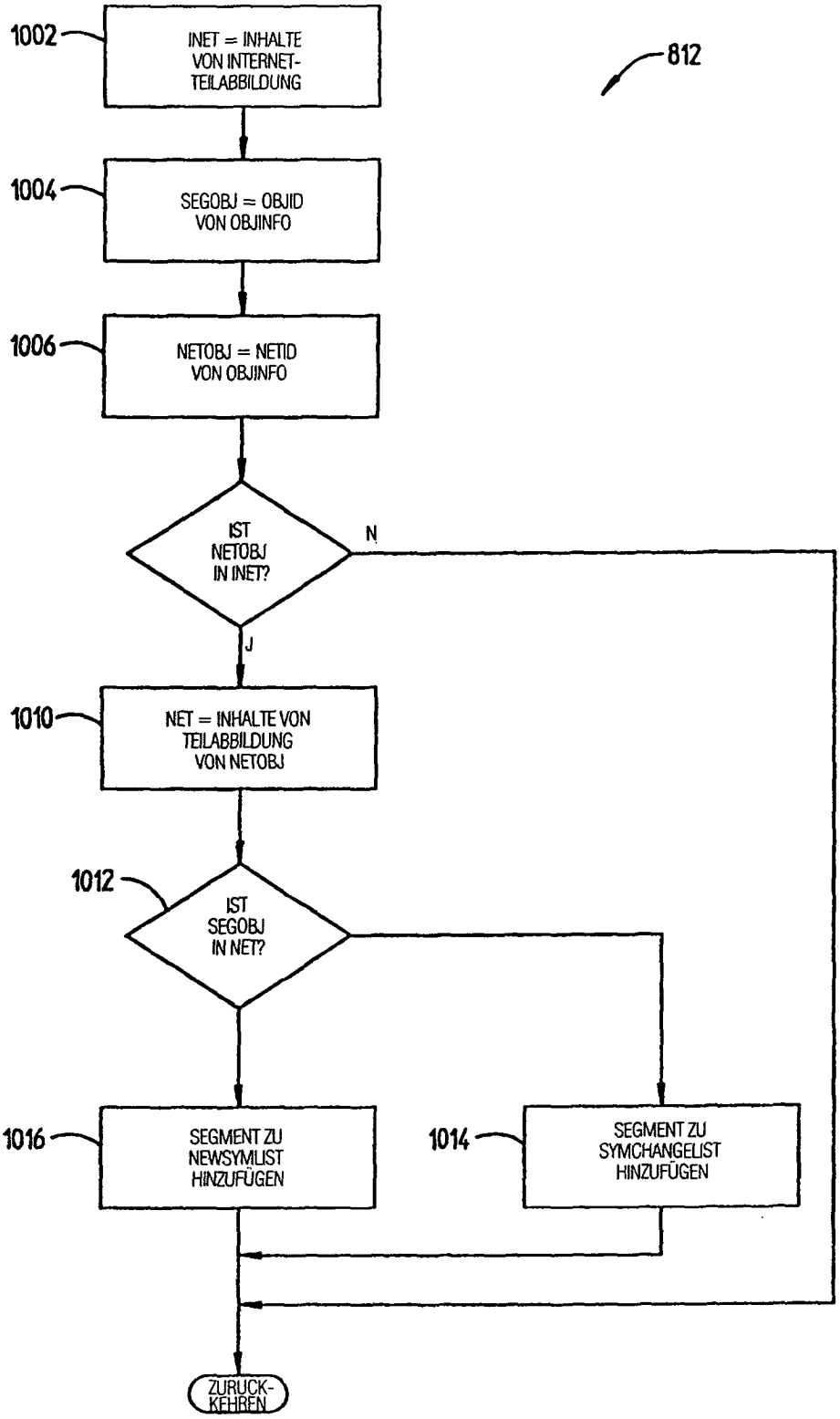
1002
INET = INHALTE VON INTERNET-TEILABBILDUNG
812
1004
SEGOBJ = OBJID VON OBJINFO
1006
NETOBJ = NETID VON OBJINFO
IST NETOBJ IN INET?
N
J
1010
NET = INHALTE VON TEILABBILDUNG VON NETOBJ
1012
IST SEGOBJ IN NET?
1016
SEGMENT ZU NEWSYMLIST HINZUFÜGEN
1014
SEGMENT ZU SYMCHANGELIST HINZUFÜGEN
ZURÜCK-KEHREN

FIG 10

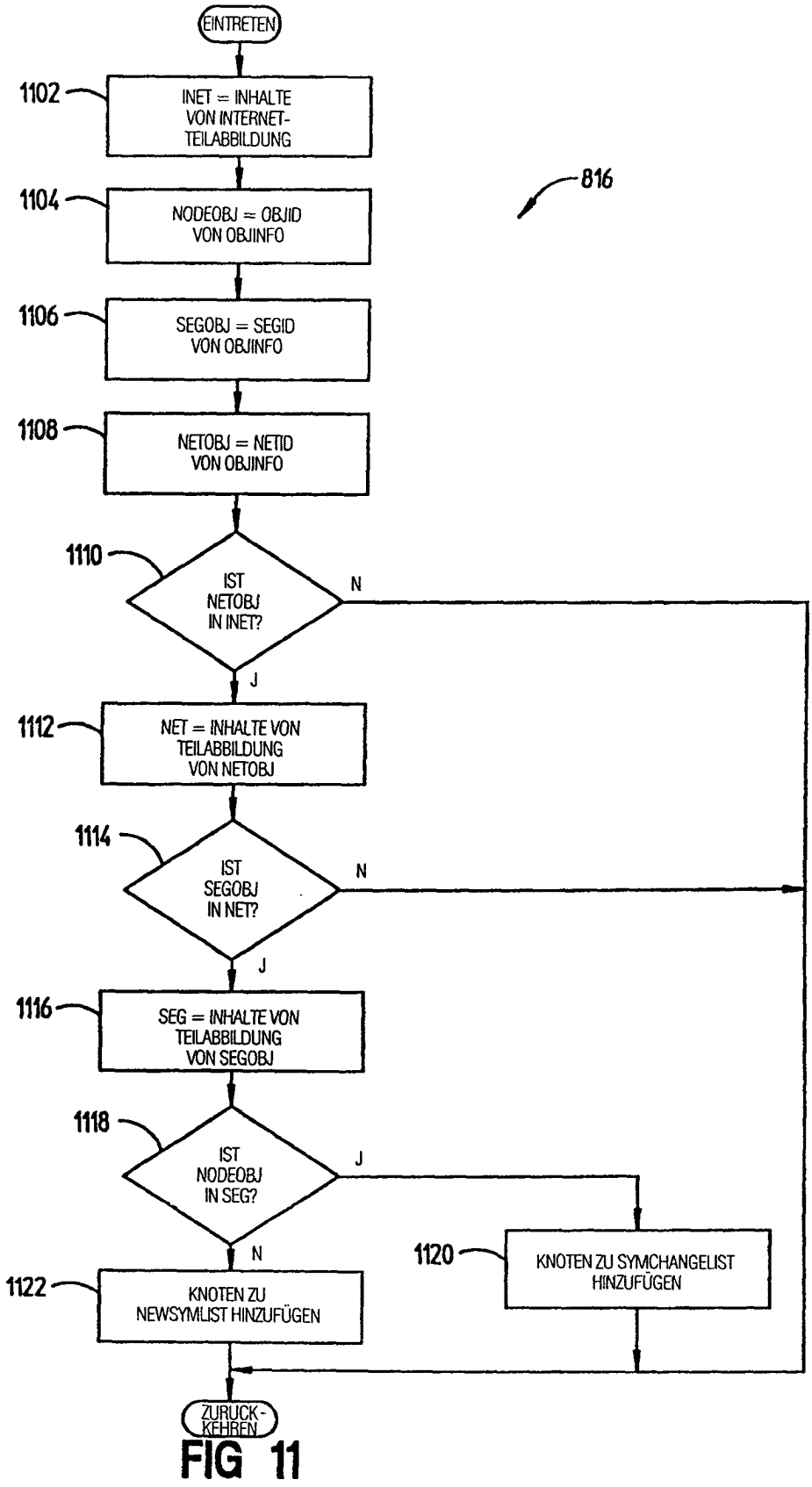
EINTRETEN
1102
INET = INHALTE VON INTERNET-TEILABBILDUNG
1104
NODEOBJ = OBJID VON OBJINFO
816
1106
SEGOBJ = SEGID VON OBJINFO
1108
NETOBJ = NETID VON OBJINFO
1110
IST NETOBJ IN INET?
N
J
1112
NET = INHALTE VON TEILABBILDUNG VON NETOBJ
1114
IST SEGOBJ IN NET?
N
J
1116
SEG = INHALTE VON TEILABBILDUNG VON SEGOBJ
1118
IST NODEOBJ IN SEG?
J
N
1120
KNOTEN ZU SYMCHANGELIST HINZUFÜGEN
1122
KNOTEN ZU NEWSYMLIST HINZUFÜGEN
ZURÜCK-KEHREN

FIG 11

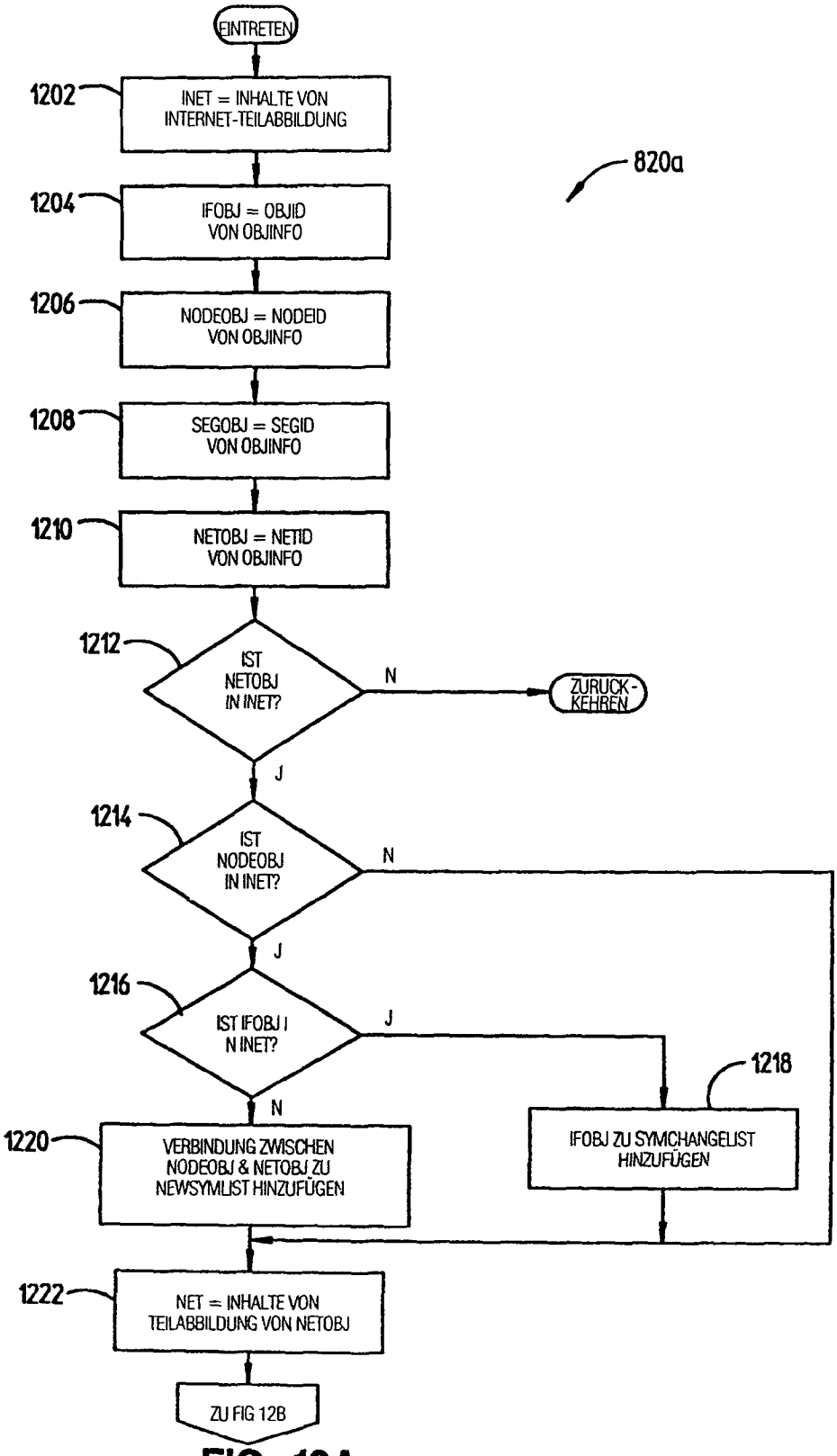
EINTRETEN
1202
INET = INHALTE VON INTERNET-TEILABBILDUNG
820a
1204
IFOBJ = OBJID VON OBJINFO
1206
NODEOBJ = NODEID VON OBJINFO
1208
SEGOBJ = SEGID VON OBJINFO
1210
NETOBJ = NETID VON OBJINFO
1212
IST NETOBJ IN INET?
N
ZURÜCK-KEHREN
J
1214
IST NODEOBJ IN INET?
N
J
1216
IST IFOBJ I N INET?
J
1218
N
1220
VERBINDUNG ZWISCHEN NODEOBJ & NETOBJ ZU NEWSYMLIST HINZUFÜGEN
IFOBJ ZU SYMCHANGELIST HINZUFÜGEN
1222
NET = INHALTE VON TEILABBILDUNG VON NETOBJ
ZU FIG 12B

FIG 12A

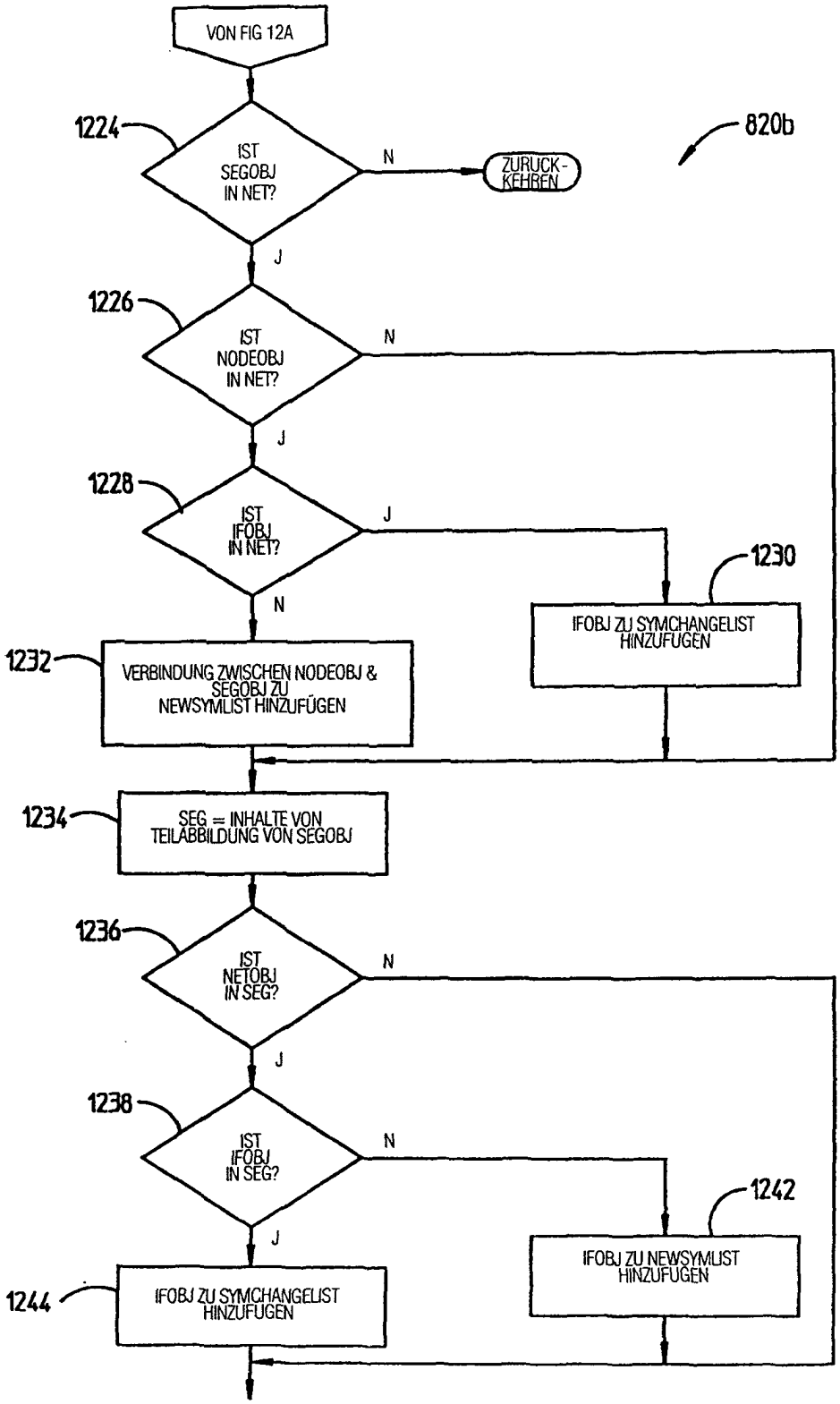
VON FIG 12A
820b
1224
IST SEGOBJ IN NET?
N
ZURUCK-KEHREN
J
1226
IST NODEOBJ IN NET?
N
J
1228
IST IFOBJ IN NET?
J
1230
IFOBJ ZU SYMCHANGELIST HINZUFUGEN
N
1232
VERBINDUNG ZWISCHEN NODEOBJ & SEGOBJ ZU NEWSYMLIST HINZUFÜGEN
1234
SEG = INHALTE VON TEILABBILDUNG VON SEGOBJ
1236
IST NETOBJ IN SEG?
N
J
1238
IST IFOBJ IN SEG?
N
1242
IFOBJ ZU NEWSYMLIST HINZUFUGEN
J
1244
IFOBJ ZU SYMCHANGELIST HINZUFUGEN

FIG 12B

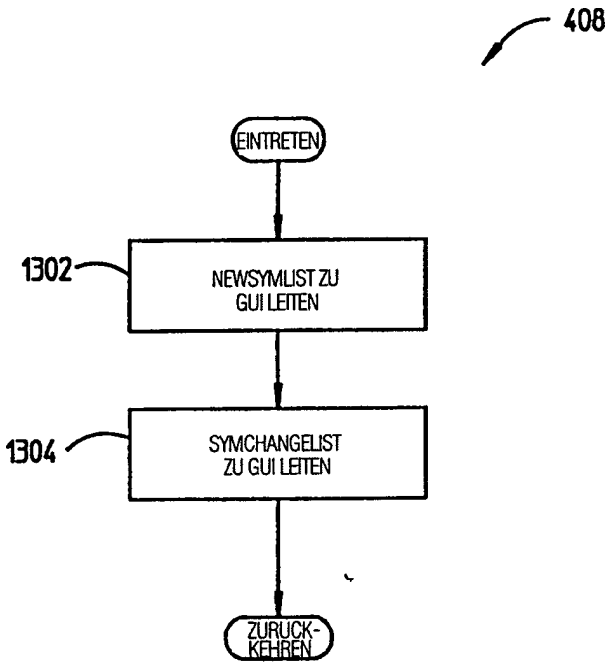
408
EINTRETEN
1302
NEWSYMLIST ZU GUI LEITEN
1304
SYMCHANGELIST ZU GUI LEITEN
ZURUCK-KEHREN

FIG 13

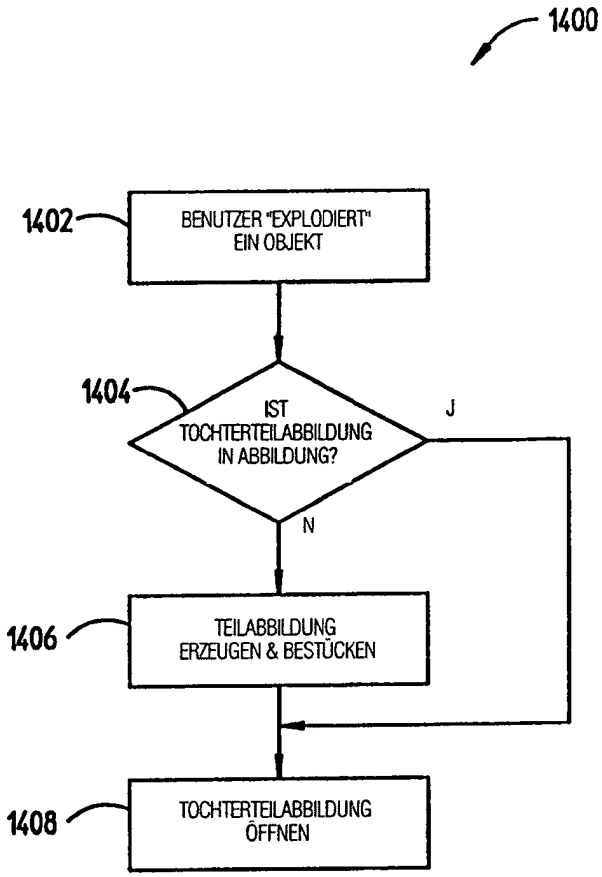
1400
1402
BENUTZER "EXPLODIERT" EIN OBJEKT
1404
IST TOCHTERTEILABBILDUNG IN ABBILDUNG?
J
N
1406
TEILABBILDUNG ERZEUGEN & BESTÜCKEN
1408
TOCHTERTEILABBILDUNG ÖFFNEN

FIG 14

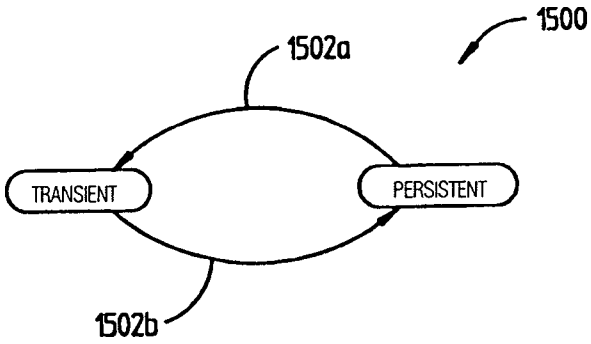

FIG 15

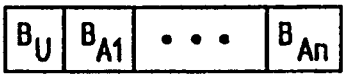

FIG 16

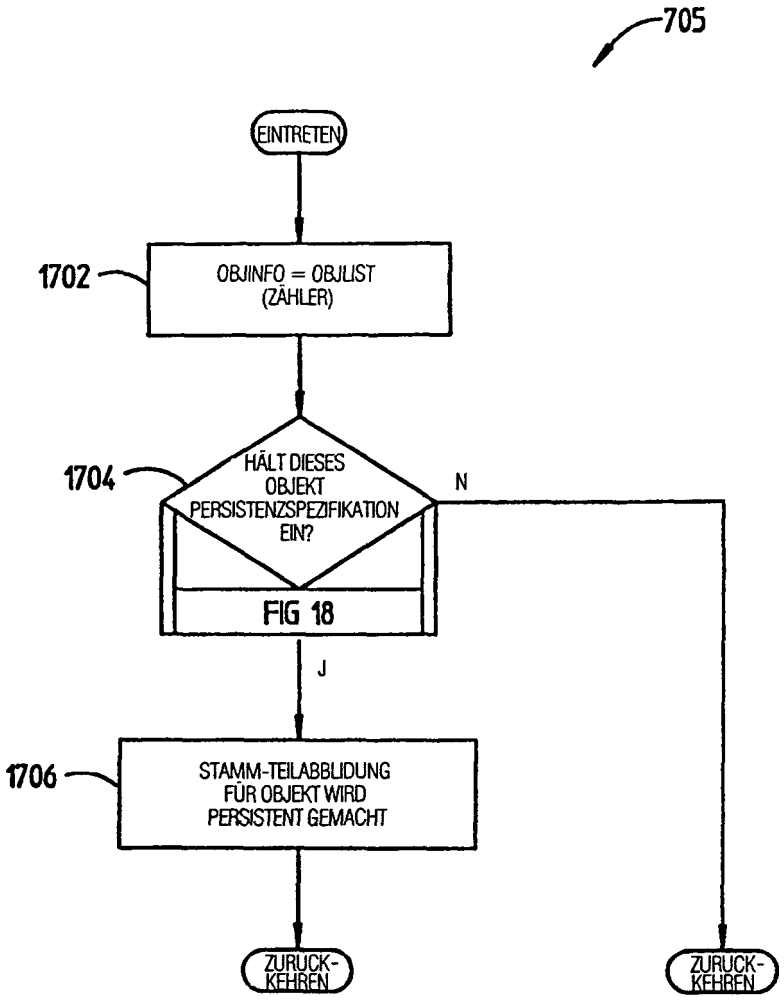
705
EINTRETEN
1702
OBJINFO = OBJLIST (ZÄHLER)
1704
HÄLT DIESES OBJEKT PERSISTENZSPEZIFIKATION EIN?
N
FIG 18
J
1706
STAMM-TEILABBILDUNG FÜR OBJEKT WIRD PERSISTENT GEMACHT
ZURÜCK-KEHREN
ZURÜCK-KEHREN

FIG 17

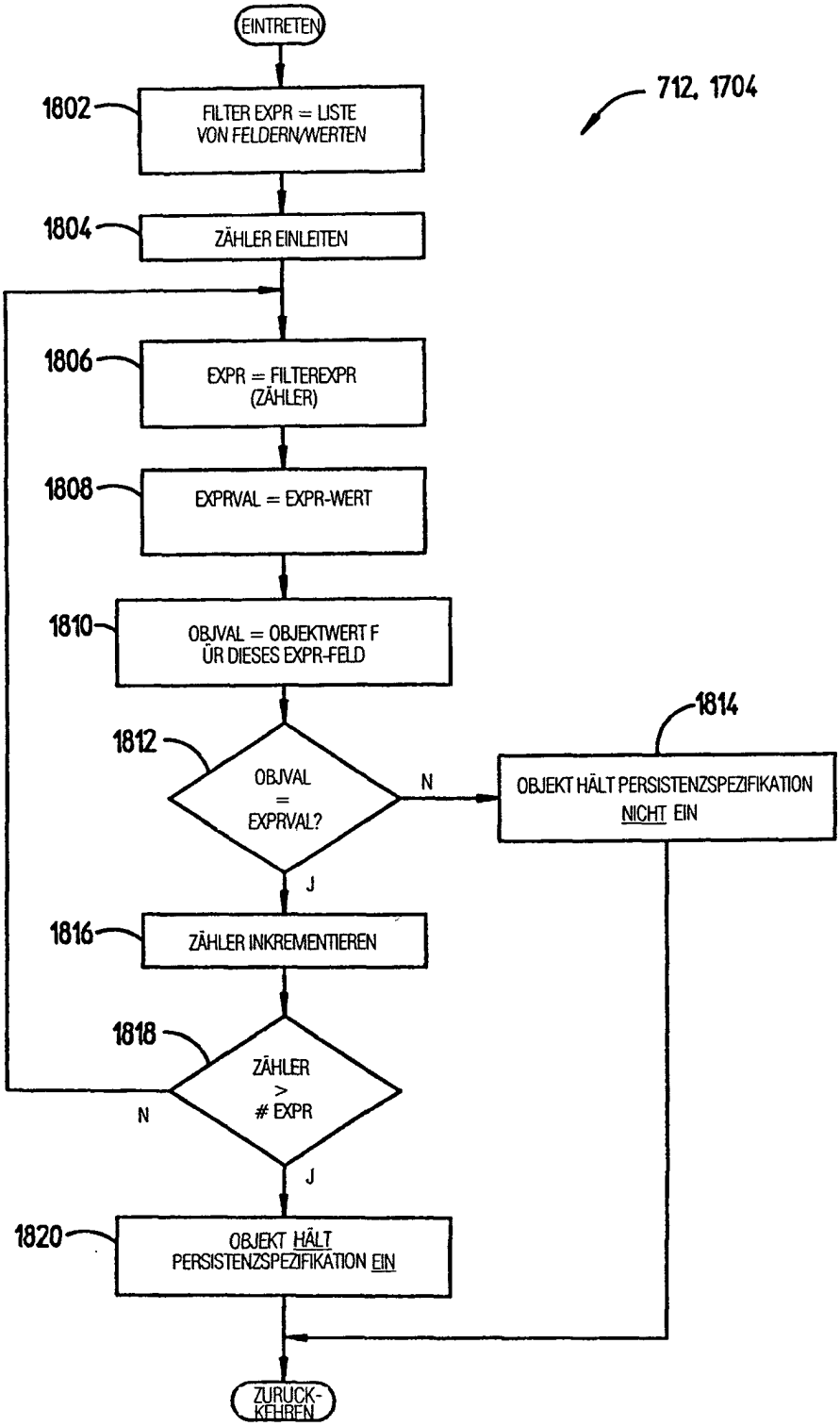
EINTRETEN
712, 1704
1802
FILTER EXPR = LISTE VON FELDERN/WERTEN
1804
ZÄHLER EINLEITEN
1806
EXPR = FILTEREXPR (ZÄHLER)
1808
EXPRVAL = EXPR-WERT
1810
OBJVAL = OBJEKTWERT FÜR DIESES EXPR-FELD
1812
OBJVAL = EXPRVAL?
N
1814
OBJEKT HÄLT PERSISTENZSPEZIFIKATION NICHT EIN
J
1816
ZÄHLER INKREMENTIEREN
1818
ZÄHLER > # EXPR
N
J
1820
OBJEKT HÄLT PERSISTENZSPEZIFIKATION EIN
ZURÜCK-KEHREN

FIG 18