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

(10) **DE 600 31 664 T2** 2007.08.30

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 175 661 B1**
(21) Deutsches Aktenzeichen: **600 31 664.5**
(86) PCT-Aktenzeichen: **PCT/US00/10752**
(96) Europäisches Aktenzeichen: **00 926 224.7**
(87) PCT-Veröffentlichungs-Nr.: **WO 2000/063841**
(86) PCT-Anmeldetag: **20.04.2000**
(87) Veröffentlichungstag der PCT-Anmeldung: **26.10.2000**
(97) Erstveröffentlichung durch das EPA: **30.01.2002**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **02.11.2006**
(47) Veröffentlichungstag im Patentblatt: **30.08.2007**

(51) Int Cl.[8]: ***G06T 11/00*** (2006.01)

(30) Unionspriorität:
130234 P **21.04.1999** **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:
SPSS, Inc., Chicago, Ill., US

(72) Erfinder:
WILKINSON, Leland, Chicago, IL 60622, US

(74) Vertreter:
derzeit kein Vertreter bestellt

(54) Bezeichnung: **COMPUTERVERFAHREN UND VORRICHTUNG ZUM SCHAFFEN VON SICHTBARER GRAPHIK UNTER VERWENDUNG VON GRAPH ALGEBRA**

Beschreibung

GEBIET DER ERFINDUNG

[0001] Diese Erfindung betrifft allgemein ein Computerverfahren und eine Vorrichtung zum mathematischen Konstruieren von Graphen und zum ästhetischen Repräsentieren von Graphen als Graphik und insbesondere ein Computerverfahren und eine Vorrichtung zum Konstruieren von Graphen unter Verwendung einer Graphalgebra, gefolgt vom visuellen oder andersartigen Repräsentieren der Graphen als eine quantitative, ästhetische graphische Repräsentation.

HINTERGRUND DER ERFINDUNG

[0002] Bis vor kurzem wurde Graphik typischerweise mit der Hand gezeichnet, um mathematische, statistische und geometrische Relationen zu repräsentieren. Computergraphikprogramme, besonders naturwissenschaftliche und mathematische Plotpakete, haben diese Aufgabe sehr erleichtert, ohne aber ihren Ad-hoc-Aspekt zu verändern. Ebenso wenig haben statistische oder mathematische Pakete, die komplexere Graphik generieren, zu unserem Verständnis beigetragen, wie sie erzeugt werden.

[0003] Es besteht ein Bedarf an einem Computerverfahren und System, die die Fähigkeit verleihen, Graphik systematisch zu konstruieren, um komplexere multivariate Umgebungen zu handhaben. Dies ist auch in Verbindung mit Data-Mining-Computersystemen der Fall. Leider übertrifft die Leistungsstärke von Data-Mining-Systemen bei weitem die in ihren Anzeigen benutzten Computergraphikverfahren. Die meisten Data-Mining-Computersysteme stützen sich noch immer auf Torten-, Linien- und Balkendiagramme der Schichten von Datenkuben (Vielweg-Aggregationen einer Teilmenge einer Datenbank). Diesen Diagrammen gelingt es nicht, die Relationen zwischen den Entitäten aufzuzeigen, die sie repräsentieren, weil sie über keine tiefe Grammatik für ihre Generierung verfügen. Sie sind ganz einfach an Facetten des Datenkubus festverdrahtet. Bohrt man beispielsweise durch den Kubus, um eine andere Datenschicht zu betrachten, dann erhält man nur ein einfaches Tortendiagramm. Eine ähnliche Festverdrahtung gibt es in Anzeigen von Baumklassifikatoren, Neuronennetzen und anderen Algorithmen.

[0004] Es besteht auch Bedarf an einem Verfahren und einer Vorrichtung zum Erzeugen von ästhetischer Graphik aus Daten unter Verwendung von Graphalgebra. Ein Beispiel für eine Graphalgebra findet sich in dem Papier „A Graph Algebra" von L. Wilkinson, Proceedings of the Symposium on the Interface, S. 341–351.

ZUSAMMENFASSUNG DER ERFINDUNG

[0005] Gemäß der vorliegenden Erfindung wird ein Computerverfahren zum Erzeugen von quantitativer Graphik unter Verwendung von Graphalgebra bereitgestellt.

[0006] Gemäß einem anderen Aspekt der vorliegenden Erfindung wird ein Datenverarbeitungssystem bereitgestellt, um Graphen mathematisch unter Verwendung von Graphalgebra zu konstruieren und die Graphen ästhetisch als Graphik anzuzeigen, die eine visuelle oder andere sensorische Anzeige des zugrunde liegenden Graphen sein kann.

[0007] Gemäß noch einem anderen Aspekt der vorliegenden Erfindung wird ein nichtflüchtiges Speichermedium mit in einer maschinenlesbaren Form codierten Computersoftware bereitgestellt, um Graphen mathematisch unter Verwendung einer Graphalgebra zu erzeugen und den mathematischen Graphen ästhetisch anzuzeigen, beispielsweise durch eine visuelle Repräsentation.

[0008] Gemäß der Erfindung enthält ein Verfahren zum Erzeugen von quantitativer Graphik die Ausführung der folgenden Schritte auf einem Computer: Indexieren der Daten, um eine Datenmenge zu bilden; Umwandeln der Datenmenge in eine variable Datenstruktur, wobei die variable Datenstruktur eine Indexmenge ist; Umwandeln der variablen Datenstruktur in eine Menge von Variablen durch Verwendung von mindestens einer Operation aus der aus einem Blend-Operator, einem Cross-Operator und einem Nest-Operator bestehenden Gruppe; Abbilden der Menge der Variablen in eine Punktmenge; und Abbilden der Punktmenge in eine ästhetische Repräsentation, die eine visuelle Graphik sein kann.

[0009] Gemäß einem anderen Aspekt der vorliegenden Erfindung wird ein Verfahren zum Erzeugen von quantitativer Graphik bereitgestellt, das folgende Schritte enthält: Bereitstellen einer Variablenliste; Bereitstellen einer Liste von Punktrepräsentationen; Bereitstellen einer Liste von Koordinatensystemen; Bereitstellen ei-

ner Liste von ästhetischen Repräsentationen; Auswählen von mindestens einer Variablen aus einer Variablenliste; Auswählen von mindestens einer Punktrepräsentation aus einer Liste von Punktrepräsentationen; Auswählen von mindestens einem Koordinatensystem aus der Liste von Koordinatensystemen; Auswählen von mindestens einer ästhetischen Repräsentation aus der Liste von ästhetischen Repräsentationen; Bewegen der mindestens einen Variablen an eine vorbestimmte Position; und Anzeigen einer sichtbaren Graphik, die die mindestens eine Variable, die mindestens eine Punktrepräsentation, das mindestens eine Koordinatensystem und die mindestens eine ästhetische Repräsentation reflektiert.

[0010] Gemäß einem anderen erfindungsgemäßen Aspekt enthält das Datenverarbeitungssystem zum Konstruieren von Graphen, das die Graphen mathematisch und ästhetisch als Graphik repräsentiert: einen Computerprozessor; und einen ansprechbar an den Computerprozessor gekoppelten Speicher, der einen Satz von Computerbefehlen enthält zum: (a) Indexieren von Daten, um eine Datenmenge zu bilden, (b) Umwandeln der Datenmenge in eine variable Datenstruktur, wobei die variable Datenstruktur eine Indexmenge, einen Bereich und eine Funktion hat; (c) Umwandeln der variablen Datenstruktur in eine Menge von Variablen durch Verwendung von mindestens einem aus der folgenden Gruppe ausgewählten Operator: einem Blend-Operator, einem Cross-Operator und einem Nest-Operator; (d) Abbilden der Menge der Variablen in eine Menge mathematischer Punkte; und (e) Abbilden der Menge mathematischer Punkte in eine ästhetische Repräsentation.

[0011] Gemäß noch einem anderen Aspekt der vorliegenden Erfindung wird ein nichtflüchtiges Speichermedium bereitgestellt, das in einem maschinenlesbaren Format codierte Computersoftware zum Erzeugen von quantitativer Graphik enthält. Das nichtflüchtige Speichermedium enthält: einen Satz von Computerbefehlen zum Indexieren von Daten, um eine Datenmenge zu bilden; einen Satz von Computerbefehlen zum Umwandeln der Datenmenge in eine variable Datenstruktur, wobei die variable Datenstruktur eine Indexmenge, einen Bereich und eine Funktion hat; einen Satz von Computerbefehlen zum Umwandeln der variablen Datenstruktur in eine Menge von Variablen durch Verwendung mindestens eines Operators, der aus der die Folgenden umfassenden Gruppe ausgewählt ist: einen Blend-Operator, einen Cross-Operator und einen Nest-Operator; einen Satz von Computerbefehlen zum Abbilden der Menge der Variablen in eine Punktmenge; und einen Satz von Computerbefehlen zum Abbilden der Punktmenge in eine ästhetische Repräsentation.

[0012] Die erfindungsgemäßen Verfahren, Systeme und Einrichtungen ermöglichen die Manipulation von Daten auf viele verschiedene Weisen und auch ihre Repräsentation durch Graphik auf viele verschiedene Weisen. So ist die Erzeugung von Graphik gemäß der vorliegenden Erfindung nicht durch die Beschränkungen von graphischen Repräsentationen begrenzt, die nur Aggregationen einer Teilmenge einer Datenbank sind.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0013] Die vorliegende Erfindung lässt sich leichter mit Bezug auf die beigefügten Zeichnungen verstehen, von denen:

[0014] Abb. 1 ein Flussdiagramm ist, das das Verfahren zum Formen von Graphik aus Daten gemäß der vorliegenden Erfindung schematisch veranschaulicht;

[0015] Abb. 2 ein Objektdiagramm ist, das die primären Softwarekomponenten des Datenverarbeitungssystems und ihre Relation zueinander zeigt;

[0016] Abb. 3 ein Objektdiagramm des Datenansichtspakets der vorliegenden Erfindung ist;

[0017] Abb. 4 ein Objektdiagramm des Serveraspekts der vorliegenden Erfindung ist;

[0018] Abb. 5 ein Objektdiagramm des Rahmenmodells der vorliegenden Erfindung ist;

[0019] Abb. 6 ein Diagramm des Algebrapakets der vorliegenden Erfindung ist;

[0020] Abb. 7 ein Objektdiagramm eines einfachen algebraischen Ausdrucks ist;

[0021] Abb. 8 ein Objektdiagramm eines algebraischen Ausdrucks ist;

[0022] Abb. 9 ein Objektdiagramm eines algebraischen Ausdrucks ist;

[0023] Abb. 10 ein Objektdiagramm des Dimensionsaspekts der vorliegenden Erfindung ist;

[0024] Abb. 11 ein Objektdiagramm ist, das den Rahmenaspekt der vorliegenden Erfindung zeigt;

[0025] Abb. 12 ein Objektdiagramm des Anzeigers der vorliegenden Erfindung ist;

[0026] Abb. 13 ein Objektdiagramm ist, das die Software-Wechselwirkungen abbildet, die auftreten, wenn der Benutzer ein Item auswählt;

[0027] Abb. 14 ein Objektdiagramm ist, das die Controller-Schnittstelle und die Controller zeigt, die durch die Controller-Schnittstelle laufen;

[0028] Abb. 15 ein Objektdiagramm der Builder-Controller ist;

[0029] Abb. 16 ein Objektdiagramm des Elementepakets der vorliegenden Erfindung ist;

[0030] Abb. 17 ein Objektdiagramm ist, das die Wechselbeziehung zwischen dem Datenansichtspaket, dem Rahmenmodell, dem Controller und dem Graphrahmen zeigt;

[0031] Abb. 18 ein Flussdiagramm ist, das die in Patentanspruch 1 definierten Verfahrensschritte der vorliegenden Erfindung schematisch veranschaulicht; und

[0032] Abb. 19 ein Flussdiagramm ist, das die Verfahrensschritte der in Patentanspruch 6 definierten erfindungsgemäßen Ausführungsart schematisch veranschaulicht.

[0033] Alle abgebildeten Diagramme verwenden Standard-UML-Notation, um die Klassen, Schnittstellen, Komponenten und Relationen zu charakterisieren.

DETAILLIERTE BESCHREIBUNG DER ERFINDUNG

[0034] Die folgenden Bezeichnungen werden überall in der Patentbeschreibung benutzt und haben die folgenden Bedeutungen.

[0035] „Abstrakte" Klassen sind allgemeine Verfahren zum Definieren von Anwendungsprogramm-Schnittstellen (APIs = Application Program Interfaces) und zum Implementieren dieser Funktionen.

[0036] „Controller" sind Softwarekomponenten, die Interaktivität in einem Graphen steuern. Beispiele für einen Controller sind Eigenschaftsänderungs-Lauscher, die Graphrahmen belauschen. Sobald er ein Eigenschaftsänderungs-Ereignis hört, kann sich der Controller einfach auf den Graphrahmen **134** einstellen oder er kann sein Aussehen auf der Basis der im Graphrahmen **134** enthaltenen Information aktualisieren.

[0037] „Daten" heißt aufgenommene Beobachtungen von Quantitäten, Qualitäten oder Relationen. Daten haben keine zwangsläufige Organisation oder Struktur, sondern sind eine Informationssammlung und werden manchmal als Rohdaten oder Quelldaten bezeichnet. Daten enthalten auch beispielsweise Metadaten, assoziierte Daten und Annotationsdaten.

- Roh- oder Quelldaten residieren im Computerspeicher, beispielsweise in einer freien Textdatei.
- Jedes Metadatenstück hat eine Stringbeschreibung und kann einen Typ haben, um seinen Typ erkennbar zu machen.
- Assoziierte Daten sind Daten, die mit den tatsächlichen Daten assoziiert sind und für Benutzeranfragen und zum Tiefergehen (Untersuchen von Teilmengen der Daten) benutzt werden.
- Annotationsdaten sind mit den tatsächlichen Daten verbunden, die ermöglichen, dass eine Art von Annotation im Graphen erscheint.

[0038] „Datenansicht" ist eine Funktion, die Rohdaten indexiert, um eine Datenmenge zu erzeugen. Verschiedene Indexier- oder Organisationsschemata können je nach Wunsch benutzt werden wie beispielsweise hierarchische, relationale oder topologische.

[0039] „Rahmenmodellzustand" ist eine Klasse, um den Zustand des Rahmenmodells effizient über den Draht weiterzuleiten. In DataView.addFrameModelState (Rahmenmodellzustand) können Datenansicht-Implementierungen ein neues Rahmenmodell durch Verwendung des Konstruktors instantiieren, der einen Rahmenmodellzustand annimmt.

[0040] Eine „Generische Funktion" benutzt Reflexion, um ein Funktionsobjekt aus einem vollständig qualifizierten Verfahrensnamen einer Funktion zu erzeugen, die Zielvariable(n) in einer Datenansicht und eine optionale Hash-Tabelle mit zusätzlichen Parametern für die Funktion. Die Resultate der Funktion sind persistent.

[0041] „Instantiieren" bedeutet eine Instanz bereitstellen.

[0042] Ein „Lauscher" ist ein Untermodul des Computerprogramms, das nach einem Befehl sucht und das Modul (von dem das Untermodul ein Teil ist) informiert, dass die Nachricht gesendet wurde, sodass eine Aktion ausgeführt werden kann.

[0043] „Renderer" sind Wrapper für System-Toolbox/Toolkit-Zeichenobjekte und/oder eine Graphik-Klassenbibliothek. Renderer sind verantwortlich für das Erzeugen, Zeichnen, Zugriffprüfen und Pflegen von GPL-Primitiven **180** (GPL = Graphics Production Library). Zugriffprüfen wird über Pointer-Ereignisse **184** ausgeführt, die Erweiterungen von Mouse-Ereignisse **186** sind. So können Renderer Mouse-Lauscher und Mousebewegungs-Lauscher hinzufügen und entfernen. Renderer stellen auch Schichten zum Zeichnen zur Verfügung (d. h. ein Objekt in Schicht **2** sollte ein Objekt in Schicht **1** „überlagern"). Renderer arbeiten in einem von zwei „Fensterdehnungs"-Modi (RESIZE und NORESIZE – wie unten), die die Größe der Primitiven im Verhältnis zum umgebenden Fenster beeinflussen. Schließlich verbreiten Renderer Eigenschaftsänderungs-Ereignisse, wenn sich die Anzahl der Schichten („Number Of Layers") oder der Fensterdehnungstyp („Window Stretch Type") ändert. Renderer bilden ein MIN bis MAX Koordinatensystem über die tatsächliche Zeichenebene ab. Koordinatenwerte nehmen von links nach rechts und von unten nach oben zu.

[0044] „Variable" bedeutet eine Menge von Variablen der Form V = varset[X]. Ein „varset" oder eine „variable Datenstruktur" ist eine wie folgt definierte Menge:
varset $[X_1, ..., X_n] = [I, X_1, ... , X_n, f]$, wo
$X_1, ..., X_n$ n Mengen repräsentiert,
I eine Indexmenge {1, 2, ..., m} ist, die über alle natürlichen Zahlen N variiert,
$f: I \rightarrow X_1 \times X_2 \times ... \times X_n$ und f über alle mögliche Funktionen dieser Form variiert.

[0045] V kann eine kategorische Variable sein (d. h. X ist eine endliche Menge); oder V kann eine stetige Variable sein (d. h. X ist eine Menge reeller Zahlen).

[0046] Es folgt eine Beschreibung eines Softwaresystems gemäß der vorliegenden Erfindung. Die Software kann auf einem beliebigen Medium gespeichert sein, typischerweise einem optischen oder magnetischen, das dazu fähig ist, den Softwarecode aufzubewahren, und von einem Allzweck-Computersystem gelesen oder ausgeführt werden kann. Das Softwaresystem kann auch über ein Netz laufen unter Verwendung einer typischen Client-Server-Anordnung.

[0047] Es folgt eine Liste von Softwarekomponenten, die einem oder mehreren der folgenden Diagramme gemeinsam sind, und die jeweiligen Zeichnungen, auf die sie positioniert sind:

Komponente	ABB.
Achse **114**	10, 11
Controller-Schnittstelle **40**	2, 14, 15
Koordinatenspezifikation **50**	2, 5
Datendurchlauflauscher-Schnittstelle **62**	3, 4
Datenansicht-Schnittstelle **14**	1–4
Element **46**	2, 11, 16
Rahmenlayout-Schnittstelle **42**	2, 11
Glyphen-Schnittstelle **32**	1, 10, 11, 13
Graph **10**	1, 2, 11
Graphdimension **100**	5, 10
Item-Rahmen-Schnittstelle **18**	1, 3, 5
Legende **112**	10, 11
Legendenzugriff-Ereignis **130**	10, 13
Mouse-Ereignis **186**	12, 13
Pointer-Ereignis **184**	12, 13
Primitive Schnittstelle **34**	1, 12, 13
Skalierungsspezifikation **48**	2, 5
Transformation **36**	1, 11, 12
Variable Transformationsspezifikation **52**	2, 5

[0048] Mit Bezug auf die Abbildungen im Allgemeinen und insbesondere auf Abb. 2 ist ein Objektdiagramm veranschaulicht, das die Primärelemente der erfindungsgemäßen Software zum Erzeugen einer graphischen Repräsentation von Daten abbildet. Die vier Hauptkomponenten des Systems sind Graph **10**, Datenansicht-Schnittstelle **14**, Controller-Schnittstelle **40** und Rahmenlayout-Schnittstelle **42**.

[0049] Der Graph **10** kann durch Festsetzen eines algebraischen Modellausdrucks oder Strings **44** und durch Manipulieren einer Anzahl von Eigenschaften vervollständigt werden, die Folgendes umfassen: Elemente **46**, Skalierungsspezifikationen **48**, Koordinatenspezifikationen **50** und Variable Transformationsspezifikationen **52**. Nachdem die Datenansicht-Schnittstelle **14** des vorliegenden Systems Daten empfangen und manipuliert hat, meldet sie den anderen Softwarekomponenten Änderungen der Daten. Die Controller-Schnittstelle **40** ist für alle Wechselwirkungen mit dem Graphen **10** verantwortlich, den Aufbau des Graphen eingeschlossen. Die Rahmenlayout-Schnittstelle **42** steuert das Aussehen und den Eindruck des Graphen **10**.

[0050] In Abb. 1 wird ein Blockdiagramm gezeigt, das das erfindungsgemäße Verfahren und System zum Erzeugen einer graphischen Repräsentation von Daten schematisch veranschaulicht. Abb. 1 bildet die gesamte Operationsfolge ab, die die Computersoftware ausführt, um die Information in graphischer Form als Graph **10** visuell zu repräsentieren. Die Komponenten des Graphs **10** sind Daten **12**, die aufgenommene Beobachtungen von Quantitäten, Qualitäten oder Relationen sind. Vor der Repräsentation in graphischer Form haben die Daten **12** keine zwangsläufige Organisation oder Struktur.

[0051] Wie Abb. 1 zeigt, ist die Datenansicht **14** eine Funktion, die eine Datenmenge **16** aus den Daten **12** erzeugt. Verschiedene Datenansichten **14** nutzen verschiedene Schemata, um die Daten **12** zu organisieren. Das Datenansicht-Objekt implementiert die Datenansicht **14** durch Festlegen eines Indexierungsschemas und durch Assoziieren der Daten **12** mit den Indizes, um die Datenmenge **16**, eine indexierte Menge der Daten **12**, zu erzeugen. Jedes Element in der Datenmenge **16** wird als ein „Eintrag" oder ein „Wert" bezeichnet. Die Datenmenge **16** ist durch ein Schema indexiert, das dem Benutzer ermöglicht, Daten **12** in der Datenmenge **16** aufzufinden und die separaten Elemente miteinander zu assoziieren.

[0052] Der Item-Rahmen **18** ist eine Schnittstelle, die eine Kollektion von Funktionen auf Indexmengen spezifiziert hat. Diese Funktionen sind Transformationen auf Indexmengen; sie erzeugen Indexmengen. Einige Funktionen im Item-Rahmen **18** können Indizes permutieren; beispielsweise kann eine Baumstruktur (siehe beispielsweise Abb. 8 und Abb. 9) in eine sequentielle Indexliste umgewandelt werden. Andere Funktionen des Item-Rahmens **18** können eine Datenmenge **16** zur Teilmenge machen, indem einige Indizes in Nullwerte umgewandelt werden (d. h. sie werden gelöscht). Funktionen des Item-Rahmens **18** werden benutzt, um zur Erzeugung von Variablen erforderliche Daten **12** zur organisieren und herauszufiltern.

[0053] Nachdem die Daten **12** in Datenmengen **16** indexiert worden sind, bildet eine Funktion Variable **Abb.** 20 eine Datenmenge **16** auf eine Variable Datenstruktur ab, die auch „Varset" genannt wird. Eine Variable Datenstruktur ist eine wie folgt definierte Menge:

varset [X_1, ..., X_n] = [I, X_1, ..., X_n, f], wo
X_1, ..., X_n n Mengen repräsentiert,
I eine Indexmenge {1, 2, ..., m} ist, die über alle natürlichen Zahlen N variiert,
f: I → X_1 × X_2 × ... × X_n und f über alle mögliche Funktionen dieser Form variiert.

[0054] Die Variable Datenstruktur ist auch eine Menge von Variablen der Form V = varset[X], wo die Menge von Variablen nicht der Algebra ausgesetzt wurde. Variablen sind kategorisch oder stetig, was nur darauf basiert, wie die Variablen definiert sind. V ist eine kategorische Variable, wenn X eine endliche Menge ist; V ist eine stetige Variable, wenn X eine Menge reeller Zahlen ist.

[0055] Die Abbildung geht von einer Keyindexmenge auf Wertmengen in der Datenmenge **16**. Die Variable **Abb.** 20 muss den Bereich jeder Variablen überwachen, sodass Fehler in der Datenmenge **16** abgefangen werden können. Liegt beispielsweise ein Wert in der Datenmenge **16** nicht im Bereich, muss die Variable **Abb.** 20 entweder in der Lage sein, den Wert passend zuzuweisen oder melden, dass der Variablenwert nicht verarbeitet werden kann. Die Variable **Abb.** 20 muss auch weiterhin für alle möglichen Werte in einer Domäne Werte im Bereich korrekt zurückliefern, während sich die Datenmenge **16** mit der Zeit ändert.

[0056] Nachdem die Datenmenge **16** durch die Variable **Abb.** 20 in die Variable Datenstruktur **22** abgebildet worden ist, implementiert das Algebraobjekt eine Algebra **24** auf der Variablen Datenstuktur **22**. Die Ausgabe der Algebra **24** ist eine Menge von Variablen **22**. Eine „Algebra" besteht aus einem Mengensystem, Mengenoperatoren und Regeln für die Kombination der Operatoren. Ein Operator ist eine auf der Menge definierte Funktion, die einen Wert in dieser Menge zurückliefert.

[0057] Die Algebra **24** hat drei binäre Operatoren-Blend-Operator (+) **26**, Nest-Operator (/) **28** und Cross-Operator (*) **30** – und eine Menge assoziierter Regeln. Die hierarchische Ordnung der Operatoren ist Blend-Operator (+) **26**, Cross-Operator (*) **30** und Nest-Operator (/) **28**. Blend **26** wird zuletzt berechnet, während Nest **28** zuerst berechnet wird. Die Ordnung der Operatoren kann durch den Gebrauch von Klammern geändert werden. Die Regeln für den Operator schließen Assoziativität, Distributivität und Kommutativität ein. Nur Blend **26** ist kommutativ. Der Blend-Operator **26** involviert eine Vereinigung im Bereich. Der Cross-Operator **30** involviert ein kartesisches Produkt im Bereich. Der Nest-Operator **28** schichtet Werte.

[0058] Die Softwarekomponente, die alle algebraischen Spezifikationen zum Abbilden von Datenmengen auf Mengen von Variablen enthält, ist das Rahmenmodell **94**, das in **Abb. 5** veranschaulicht ist. Das Rahmenmodell **94** enthält auch die Struktur von Dimensionen, aus denen ein Graphrahmen **134** (von **Abb. 11**) wird und Daten **12** (von **Abb. 1**) für Elemente **46**.

[0059] Die algebraischen Spezifikationen enthalten Variable Transformationsspezifikationen **52**, Skalierungsspezifikation **48**, Koordinatenspezifikationen **50**, Elementspezifikation **96** und einen Baum **98**. Die tatsächliche Rahmenstruktur wird erzeugt, indem alle diese Spezifikationen in ein Baumobjekt **98** interpretiert werden, und auch die Element-Graphdimensionen **100** und die Item-Rahmen-Schnittstelle **18**, die die Daten **12** für den Graphen **10** sind.

[0060] Der Baum **98** umfasst Graphdimensionen **100** (von **Abb. 10**), die in einer Form organisiert sind, die für eine Ansicht zur Konstruktion eines Graphen „vorbereitet" ist. Der Baum **98** kann aus einer beliebigen Anzahl von Teilbäumen bestehen, wobei jede „Ebene" eines Baums eine Cross-Dimension und die Zeilen Nest-Dimensionen repräsentieren. Blend-Operationen **26** werden durch Blenden von zwei Bäumen **98** mit identischen Strukturen (die algebraische Expansion garantiert der Ästhetik, dass dies der Fall ist) ausgeführt. Das Blenden **26** von zwei Bäumen erzeugt neue Ebenen in einem Baum. Das Nesten **28** von zwei Bäumen erzeugt „Zeilen" in einem Baum. Es folgen Beispiele von algebraischen Aussagen und ihre Baumäquivalente:

BEISPIEL 1.

Algebraischer Ausdruck: a·b·c·d·e·f

Baumäquivalent:

```
|
| [e]
|   [d]
|     [c]
|       [b]
|         [a]
|
```

BEISPIEL 2.

Algebraischer Ausdruck: a·b + a·c

Baumäquivalent:

```
| [c]  + [b]   =  [c + b]
|  [a]   [a]       [a + a]
|
```

BEISPIEL 3.

Algebraischer String: a·b/c·1·c

Baumäquivalent:

```
|
| [1]
|   [c]
|     [b/c = 1]
|       [a]
|     [b/c = 2]
|       [a]
|     [b/c = 3]
|       [a]
|
```

BEISPIEL 4.

Algebraischer Ausdruck: y·1·(a·b)/c·1

Baumäquivalent:

```
|
| [1]
|   [c]
|     [b/c = 1]
|       [a/c = 1]
|         [1]
|           [y]
|     [b /c = 2]
|       [a/c = 2]
|         [1]
|           [y]
|
```

[0061] Abb. 6 bildet das Algebrapaket als ein Tertiärbaummodell ab. Das Algebrapaket ist ein einfaches Objektmodell für algebraische Ausdrücke, drei Klassen umfassend: Algebra **24**, Faktor **106** und Ausdruck **108**. Im gezeigten Modell ist die Algebra **24** die abstrakte Überklasse, und Faktor **106** und Ausdruck **108** sind Unterklassen der Überklasse. Die Algebra **24** wandelt den String **44** (in Abb. 2 gezeigt) in einen Baum **98** um (von Abb. 5). Der Faktor **106** ist eine Grundeinheit der Algebra **24**. Faktoren **106** entsprechen direkt den Stringnamen der Menge der Variablen **22** (von Abb. 1). Beispielsweise sind „a" und „b" Faktoren im Ausdruck „a + b".

[0062] Ausdrücke **108**, die auch durch Tertiärbäume repräsentiert werden, bestehen aus einer linken Seite, einer rechten Seite und einem Operator. Die Seiten sind Instanzen der Algebra **24** (d. h. entweder Ausdrücke **108** oder Faktoren **106**). Der Operator ist einer der Folgenden: Blend (+) **26**, Nest (/) **28** und Cross (*) **30**.

[0063] Stringausdrücke **44** werden in Monome expandiert und dann geparst, um ein Ausdrucksobjekt zu erzeugen. Die Expansion wird durch eine statisches Verfahren im Ausdruck **108** ausgeführt, das den ursprünglichen String **44** untersucht und in Monome in einem neuen String expandiert, wobei die Assoziativitäts-, Distributivitäts- und Kommutativitätsregeln berücksichtigt werden. Die Operatoren für die Expansion sind Blend (+) **26**, Nest (/) **28** und Cross (*) **30**. Ein anderes statisches Verfahren nimmt dann den expandierten String und parst ihn, um ein wirkliches Ausdrucksobjekt zu erzeugen. Es ist auch denkbar, dass ein beliebiges algebraisches Ausdrucksobjekt von Hand erzeugt werden könnte.

[0064] Die unten gezeigten Beispiele veranschaulichen, wie ein algebraischer Ausdruck handcodiert wird:

BEISPIEL 1

Ausdruck: (einfache Cross-Operation mit zwei Variablen): (a·b)
Der Ausdruck kann wie folgt handcodiert werden:
Ausdruck =
neuer Ausdruck (neuer Faktor („a"), Algebra.CROSS, neuer Faktor(„b"))

[0065] Das Objektdiagramm für dieses Beispiel wird in Abb. 7 gezeigt.

BEISPIEL 2

Ausdruck: (a + b)·c
Der Ausdruck wird in Monome expandiert: a·c + b·c

Der Ausdruck kann wie folgt handcodiert werden:
Faktor a = neuer Faktor („a");
Faktor b = neuer Faktor („b");
Faktor c = neuer Faktor („c");
Ausdruck ex1 = neuer Ausdruck (a, Algebra.CROSS, c);
Ausdruck ex2 = neuer Ausdruck (b, Algebra.CROSS, c);
Ausdruck Hauptausdruck =
neuer Ausdruck (ex1, Algebra.BLEND, ex2)

[0066] Das Objektdiagramm ist in Abb. 8 gezeigt.

BEISPIEL 3

Ausdruck: (a*b)/c
Der Ausdruck wird in Monome expandiert: a/c * b/c
Der Ausdruck kann wie folgt handcodiert werden:
Faktor a = neuer Faktor („a");
Faktor b = neuer Faktor („b");
Faktor c = neuer Faktor („c");
Ausdruck ex1 = neuer Ausdruck (a, Algebra.NEST, c);
Ausdruck ex2 = neuer Ausdruck (b, Algebra.NEST, c);
Ausdruck Hauptausdruck =
neuer Ausdruck (ex1, Algebra.CROSS, ex2)

[0067] Das Objektdiagramm ist in Abb. 9 gezeigt.

[0068] Es werden auch Algebra-Ausnahmen **110** für bestimmte Funktionalitäten bereitgestellt, die nicht in der normalen Algebra-Klasse **24** verarbeitet werden können.

[0069] Abb. 3 zeigt ein Objektdiagramm des gesamten Datenansicht-Pakets des vorliegenden Systems. Das Datenansicht-Paket enthält die Klassen, die dem Programm erlauben, auf Daten zuzugreifen und sie zu manipulieren. Die Datenansicht-Schnittstelle **14** stellt Daten **12** in Form der Geometrie zur Verfügung, die zum Erzeugen eines Graphen erforderlich ist. Die Datenansicht-Schnittstelle **14** fungiert auch als „Linse" für die Software, indem sie darstellt, was der Rest des Systems „sieht", wenn er Daten **12** anfordert. Die abstrakte Datenansicht-Klasse **54** implementiert Funktionen der Datenansicht-Schnittstelle **14**. Die Geometrie für jedes Element eines jeden Panels (Facette) eines Graphen ist in der Item-Rahmen-Schnittstelle **18** gespeichert, die im Wesentlichen eine Tabelle ist.

[0070] Das Datenansicht-Paket stellt auch einen Weg bereit, um die Assoziierte Datenschnittstelle **56** (Graph-Metadaten) einzubeziehen. Die Assoziierte Datenschnittstelle **56** hat eine Unterstützungsklasse, die Assoziierte Datenunterstützung **58**, die die Routinen unterstützt, die von der Software gebraucht werden, um auf andere Daten zuzugreifen und diese Daten mit den primären Daten **12** zu verknüpfen.

[0071] Das Datenansicht-Paket enthält auch das Datendurchlauf-Ereignis **60** und die entsprechende Datendurchlauflauscher-Schnittstelle **62**, die Clienten benachrichtigt, wenn ein Datendurchlauf bevorsteht und wenn der Datendurchlauf abgeschlossen ist. Die Datenansicht **14** feuert jedes Mal ein Datendurchlauf-Ereignis **60** ab, wenn die Datenansicht **14** geändert worden ist; beispielsweise, wenn der Datenansicht neue Daten hinzugefügt worden sind.

[0072] Ein vollständiger Datendurchlauf funktioniert wie folgt:
- Ein Client ruft DataView.beginDataPass() auf.
- Die Datenansicht-Schnittstelle **14** löscht alle vorherigen Spezifikationen (einschließlich Skalierungsspezifikation **48** (von Abb. 2), Elementspezifikation **96** (von Abb. 5), Variable Transformationsspezifikation **52** (von Abb. 2), Koordinatenspezifikation **50** (von Abb. 2) und Rahmenmodellzustand).
- Datenansicht-Schnittstelle **14** benachrichtigt Lauscher über aboutToDataPass().
- Lauscher fügen für den Datendurchlauf erforderliche Spezifikationen hinzu.
  – Beispielsweise fügt das Rahmenmodell **94** (in Abb. 5 gezeigt) – das auch eine Datendurchlauflauscher-Schnittstelle **62** ist –
  eine Kopie der Spezifikationen hinzu (in Form eines Rahmenmodellzustands-Objekts).
- Der Datendurchlauf findet statt.

- Wenn der Datendurchlauf abgeschlossen ist, benachrichtigt die Datenansicht-Schnittstelle **14** Lauscher über finishedDataPass().
- Die Lauscher können, falls erforderlich, ihre Daten über DataView.getResults() abfragen.

[0073] Die wichtigste von der Datenansicht-Schnittstelle **14** ausgeführte Funktion ist die Verarbeitung der in einem Rahmenmodell enthaltenen Spezifikationen. Diese Spezifikationen sind die Datenmanipulationen, die ausgeführt werden müssen, um die Elemente **46** (von Abb. 2) in einem Graphen zu zeichnen. Diese Spezifikationen sind die „Graphgrammatik"; sie enthalten Algebraischen Ausdruck, Skalierspezifikation **48** (von Abb. 2), Elementspezifikation **96** (von Abb. 5), Variable Transformationsspezifikation **52** (von Abb. 2) und Koordinatenspezifikation **50** (von Abb. 2), wie oben erörtert. Die Datenansicht-Schnittstelle **14** nimmt die Spezifikationen und erzeugt die Geometrie für die Elemente im Rahmenmodell (von Abb. 12). Andere Erfordernisse sind Pinselunterstützung und Fallebenenzugriff. Ausnahmen sind vorgesehen für die konkrete Datenansicht-Schnittstelle **14**, die eine bestimmte Funktionalität nicht unterstützen kann.

[0074] Eine Unterstützungsklasse, Datenansicht-Unterstützung **64**, wird bereitgestellt zusammen mit einer Implementierung der Datenansicht-Schnittstelle **14** unter Verwendung von JDBC-Datenansicht **66** (JDBC = Java Database Connectivity), die Quelldaten von der JDBC-Quellspezifikation **68** erfasst. Die JDBC-Datenansicht **66** stellt eine Verknüpfung mit einer Datenbank zur Verfügung, sodass das Programm Daten von dieser Datenbank benutzen kann. Datenansicht-Unterstützung **64** ist eine Implementierung der Datenansicht-Schnittstelle **14**, die den DMS-Item-Rahmen **70** (DMS = Datenmanagementsystem) benutzt. Der DMS-Item-Rahmen **70** fungiert als Pointer, der auf ein einzelnes Item in einem Datenmanagementsystem zeigt. Die JDBC-Datenansicht **66** enthält einen JDBC-Datenansicht-Anpasser (nicht gezeigt), um für die Daten **12**, Metadaten und synthetischen Variablen die Datenquelle(n) einzurichten.

[0075] Der JDBC-Datenansicht-Anpasser (JDBC = Java Database Connectivity) ist eine Benutzerschnittstelle (UI = User Interface) zum Anpassen der JDBC-Datenansicht **66**. Er verleiht die Fähigkeit, alle Datenbankanforderungen in einer JDBC-Datenansicht **66** anzupassen, sowie die Fähigkeit, synthetische Variablen über Funktionen hinzuzufügen. Im Folgenden werden die UI-Screens beschrieben:

- Hauptdaten (2 Screens) – Der erste Screen verlangt den JDBC-Treiber, die Datenbank-URL, einen Benutzernamen, ein Passwort und eine SQL-Anweisung (SQL = Sequential Query Language). Der zweite Screen fordert den Benutzer auf, (a) jede Variable in der Resultatsmenge als kategorisch zu kennzeichnen und (b) für jede Variable einen Typ auszuwählen (Zahl, Text).
- Tabellen-Meta (2 Screens) – Der erste Screen verlangt den JDBC-Treiber, die Datenbank-URL (Universal Resource Locator), einen Benutzernamen, ein Passwort und eine SQL-Anweisung. Der zweite Screen fordert den Benutzer auf, Folgendes auszuwählen: (a) eine Spalte in der Resultatsmenge, die die Beschreibungen der Metadaten enthält, (b) eine Spalte, die die MIMF-Typen für die Metadaten enthält und, (c) eine Spalte, die die eigentlichen Metadaten enthält.
- Variablen-Meta (2 Screens) – Der erste Screen verlangt den JDBC-Treiber, die Datenbank-URL, einen Benutzernamen, ein Passwort und eine SQL-Anweisung. Der zweite Screen fordert den Benutzer auf, Folgendes zu kennzeichnen: (a) die Spalte in der Resultatsmenge, die mit den Variablennamen übereinstimmt, (b) die für Metadaten zu benutzenden Spalten und (c) die jeweiligen MIME-Typen.
- Wert-Meta (2 Screens) – Der erste Screen verlangt den JDBC-Treiber, die Datenbank-URL, einen Benutzernamen, ein Passwort und eine SQL-Anweisung. Der zweite Screen fordert den Benutzer auf, Folgendes zu kennzeichnen: (a) die Spalte in der Resultatsmenge, die mit den Kategorienamen übereinstimmt, (b) die für Metadaten zu verwendenden Spalten und (c) die jeweiligen MIME-Typen.
- Item-Meta (2 Screens) – Der erste Screen verlangt den JDBC-Treiber, die Datenbank-URL, einen Benutzernamen, ein Passwort, eine SQL-Anweisung und die Variable, zu der Wert-Metadaten zu addieren sind. Der zweite Screen fordert den Benutzer auf, die für Metadaten zu benutzenden Spalten und die jeweiligen MIME-Typen zu kennzeichnen.
- Funktionen (1 Screen) – Der Screen fordert eine auf einer Funktion basierende neue (synthetische) Variable an sowie eine oder mehrere der bestehenden Variablen. Doppelklicken auf der Box für die synthetische Variable ermöglicht dem Benutzer, die synthetische Variable zu bearbeiten. „Add" addiert eine neue synthetische Variable und „Remove" entfernt die synthetische Variable.

[0076] Außerdem wird im Datenansicht-Paket ein Modell für die Assoziierte Datenschnittstelle **56** (Metadaten) zum Verknüpfen von Information mit beliebigen Objekten bereitgestellt. Die Assoziierte Datenschnittstelle **56** kann an ein einzelnes Item **71** von Daten **12**, Gruppen von Items **71**, Mengen von Variablen **22**, Kategorien innerhalb einer Menge der Variablen **22** oder ganze Tabellen angeschlossen werden. Die Abstrakte Klasse Item-Unterstützung **74** ist ein allgemeines Verfahren zum Definieren einer Anwenderprogramm-Schnittstelle (API = Application Program Interface) und zum Implementieren einer Funktion der Item-Schnittstelle **72**. Die

Item-Unterstützung **74** kann irgendeins der Folgenden benutzen: ein Text-Item **76**, ein Datum-Item **78** oder ein Zahl-Item **80**. Der Abstrakte Item-Rahmen **82** implementiert Funktionen der Item-Rahmen-Schnittstelle **18**. Der Abstrakte Item-Rahmen **82** hat eine Unterstützungsklasse Item-Rahmen-Unterstützung **84**.

[0077] Die elementare Zeichenschnittstelle, Primitive Graphschnittstelle **34** (von Abb. 1), implementiert die Assoziierte Datenschnittstelle **56**, sodass Metadaten in jedes auf den Screen gezeichnete Objekt eincodiert werden können. Die Assoziierte Datenschnittstelle **56** kann aus jedem gültigen Java-Objekt bestehen. Jedes assoziierte Datenstück enthält eine String-Beschreibung **44** der assoziierten Daten, die Daten selbst als Java-Objekt und einen optionalen MIME-Typ für diese Daten. Der MIME-Typ kann von Clienten dazu benutzt werden, um einen passenden „Abspieler" für die Metadaten zu bestimmen.

[0078] Funktionen werden unter Verwendung von statischen finalen Klassen implementiert. Die Verfahren in dieser Klasse operieren auf einem Primitiv **180** (von Abb. 12). Die Generische Funktionsklasse (nicht gezeigt) ist ein Funktionsobjekt, das Java-Reflexion auf den statischen Verfahren benutzt. Auf einem String-Funktionsnamen basierend, sucht ein Generisches Funktionsobjekt das korrekte Verfahren auf und wird ein funktionskonformes Objekt. So erhalten wir den Effekt von potentiell vielen Arten von Funktionsobjekten unter Verwendung von relativ wenigen Klassen. Außerdem müssen wir nur zusätzliche primitive Operationen zu den statischen finalen Klassen addieren, um mehr Funktionen einzubeziehen.

[0079] Nachdem die Menge von Variablen **22** erlangt worden ist, besteht der nächste Schritt des erfinderischen Verfahren aus dem Abbilden einer Menge aus der Menge der Variablen **22** in eine Punktmenge. Dies wird durch eine Glyphe **32** erreicht – eine Schnittstelle, die eine bestimmte Graphfunktion benutzt, um einen Primitiven Graphen **34** aus einer Menge in der Menge der Variablen **22** zu erzeugen. Die Glyphe **32** führt auch andere organisatorische Aufgaben aus; beispielsweise die Mengen in der Menge der Variablen **22** mit den Dimensionen eines geometrischen Raums in Beziehung zu setzten, in den der Graph eingebettet wird.

[0080] Der resultierende Primitive Graph **34** ist eine Teilmenge von der Cross-Operation unterworfenen Mengen. Das Graphobjekt ist eine Kollektion, die den Primitiven Graphen **34** und die Verfahren umfasst, die zur Repräsentation des Primitiven Graphen **34** als ein geometrisches Objekt erforderlich sind.

[0081] Nachdem der Primitive Graph **34** gewonnen worden ist, umfasst der nächste Schritt des erfinderischen Verfahrens die Transformation des Primitiven Graphen **34** und seine Repräsentation in Form eines Koordinatensystems. Das geschieht durch die Transformation **36**. Die Transformation **36** ist ein System oder Schema zum Auffinden eines Raumpunktes, dessen Koordinaten gegeben sind. Dieses Schema umfasst eine Achse **114** für jede Dimension, eine Skale (siehe Abb. 10) für jede Achse **114** und ein Verfahren zum Auffinden eines beliebigen Raumpunkts. Das bekannteste Koordinatensystem ist das kartesische Koordinatensystem. Das Standard-Koordinatensystem ist das 2D-Koordinatensystem **140**, ein Graphrahmen für zweidimensionale Graphen. Das Transformationsobjekt transformiert den Primitiven Graphen **34** und repräsentiert ihn in rechtwinkligen Koordinaten, Polarkoordinaten oder anderen Koordinatensystemen einschließlich beispielsweise konformer Abbildungen und geographischer Projektionen. Die Transformationen **36** sind jedoch auf Transformationen beschränkt, die die fünktionale Relation zwischen der Domäne und dem Bereich eines bestimmten Primitiven Graphen **34** beibehalten.

[0082] Die Koordinatenspezifikation **50** wird benutzt, um Facettierung zu bezeichnen, wenn die Anzahl der Dimensionen zweideutig ist. Jede Koordinatenspezifikation macht eine bestimmte Facette explizit. Beispielsweise könnte ein Graph mit sechs Dimensionen a*b*c*d*e*f ein im 2D-Raum zweimal facettierter zweidimensionaler Graph oder ein im 3D-Raum facettierter dreidimensionaler Graph sein. Die Verwendung der Koordinatenspezifikation **50** macht deutlich, wie der Graph facettiert ist.

[0083] Ein letzter Schritt in der Konstruktion des Graphen involviert die ästhetische Repräsentation der Punktmenge in die Graphik. Dieser Schritt umfasst das Abbilden der Punktmenge in eine ästhetische Repräsentation durch Anwendung der Ästhetik **38**. Die Ästhetik **38** ist eine Funktion, die Punkte oder Punkteinträge in Strings oder reelle Zahlen abbildet, die als Eingabe in eine physikalische Anzeigeeinrichtung dienen. Die Ästhetik **38** ist auch ein Objekt, das ästhetische Funktionen in die Konstruktion eines Graphen **10** implementiert.

[0084] Der Graph **10** ist ein unter einer oder mehreren Ästhetischen Funktionen **38** erstelltes Kombinationsbild des Primitiven Graphen **34**. Das graphische Objekt ist dafür verantwortlich, den Graphen **10** in einem Anzeigesystem zu realisieren.

[0085] Abb. 10 präsentiert eine Übersicht der Dimensionsaspekte der vorliegenden Erfindung. Im Allgemei-

nen ist die Graphdimension **100** eine Neudarstellung der Daten **12**, wobei die Datenwerte nach einer numerischen Skale abgebildete wurden. Der String **44** (von Abb. 2) enthält die Befehle zum Durchführen der Neudarstellung der Daten **12**. Die Graphdimension **100** parst einen Ausdrucksbaum, um zu bestimmen, wie die Daten in Form von Mengen der Variablen **22** (von Abb. 1) kombiniert werden. Die Graphdimension **100** bildet dann die Items einer oder mehrerer Variabler Mengen **22** auf eine Skale, die eine Dimension in einem Graphrahmen **134** (in Abb. 11 gezeigt) ist. Ein in der Graphdimension **100** enthaltener Guide **116** wird verwendet, um die Resultate nach einer numerischen Skale abzubilden.

[0086] Im Fall der kategorischen Variablen ist die numerische Skale eine ganze Zahl von Null bis zu einem durch (Anzahl der Kategorien – 1) definierten Wert; im Fall der stetigen Variablen geht die numerische Skale von min(Daten) bis max(Daten). Die Graphdimension **100** bildet Attribute ab, die eines der durch den „Typ" der Graphdimension **100** definierten Attribute sein können wie beispielsweise Position (z. B. Größe, Form), Farbe (z. B. Farbton, Helligkeit oder Sättigung), Bewegung, Rotation, Unschärfe, Transparenz oder Textur (z. B. Muster, Orientierung oder Körnung). Als Alternative kann das Attribut nichtvisuell sein wie beispielsweise Sound. Die Graphdimension **100** wird als ein „Modell" angesehen und die Legende **112** und Achse **114** sind „Ansichten" des Modells.

[0087] Die Graphdimension **100** wird die Assoziierte Datenschnittstelle **56** (von Abb. 3) (Metadaten) von allen Mengen der Variablen **22** präservieren, die sie enthält. Metadaten der variablen Schicht werden kombiniert und können über getAssociatedData() abgerufen werden. Metadaten der kategorischen Schicht werden innerhalb der Kategorien präserviert und kombiniert und können über getAssociatedDataFor() abgerufen werden.

[0088] Wie in Abb. 10 gezeigt ist, enthält der Guide **116** ein Scalebuilder-Objekt **118**, das die numerische Skale für die Dimension aufbaut. Die numerische Skale kann ein beliebiges Scalebuilder-Objekt **118** sein, und die Attribute sind ein Array von Java-Objekten. Gibt es weniger Attribute als Skalenwerte, dann werden die Guides **116** durch die Attributliste recyceln. Die abstrakte Klasse Guide **116** behandelt den Kategorischen Guide **120** und den Stetigen Guide **122** auf dieselbe Weise. Der Kategorische Guide **120** bildet Attribute und Kategorien auf eine numerische Skale ab, die von Null bis zum Wert n-1 geht, wo n die Anzahl der Kategorien repräsentiert. Der Stetige Guide **122** bildet Attribute auf eine stetige numerische Skale ab.

[0089] Der Scalebuilder **118** nimmt die Minimum- und Maximumwerte für eine Zahlengruppe und erzeugt eine Skale. Der Client kann den Gebrauch eines jeden der Folgenden anfordern:

- ein spezifisches Minimum,
- ein spezifisches Maximum,
- eine spezifische Anzahl von Strichen und/oder
- einen spezifischen Abstand zwischen Strichen (Delta).

[0090] Man beachte, dass das Festlegen der Anzahl der Striche Delta außer Kraft setzen kann und umgekehrt.

[0091] Der Guide **116** ist dafür verantwortlich, die Werte auf der numerischen Skale auf die passenden Attribute abzubilden. Sowohl der Guide **116** als auch der Scalebuilder **118** sind Abstraktionen. Konkrete Guides **116** sind entweder der Kategorische Guide **120** oder der Stetige Guide **122**, was von den Daten abhängt, und Konkrete Scalebuilder **118** sind Lineare Skale **124**, Logarithmische Skale **126** oder Zeitskale **128**.

[0092] Die Achse **114** und Legende **112** benutzen das Modell, um Bilder zu zeichnen. Der Graphrahmen **134** (von Abb. 11) benutzt auch dieses Modell.

[0093] Die Achse **114** ist eine Legende für positionale Graphdimensionen. Den Achsen sollten nur eine Graphdimension zugeordnet werden und sie sollte den Typ POSITIONAL haben. Achsen verwenden nicht das GPL-Symbol für die Strichzeichen, sondern sie verwenden Instanzen der GPL-Linie (sodass Transformationen korrekt funktionieren). Achsen berücksichtigen auch Nebenstriche und ein Lineal. Die Nebenstriche werden unter Verwendung eines Scalebuilders bestimmt, um eine Skale zwischen den Hauptstrichen aufzubauen. Sowohl das Lineal als auch die Striche (und die Nebenstriche) haben ihre eigenen Instanzen der Ästhetik.

[0094] Die Legende **112** ist ein Bild von einer oder mehreren Graphdimensionen **100**. Die Legende **112** nimmt die Information von der Graphdimension **100** und erzeugt ein Bild unter Verwendung von Objekten der Primitiven Graphschnittstelle **34** in einer Renderer-Schnittstelle **152** (von Abb. 12). Legenden werden aus Segmenten zusammengesetzt, die den Werten auf den Legenden entsprechen (diese sind Kategorien in einem kategorischen Fall). Jedes Segment hat vier Stücke: den Hauptstrich, das Strichlabel, die Nebenstriche und das

Lineal. In einer alternative Ausführungsart können Legenden an der Nord-, Süd-, Ost- oder Westseite einen Label haben und können vertikal oder horizontal ausgerichtet werden. Die einzelnen Legendenelemente umfassen Text, der mit einem Symbol assoziiert ist. Die Strings werden als „Strichlabel" angesehen, und die Symbole werden als „Striche" angesehen. Das Standardsymbol ist ein Symbol **170**, kann aber auf jeden Symboltyp eingestellt werden – z. B. Polygon **154**, Rechteck **164**, Kreis **172**; dies kann jedoch außer Kraft gesetzt werden, wenn die Graph-Dimension **100** den Typ FORM (SHAPE) hat. Symbole sind einfach Formen, die eine Position und eine Größe haben. Die in dieser Schnittstelle definierten Konstanten beschreiben, wie das Symbol aussehen sollte.

[0095] Die Legende **112** funktioniert durch Aufsuchen der entsprechenden Attribute für einen von einer Graphdimension **100** ermittelten gegebenen Skalenwert und die Benutzung des Attributs, um die Ästhetik **38** des Strichs (d. h. des Symbols) zu modifizieren. Die Ästhetikklasse **38** enthält alle Ästhetikattribute zum Zeichnen eines primitiven Graphen. Die Attribute können visuell oder, wie beispielsweise Sound, nichtvisuell sein. Die Renderer-Schnittstelle **152** (von Abb. 12) wendet die Ästhetik **38** an, wenn der primitive Graph gezeichnet wird.

[0096] Der Strichlabel wird auch ermittelt, indem der Skalenwert auf der Graphdimension **100** aufgesucht wird. Viele Optionen steuern das Aussehen von Legende **112** und Layout; diese werden durch das in Abb. 11 gezeigte Rahmenlayout-Objekt **42** des Graphrahmens festgelegt.

[0097] Die Legenden **112** registrieren sich bei einer Renderer-Schnittstelle **152**, um das Pointer-Ereignis **184** (in Abb. 12 gezeigt) zu empfangen. Nach Empfang eines Pointer- Ereignisses **184**, wird die Legende **112** bestimmen, auf welchen Teil der Legende **112** zugegriffen wurde und wird dann ein Legendenzugriff-Ereignis **130** an die Legendenzugriff-Lauscher-Schnittstelle **132** (in Abb. 10 gezeigt) abfeuern. Die Legende **112** wird Legendenzugriff-Lauscher **132** über eins von vier Verfahren benachrichtigen, je nachdem auf welchen Teil zugegriffen wurde: einen Strich, einen Strichlabel, den Legendenlabel oder einen anderen Teil der Legende **112**. Legendenzugriff-Ereignis **130** erbt vom Pointer-Ereignis **184**, das vom Mouse-Ereignis **186** erbt, sodass die gesamte mit dem Zugriff assoziierte Information präserviert ist.

[0098] Das Legendenzugriff-Ereignis **130** enthält Information bezüglich sowohl der Position des Zugriffs als auch der Werte auf Legende **112**, die dem Zugriff entsprechen, wenn solche Werte existieren (d. h. wenn auf einen Strich oder Strichlabel zugegriffen wurde). Die Größe des „Werte"-Arrays wird der Anzahl der Graphdimensionen **100** auf der Legende **112** gleich sein.

[0099] Der einzige Unterschied zwischen Legende **112** und Achse **114** besteht darin, wie die beiden gezeichnet sind. Da Legende **112** und Achse **114** jeweils ihren eigenen Scalebuilder **118** haben, erbt die Achse **114** von der Legende **112**. Der Scalebuilder **118** erzeugt eine unabhängige Skale für die Achse **114**, sodass sie ohne Modellwechsel angepasst werden kann. Die Achse **114** fügt auch Nebenstriche und ein Lineal hinzu und redefiniert die Hauptstriche nicht als Symbole **170**, sondern als Linien **156**. Die Nebenstriche werden unter Verwendung eines Scalebuilder-Objekts **118** erzeugt, das zwischen zwei Hauptstrichen eine Skale in einem Teil der Achse **114** aufbaut und für jeden Wert auf der Skale einen Nebenstrich zeichnet. So hängt die Positionierung von Nebenstrichen von der bestimmten Skale ab, die zum Erzeugen der Striche benutzt wird – z. B. Lineare Skale **124**, Logarithmische Skale **126** oder Zeitskale **128**. Wie im Fall der Legende **112** kann die Achse **114** zwischen Datenkoordinaten und Renderer-Koordinaten übersetzen.

[0100] Abb. 11 ist ein Objektdiagramm des Rahmen- und Layoutaspekts der vorliegenden Erfindung. Das Design für Rahmen wird in Modell und Ansicht aufgeteilt.

- Das Modell, Rahmenmodell **94** (in Abb. 5 gezeigt), ist eine Softwarekomponente, die die Struktur von Graph **10** bereitstellt und Graphdimensionen **100** (von Abb. 10), Dimensionen für Elemente **46** und Daten **12** für Element **46** enthält.
- Die Ansicht, Graphrahmen **134**, ist eine Komponente, die ein Bild des Modells durch Aufbau von Graphen aus dem Baum **98** von im Rahmenmodell **94** enthaltenen Dimensionen bereitstellt.

[0101] Rahmenmodell **94** und Graphrahmen **134** kommunizieren über Rahmenmodelländerungs-Ereignisse **102** (in Abb. 5 gezeigt). Rahmenmodell **94** strahlt auch Rahmenmodelländerungs-Ereignisse **102** an die Rahmenmodelländerungs-Lauscher-Schnittstelle **104** aus.

[0102] Die Rahmenlayout-Schnittstelle **42** (von Abb. 2) stellt detaillierte Steuerung über das Aussehen und den Eindruck eines Graphen bereit. Die abstrakte Klasse Basislayout **142** stellt Setter/Getter-Verfahren für viele Eigenschaften bereit, wie beispielsweise Schriftarten, Farben, Positionierungen, wo Striche die Achse **114**

schneiden und viele andere. Die Rahmenlayout-Schnittstelle **42** funktioniert, indem sie API-Aufrufe auf gpl.graph-Objekten macht, während sie aufgebaut werden. Die Rahmenlayout-Schnittstelle **42** erhält zwei Gelegenheiten, den Graphrahmen **134** zu ändern: die erste, bevor die Komponenten eines Graphrahmens **134** aufgebaut worden sind; und danach die zweite. Der Graphrahmen **134** umfasst seine Elemente **46** (einschließlich des Rahmenelements **136**), seine Legende **112** und seine Achsen **114**.

[0103] Das Basislayout **142** ist eine Rahmenlayout-Schnittstelle **42**, die die Verfahren zur elementaren Steuerung des Aussehens und Ausdrucks der Graphrahmen **134** bereitstellt. Das Basislayout **142** stellt nur die Standardwerte bereit; die Unterklassen modifizieren die panelierten Graphen.

[0104] Konkrete Rahmenlayouts können Größen, Beabstandung und Sichtbarkeit modifizieren, um ein bestimmtes Aussehen zu erzielen. Wie in Abb. 11 gezeigt ist, werden vier konkrete Layouts bereitgestellt: Zeilenplotlayout **144**, Trellislayout **146**, Tortendiagrammlayout **148** und Standardlayout **150**. Jedes dieser Layouts ist eine Rahmenlayout-Schnittstelle **42** und wird unten zusammengefasst:

- Das Zeilenplotlayout **144** erzeugt einen Graphen mit „Zeilenplot"-Aussehen und Ausdruck. Einige Charakteristiken dieses Aussehens und Ausdrucks schließen ein:
  a) grauen Hintergrund mit weißem Gitternetz,
  b) weißen Raum zwischen Panelen und
  c) alternierende X-Achsen-Positionen zwischen Panelen.
- Das Trellislayout **146** basiert auf den „Trellis"-Anzeigen aus „Visualizing Data" von W.S. Cleveland und erzeugt einen Graphen mit einem „Trellis"-Aussehen und Eindruck. Einige Charakteristiken dieses Aussehens und Eindrucks schließen ein:
  a) weißen Hintergrund und hellgraues Gitternetz,
  b) Streifenlabel auf jedem Tochterrahmen,
  c) weißen Raum zwischen Panelen und
  d) alternierende X-Achsen-Positionen zwischen Panelen: alternierend zwischen Kopf des gesamten Graphen und Fuß des gesamten Graphen.
- Das Standardlayout **150** ähnelt dem Zeilenplotlayout **144**, versucht aber nicht, Achsenlabel in panelierten Graphen zu unterdrücken. Standardlayout **150** wird benutzt, wenn der Graphrahmen **134** kein Layout spezifiziert.
- Das Tortendiagrammlayout **148** kann benutzt werden, um das Standardlayout **150** zu erweitern.

[0105] Wie oben beschrieben ist, ist der Graphrahmen **134** ein Bild eines Rahmenmodells **94**. Der Graphrahmen **134** zeichnet eine Hintergrundbox, Gitternetz, einen Kopftitel und einen Fußtitel (Fußnote) und schließt Achsen **114**, eine Legende **112** und null oder mehrere Elemente **46** ein.

[0106] Der Graphrahmen **134** benutzt die im Baum **98** des Rahmenmodells **94** erzeugten Dimensionen, um die Achsen **114** zu erzeugen. Die Anzahl der erzeugten Achsen **114** basiert auf dem besonderen Koordinatensystem **138**, das für den Graphrahmen **134** installiert ist. Die Transformation **36** (in Abb. 1 gezeigt) muss den Graphrahmen **134** mit einem Koordinatensystem **138** versorgen, das für die bestimmte Transformation **36** angemessen ist. Das Standard-Koordinatensystem **138** ist das 2D-Koordinatensystem **140**. Die Achsen **114** sind skaliert unter Verwendung der vom Baum **98** erhaltenen POSITIONS-Dimensionen. Eine beliebige Anzahl von Element-Objekten **46** benutzen die Achsen **114** innerhalb des Rahmens, um sich selbst zu zeichnen. Elementobjekte **46** können dem Graphrahmen **134** nichtpositionelle Dimensionen hinzufügen, wodurch bewirkt wird, dass er eine Legende **112** zeichnet. Pro Graphrahmen **134** wird nur eine Legende **112** gezeichnet.

[0107] Der Graphrahmen **134** hat auch ein „Quaderhülle" genanntes Hintergrundpanel. Die Ästhetik **38** der Quaderhülle ist die „Standard-Ästhetik" eines Graphrahmens **134**.

[0108] Der Graphrahmen **134** hat auch sowohl ein Gitternetz als auch zwei Titelmengen für die Labelerzeugung. Das Gitternetz hat sein eigenes Ästhetikobjekt **38**, das dazu benutzt werden kann, das Aussehen des Gitternetzes zu ändern. Ein „Titel" erscheint am Kopf des Graphrahmens **134** und der zweite „Fußtitel" erscheint am Fuß (wie eine Fußnote). Der Kopftitel kann ein String **44** (von Abb. 2) sein oder ein Textarray **166**, oder er kann zum Ziehen-und-Ablegen verschiedener Elementobjekte **46** zwischen Rahmen an die Legende **112** gesetzt werden.

[0109] Innerhalb des Graphrahmens **134** zeichnen die Elemente **46** eine Datenrepräsentation. Alle Elemente **46** operieren unabhängig von einander, und es können viele verschiedene Elemente **46** dem Graphrahmen **134** hinzugefügt werden. Jedes Element **46** hat seine eigene Elementspezifikation **96** (von Abb. 5), die dem Rahmenmodell **94** hinzugefügt wird. Nachdem die Datenansicht-Schnittstelle **14** das Rahmenmodell **94** verar-

beitet, wird die Item-Rahmen-Schnittstelle **18** (in Abb. 1 gezeigt) die Daten für die Elemente **46** enthalten – d. h. die zum Zeichnen der Elemente erforderliche Geometrie. Beispielsweise wird die Item-Rahmen-Schnittstelle **18** für ein auf Mittelwerten basierendes Balkendiagramm nur die zu zeichnenden Mittelwerte enthalten. Die Item-Rahmen-Schnittstelle **18** für ein Streuungsdiagramm wird alle Rohwerte für das Streuungsdiagramm enthalten. Die Item-Rahmen-Schnittstelle **18** für ein glätteres Diagramm wird alle x,y-Paare für das glättere Diagramm enthalten.

[0110] Je nach Konstruktion des Baums **98** kann der Graphrahmen **134** ein Rahmenelement **136** zum Panelieren erzeugen. Verlangt die Struktur des Baums **98** mehr POSITIONS-Dimensionen als der Graphrahmen **134** handhaben kann, dann wird der Graphrahmen **134** ein Rahmenelement **136** benutzen und es an den Baum **98** weitergeben. Das Rahmenelement **136** wird dann den Baum **98** untersuchen, um passende „Tochter"-Graphrahmen im Inneren des ursprünglichen Graphrahmens **134** zu erzeugen und zu platzieren. Jeder „Tochter"-Graphrahmen erhält seinen eigenen Baum **98**, und der Prozess kann sich wiederholen. Die zum Mutter-Graphrahmen **134** hinzugefügten Elemente **46** werden wie eine biologische Zelle „geteilt" – d. h. das Element **46** teilt sich in und enthält n Klone seiner selbst – und jeder Klon wird in einen Graphrahmen **134** platziert.

[0111] Die Glyphen-Schnittstelle **32** (in Abb. 1 gezeigt) kann bewirken, dass eine Transformation **36** bei einem Graphrahmen **134** „gesetzt" oder „registriert" wird. Wenn sie bei einem Graphrahmen **134** registriert ist, beeinflusst die Transformation **36** die Form des Rahmens, der Gitterlinie, der Achsen **114** und der Elemente **46**. Eine beim Graphrahmen **134** registrierte Polartransformation würde beispielsweise bewirken, dass der Graph kreisförmig wäre, würde aber den Titel oder die Legende **112** nicht beeinflussen. Eine beim Graphrahmen **134** registrierte Ähnlichkeitstransformation würde den gesamten Graphen dimensionieren oder auf der Anzeige bewegen.

[0112] Alles Gezeichnete muss die Glyphen-Schnittstelle **32** (von Abb. 1) implementieren. Glyphen zeichnen sich oder bauen sich auf durch Erzeugen von Objekten einer Primitiven Graphschnittstelle **34** innerhalb einer Instanz der Renderer- Schnittstelle **152**. Die Glyphen-Schnittstelle **32** kann dann das Bild durch Manipulation dieser Objekte ändern. rebuild()-Aufrufe bewirken das Löschen und das Rekonstruieren dieser Objekte, während Renderer.refresh()-Aufrufe das Screenbild aktualisieren, wobei irgendwelche durch die Glyphe **32** an Objekten der Primitiven Graphschnittstelle **34** ausgeführten Änderungen reflektiert werden.

[0113] Einzelne Glyphen **32** sind für die Dokumentation verantwortlich, wenn ein rebuild() erforderlich ist. Obwohl die Glyphen **32** auf jede Schicht der Renderer-Schnittstelle **152** gesetzt werden können (die Schichtung bestimmt die Zeichenordnung), können sie ihre Mutter oder ihren Ursprung zurückverfolgen. Beispielsweise wäre die Mutter einer Legende **112** ein Graphrahmen **134**.

[0114] Als Standard wird der Graphrahmen **134** den gesamten Raum in einer Renderer-Schnittstelle **152** einnehmen. Wenn also beim Graphrahmen **134** keine Skaliertransformation **36** registriert ist, dann wird der Graphrahmen **134** das gesamte Fenster ausfüllen, sodass die Achsen **114**, Legenden **112** und Titel nicht sichtbar wären. Aus diesem Grund sollte eine Skaliertransformation **36** – wie ein Affine 2D-Transformation – die den Graphrahmen **134** um etwas unter 1,0 skaliert, beim Graphrahmen **134** registriert sein.

[0115] Der Graphrahmen **134** ist auch ein Mouse-Lauscher. Wenn der Graphrahmen **134** ein Pointer-Ereignis **184** (in Abb. 12 gezeigt) von einer Renderer-Schnittstelle **152** hört und erkennt, dass die zugegriffene Primitive Graph-Schnittstelle **34** Teil eines der Folgenden war: Hintergrund, Gitternetz oder Titel des Graphrahmens **134**, dann wird der Graphrahmen **134** ein Glyphenzugriff-Ereignis **194** an interessierte Lauscher abfeuern, wodurch sie informiert werden, dass auf einen Rahmen zugegriffen wurde.

[0116] Abb. 12 ist ein Objektdiagramm des Anzeigers der vorliegenden Erfindung. Die Renderer-Schnittstelle **152** ist dazu ausgelegt, von den Zeichenwerkzeugen des Systems unabhängig zu sein. In der bevorzugten Ausführungsart werden Implementierungen der Renderer-Schnittstelle **152** mit Standard-JDK 1.1-Graphik, Java2D und Java3D arbeiten. Diese Unabhängigkeit wird dadurch erzielt, dass primitive graphzeichnende Objekte als Schnittstellen bereitgestellt werden; dazu gehören beispielsweise Polygon **154**, Linien **156**, Symbollinie **158**, Linie **160**, Bild **162**, Rechteck **164**, Text **166**, wovon Textbeschreibung **168** ein Teil ist, Symbol **170**, Kreis **172**, Hexagon **174**, Ellipse **175**, Schicht **177** (nicht gezeigt), Symbollinie **179** (nicht gezeigt) und Gleichseitiges Dreieck **176**. Primitive sind die elementaren Zeichenformen in der GPL. Jede konkrete Renderer-Schnittstelle **152** muss eine Fabrikmethode, createPrimitive() genannt, bereitstellen, um ein konkretes primitives Graphobjekt (das Objekt sollte aus einer inneren Klasse kommen) zu instantiieren. Die Renderer-Schnittstelle **152** legt ein quadratisches Renderer.MIN bis Renderer.MAX-Koordinatensystem über das Zeichenfenster des Systems. Die Werte nehmen von links nach rechts und von unten nach oben zu. Alle Koordi-

naten zum Zeichnen unterliegen dieser Skale. Die Renderer-Schnittstelle **152** verfolgt auch die Schichtung, von der die Zeichenordnung bestimmt wird. So sollten beispielsweise Objekte in Schicht **2** Objekte in Schicht **1** „überlagern".

[0117] Alle zeichnenden Primitive Graphschnittstellen erstrecken sich von der Primitiven Graphschnittstelle **34**, die Verfahrenssignaturen zum Abfragen/Ändern der Objekt-Ästhetik **38** (ein Objekt, das Farben, Druckformate, Sichtbarkeit usw. des Primitivs beschreibt), Schichtung und Handhabung der Transformationen enthält. Die Primitive Graphschnittstelle **34** erweitert die Assoziierte Datenschnittstelle **56**, sodass Metadaten in ein beliebiges Zeichenobjekt eincodiert werden können. Die abstrakte Klasse für Primitivunterstützung **178** ist eine Implementierung der Primitiven Graphsschnittstelle **34**, die die meisten der erforderlichen Verfahren handhabt, aber vom Renderer abhängige Verfahren als abstrakt bestehen lässt. Es wird auch eine abstrakte Klasse für Primitive **180** bereitgestellt, die eine als einzelne Schnittstelle manipulierbare Kollektion von Primitiven Graphschnittstellen **34** ist.

[0118] Die Primitiven Graphschnittstellen **34** sind in der vorliegenden Erfindung elementare Zeichenformen. Alle Zeichenprimitive sollten diese Schnittstelle implementieren. Primitive Graphschnittstellen **34** sind erforderlich zum Pflegen und/oder Bereitstellen: (a) ihres Ästhetikobjekts **38**, (b) der Schicht, in der die Renderer-Schnittstelle **152** sie zeichnen sollte, unabhängig davon, ob die Primitiven Graphschnittstellen **34** transformierbar sind oder nicht, und (c) einer Konstanten, wie sie in der Primitiven Graphschnittstelle **34** definiert ist, die ihren Typ beschreibt. Alle Primitiven Graphschnittstellen **34** sind auch Transformationslauscher-Schnittstellen **182** und sollte wissen, wie sie sich transformieren, wenn sie von einem Transformationsobjekt **36** dazu angewiesen werden.

[0119] Die Primitivunterstützung **178** stellt eine generische Implementierung der Primitiven Graphschnittstelle **34** bereit. Die Primitivunterstützung **178** kann zur Unterklasse gemacht oder direkt benutzt werden; sie stellt Unterstützung zur Pflege der Ästhetik **38** bereit, unabhängig davon, ob die Primitive Graphschnittstelle **34** transformierbar ist oder nicht. Die Primitivunterstützung **178** stellt auch Unterstützung zur Transformation eines Basisprimitivs bereit, indem sie die Quelltransformation anweist, „dieses" Primitiv zu transformieren.

[0120] Die abstrakte Klasse für Primitive **180** ist eine Kollektion von Primitiven, die alle als eine einzelne Primitive Graphschnittstelle **34** behandelt werden. Werden beispielsweise eine Kreisschnittstelle **172** und eine Rechteckschnittstelle **164** zu einer abstrakten Klasse für Primitive **180** hinzugefügt, dann kann die Ästhetik **38** für beide Schnittstellen mit einem Aufruf an das Primitives.setAesthetics()-Verfahren geändert werden. Dieselbe Idee gilt für Schichten und Transformationen **36**. Die innerhalb dieser abstrakten Klasse für Primitive **180** enthaltenen Primitiven können über setClip() geclippt werden.

[0121] Wenn immer ein Mouse-Ereignis **186** eintritt, erzeugt die Renderer-Schnittstelle **152** ein Pointer-Ereignis **184** aus dem Mouse-Ereignis **186** und feuert es an Mouse-Lauscher und Mousebewegungs-Lauscher ab. Würde auf eine Primitive Graphschnittstelle **34** zugegriffen, dann enthielte das Pointer-Ereignis **184** eine Referenz auf die Primitive Graphschnittstelle **34**, auf die zugegriffen wurde. Der Unterschied zwischen einem Pointer-Ereignis **184** und einem Mouse-Ereignis **186** besteht darin, dass das Pointer-Ereignis **184** ein Koordinatenobjekt (nicht gezeigt) enthält, das die Position des Ereignisses durch Renderer.MIN und Renderer.Max definiert. Das Pointer-Ereignis **184** kann auch ein Objekt Primitive Graphschnittstelle **34** enthalten, wenn auf ein solches zugegriffen würde.

[0122] Im Displayer ist eine Softwarekomponente, Java Canvas Renderer **188**, enthalten. Der Java Canvas Renderer **188** ist eine Implementierung der Renderer-Schnittstelle **152** unter Verwendung von Standard-JDK 1.1-java.awt.Graphics auf einem java.awt.Canvas-Objekt.

[0123] Die auch in **Abb. 12** gezeigten Transformationen **36** benutzen ihre jeweilige Mathematik, um Koordinaten in andere Koordinaten umzuwandeln und die abstrakte Klasse für Primitive **180** direkt zu modifizieren – d. h. zu transformieren. Eine Klasse der Transformationen **36** ist die 2D-Transfonnation **192**. Die Primitive Graphschnittstelle **34** erweitert die Transformationslauscher-Schnittstelle **182**, sodass alle primitiven Objekte Transformationsobjekte belauschen können. Nach Empfang eines Transformations-Ereignisses **190** über das transform()-Verfahren fragt die abstrakte Primitiven-Klasse **180** die Ereignisquelle ab – die das Transformationsobjekt **36** ist, von dem das Ereignis abgefeuert wurde – und benutzt es, um sich selbst zu transformieren. Die abstrakte Klasse für Primitivunterstützung **178** handhabt eine Implementierung davon, und der Client steuert, ob eine abstrakte Klasse für Primitive **180** „transformiert" wird oder nicht. Alle abstrakten Klassen für Primitive **180** können setTransformable() auf wahr oder falsch setzen. Ein als „transformierbar" gekennzeichnetes Objekt der abstrakten Klasse für Primitive **180** kann nach einer Transformation **36** ein völlig verschiedenes

Aussehen haben, während bei nicht transformierbaren primitiven Objekte nur deren Position geändert werden kann. Beispielsweise würden die Mittelpunktkoordinaten einer nicht transformierbaren Rechteck-Schnittstelle **164** geändert, aber die Rechteckform bliebe erhalten. Demgegenüber könnte eine transformierbare Rechteck-Schnittstelle **164** zu etwas werden, das nicht wie ein Rechteck aussieht.

[0124] Die Transformationen **36** stellen auch leere Koordinaten-Objekte bereit, die dem Koordinatensystem der eigentlichen Transformation **36** entsprechen, die die korrekte Anzahl von Dimensionen hat. Beispielsweise stellt eine polare Transformation Koordinaten bereit, die zwei Werte (r und Theta) handhaben können, während eine sphärische Transformation Koordinaten mit drei Werten (r, Phi und Theta) bereitstellt. Die Mathematikklasse – eine aus statischen, finalen Verfahren bestehende Klasse, die unter Verwendung von Funktionen wie Kosinus, Logarithmus, Sinus und Quadratwurzel ein Doublesarray mathematisch verarbeitet.

[0125] Die Transformationen **36** arbeiten, indem sie Transformations-Ereignisse **190** an interessierte abstrakte Klassen für Primitive **180** abfeuern. Die Lauscher werden dann aktiv, um die Transformation **36**, die die Quelle des Ereignisses ist, zu veranlassen, auf ihnen zu operieren. Das Ereignis-Modell, wie es hier benutzt wird, nimmt Transformationen **36** auf, die sich auf Mousebewegungen (wie beispielsweise Fischaugentransformation) verlassen können oder ein UI-Werkzeug, das Änderungen in der Transformation **36** bewirkt.

[0126] Abb. 13 ist ein Objektdiagramm, das die Software-Wechselwirkungen abbildet, die auftreten, wenn der Benutzer ein Item auswählt. Benutzeraktionen werden über verschiedene Ereignisse gehandhabt, einschließlich eines der Folgenden: Mouse-Ereignis **186**, Pointer-Ereignis **184**, Glyphenzugriff-Ereignis **194** oder eine der Unterklassen des Glyphenzugriff-Ereignisses **194** – d. h. Legendenzugriff-Ereignis **130**, Rahmenzugriff-Ereignis **196** oder Elementzugriff-Ereignis **198**.

[0127] Auf einer niedrigeren Ebene spricht die konkrete Renderer-Schnittstelle **152** (in Abb. 12 gezeigt) die Systemsprache für das Rendern und Detektieren von Zugriffen. Sie bestimmt, auf welches geometrische Objekt – z. B. Kreis **172** und Rechteck **164** (beide von Abb. 12) „zugegriffen" wurde und feuert als Reaktion ein Pointer-Ereignis **184** ab, eine Unterklasse des Mouse-Ereignisses **186**. Element **46** (von Abb. 2), Graphrahmen **134** (von Abb. 11), Achse **114** (von Abb. 10) und Legende **112** (von Abb. 10) lauschen auf das Pointer-Ereignis **184**, um zu bestimmen, ob auf ihre geometrische Form zugegriffen wurde. Wurde auf ihre jeweilige geometrische Form zugegriffen, dann feuern sie eine Instanz des Glyphenzugriff-Ereignisses **194** ab, die das Graphikobjekt enthält – oder die Glyphen-Schnittstelle **32** (wie Punkt, Legende **112** u. Ä.), auf die zugegriffen wurde. Controller (siehe Abb. 14 und Abb. 15) oder andere Lauscher lauschen auf Glyphenzugriff-Ereignisse **194** und starten die Aktion, für deren Ausführung der betreffende Controller ausgelegt war, indem sie direkt auf dem Graphrahmen **134** operieren. Das Pointer-Ereignis **184** kann auch eine Instanz der Primitiven Graphschnittstelle **34** abfeuern.

[0128] Abb. 4 zeigt den Programmserver. Diese Abbildung gilt, wo das erfindungsgemäße Verfahren und System über ein typisches Client-Server-Netz benutzt werden. Die in Abb. 4 gezeigte Datenansicht-Serverschnittstelle **86** verteilt Zugriff/Manipulation der Daten zum Erzeugen von Graphen. Die Datenansicht-Schnittstelle **14** erweitert java.rmi.Remote, sodass alle Datenansicht-Objekte verteilt werden können. Es gibt eine konkrete Datenansicht-Schnittstelle **14** auf der Datenansicht-Serverschnittstelle **86** für jeden an den Server angeschlossenen Client. Die Datenansicht-Serverschnittstelle **86** verwaltet die einzelnen, entfernten Datenansicht-Schnittstellen **14**, indem sie diese aktiviert/passiviert und gleiche Informationen zwischen Datenansicht-Schnittstellen **14** teilt. Das Clientprogramm muss ein Objekt der Entfernten Datenansicht **88** als seine Datenansicht-Schnittstelle **14** benutzen.

[0129] Die Entfernte Datenansicht **88** führt alle notwendigen Kommunikationen mit der Datenansicht-Serverschnittstelle **86** aus, und die Entfernte Datenansicht **88** ist auch ein Datendurchlauflauscher **62** (von Abb. 3), um Rückrufmeldungen zu empfangen, wenn der Datendurchlauf auf dem entfernten Objekt abgeschlossen ist. Die Kommunikation läuft wie folgt ab:

- Ein Client ruft dataview.beginDataPass() auf der clientseitigen Entfernten Datenansicht **88** auf.
- Die Entfernte Datenansicht **88** fordert vom Server ein Handle für ihre Datenansicht-Schnittstelle **14** an.
- Der Server weckt die korrekte Datenansicht-Schnittstelle **14** und sendet ein Handle an sie zurück.
- Die Entfernte Datenansicht **88** registriert sich bei dieser Datenansicht-Schnittstelle **14** als eine Datendurchlauflauscher-Schnittstelle **62**, falls sie nicht schon registriert ist.
- Die Entfernte Datenansicht **88** ruft dataview.beginDataPass() auf der serverseitigen Datenansicht-Schnittstelle **14** auf.
- Die serverseitige Datenansicht-Schnittstelle **14** ruft aboutToDataPass() auf.
- Dies wird von der Entfernten Datenansicht **88** empfangen und an ihre clientseitigen Lauscher weitergelei-

tet.
• Rufe von der Entfernten Datenansicht **88** clientseitigen Datendurchlauflauscher-Schnittstelle **62** werden an die serverseitige Datenansicht-Schnittstelle **14** weitergeleitet.
• Der Datendurchlauf läuft ab (serverseitig).
• Die serverseitige Datenansicht-Schnittstelle **14** ruft finishedDataPass() auf.
• Dies wird clientseitig von der Entfernten Datenansicht **88** empfangen und an Lauscher weitergeleitet.
• Vergeht eine festgesetzte Anzahl von Millisekunden ohne Unterbrechung, dann fordert die Entfernte Datenansicht **88** die Datenansicht-Serverschnittstelle **86** auf, die serverseitige Datenansicht-Schnittstelle **14** zu passivieren.

[0130] Der Server für Abstrakte Datenansicht **90** stellt eine Implementierung der Datenansicht-Schnittstelle **86** bereit, überlässt aber Unterklassen das tatsächliche Schreiben und Lesen zum Aktivieren/Passivieren. Der Server für dateibasierte Datenansicht **92** liest von und schreibt auf Dateien auf Platte.

[0131] Abb. 14 bildet die Controller-Schnittstelle **40** und die verschiedenen Controller ab, die durch die Controller-Schnittstelle **40** arbeiten. Die implementierenden Schnittstellen der Controller-Schnittstelle **40** sind der Metadatenanzeige-Controller **200**, der Kategorieordnungs-Controller **202**, der Vergleichslineal-Controller **204**, der Elementbewegungs-Controller **206**, der Elementfilter-Controller **208** (der eine abstrakte Klasse ist) und der Schwenk-und-Zoom-2D-Controller **210**.

[0132] Der Metadatenanzeige-Controller **200** zeigt von einem Element **46** (von Abb. 2) empfangene Metadaten, ein Legendenlabel oder ein Legendenstrichlabel an.

[0133] Der Kategorieordnungs-Controller **202** steuert die Ordnung, in der die Kategorien angezeigt werden. Der Vergleichslineal-Controller **204** benutzt Metadaten, um Endbenutzer vor ungültigen Vergleichen in einem Graphen **10** zu warnen. Der Elementbewegungs-Controller **206** bewegt Elemente **46** zwischen Rahmen in einem panelierten Graphen **10**. Die abstrakte Klasse Elementfilter-Controller **208** steuert, ob einzelne Fälle (Zeilen) in einem Graphen **10** enthalten sind oder nicht. Nicht in einem Graphen enthaltene Fälle sind nicht sichtbar und nicht mit Funktionen involviert, die ein Element **46** benutzen kann.

[0134] Fünf verschiedene Filtertypen sind verfügbar: Bereichs-Slidefilter-Controller **212**, Bereichs-2D-Slidefilter-Controller **214**, Slidefilter-Controller **216**, Pickerfilter-Controller **218** und Textsuchefilter-Controller **220**. Jedes dieser fünf Filter kann in die Elementfilter-Controllergruppe **222** eingesetzt werden, die als ein einzelnes Filter arbeitet.

[0135] Der Slidefilter-Controller **216** filtert einen einzelnen Punkt der Daten **12** und zeigt nur den ausgewählten Datenpunkt. Der Pickerfilter-Controller **218** arbeitet nur mit den gesamten Kategorien einer Menge der Variablen **22**. Selektion/Deselektion einer kategorischen Variable durch den Pickerfilter-Controller **218** bewirkt, dass die gesamte Kategorie gefiltert oder nicht gefiltert wird. Der Bereichs-Slidefilter-Controller **212** filtert einen Datenbereich, und es werden nur die Daten innerhalb des ausgewählten Bereichs gezeigt; umgekehrt können alle Daten gezeigt werden, die nicht im ausgewählten Bereich liegen. Der Bereichs-2D-Slidefilter-Controller **214** filtert gleichzeitig einen Datenbereich in zwei Dimensionen.

[0136] Der Schwenk-und-Zoom-2D-Controller **210** ist ein Graph-Navigationswerkzeug. Er ermöglicht einem Benutzer, in einem Graphen zu „schwenken" und zu „zoomen". Der Schwenk-und-Zoom-2D-Controller **210** und Elementfilter-Controller **208** sind Unterklassen des Panels **224**.

[0137] Abb. 15 zeigt die Relation zwischen der Controller-Schnittstelle **40** und den Builder-Controllern. Builder-Controller sind Module, die benutzt werden, um ein UI zur Erzeugung von Graphen zu konstruieren. Die Builder-Controller implementieren Schnittstellen der Controller-Schnittstelle **40** und schließen Folgendes ein: den Graphdrehpunkt-Controller **226**, den Legendenattribut-Controller **228**, den Achsenfunktions-Controller **230**, den Rahmenlayout-Controller **232**, den Elementwerkzeugleisten-Controller **234**, den Elementeigenschaft-Controller **236** und den Variablenlisten-Controller **238**. Davon sind die letzten vier – d. h. Rahmenlayout-Controller **232**, Elementwerkzeugleisten-Controller **234**, Elementeigenschaft-Controller **236** und Variablenlisten-Controller **238** – Unterklassen von JPanel **240**.

[0138] Der Graphdrehpunkt-Controller **226** stellt Drehfähigkeiten für Graphen bereit. Achsen **114** können miteinander gedreht werden, indem eine Menge von Variablen **22** (von Abb. 1) von einer Achse **114** zur anderen gezogen wird. Der Legendenattribut-Controller **228** stellt direkte Steuerung über die Abbildung von Attributen auf Daten bereit.

[0139] Der Achsenfunktions-Controller **230** stellt Steuerung der auf einer bestimmten Achse **114** benutzten Funktion bereit. Die Funktionen werden durch Reflexion aus gpl.dataview gewonnen. Funktionen können der Popup-Liste hinzugefügt oder von ihr entfernt werden über die indexierte Eigenschaft setFunctions. Der Achsenfunktions-Controller **230** belauscht den Graphen **10** (von Abb. 1) oder den Graphrahmen **134** (von Abb. 11) auf Legendenzugriff-Ereignisse **130** (von Abb. 10). Der Achsenfunktions-Controller **230** verändert einfach das Objekt Rahmenlayout-Schnittstelle **42** (von Abb. 2) in ein spezifiziertes Layout.

[0140] Der Rahmenlayout-Controller **232** stellt Steuerung über das Layout des Graphrahmens **134** oder des Graphen **10** bereit. Der Rahmenlayout-Controller **232** belauscht den tatsächlichen Graphen **10** oder Graphrahmen **134** auf Eigenschaftsänderungs-Ereignisse.

[0141] Der Elementwerkzeugleisten-Controller **234** stellt Steuerung über Elemente **46** bereit, die im Graph **10** oder Graphrahmen **134** enthalten sind. Die Elemente **46** können aus dem Graphen entfernt werden, indem man auf das Element **46** klickt und es zu einer Element-Werkzeugleiste zurückzieht. Verschiedene Elemente **46** können über die Elemente indexierte Eigenschaft der Element-Werkzeugleiste hinzugefügt und von ihr entfernt werden. Der Elementwerkzeugleisten-Controller **234** belauscht den Graphen **10** und Graphrahmen **134** auf Eigenschaftsänderungs-Ereignisse (nicht gezeigt), Rahmenzugriff-Ereignisse **196** (von Abb. 13) und die tatsächlichen Graphen **10** oder Elemente **46** auf Elementzugriff-Ereignisse **198** (von Abb. 13).

[0142] Der Elementeigenschaft-Controller **236** stellt Steuerung über die Eigenschaften eines bestimmten Elements **46** bereit. Die editierbaren Eigenschaften des Elements **46** werden in einer Tabelle angezeigt und können vom Benutzer modifiziert werden. Der Elementeigenschaft-Controller **236** lauscht auf Datenänderungs-Ereignisse von der Datenansicht-Schnittstelle **14**, Elementzugriff-Ereignisse **198** von Graphen **10** oder Elementen **46** und Eigenschaftsänderungs-Ereignisse von Graphen **10** oder Graphrahmen **134**.

[0143] Der Variablenlisten-Controller **238** stellt Steuerung darüber bereit, welche Variablen **22** in einem Graphen angezeigt werden. Die Variablen **22** können aus dem Graphen entfernt werden. Der Variablenlisten-Controller **238** lauscht auf Datenänderungs-Ereignisse von der Datenansicht-Schnittstelle **14** sowie auf Legendenlabel-Auslösung, Legenden-Auslösung und Eigenschaftsänderungs-Ereignisse (nicht gezeigt) vom Graphrahmen **134** oder Graphen **10**.

[0144] Abb. 16 ist ein Objektdiagramm des Elementepakets der vorliegenden Erfindung. Das Elementepaket enthält die tatsächlichen Elemente (Unterklassen von Element **46**) zum Zeichnen des Graphen **10**.

[0145] Es gibt zwei Hauptklassen der Elemente **46**: Einzeldomänen-Elemente **242** und Elemente **244** mit zusammenhängender Domäne. Der Unterschied zwischen beiden liegt darin, wie die Daten an die Unterklassen Intervall **246** und Punkt **248** der Einzeldomänen-Elemente **242** sowie die Unterklassen Fläche **250** und Linie **252** der Elemente **244** mit zusammenhängender Domäne zugeführt werden. Einzeldomänen-Elemente **242** zeichnen ein Einzelbild für jeden Wert in jeder Domäne und es werden ihnen nur die Daten für die Domäne zugeführt, die gerade gezeichnet wird. Dagegen zeichnen Elemente **244** mit zusammenhängender Domäne zwischen Werte über die gesamte Domäne, weshalb ihnen alle Daten für eine bestimmte Domäne zugeführt werden. Von den Elementen **244** mit zusammenhängender Domäne füllt die Unterklasse Fläche **250** eine Fläche für die Resultate einer über der Domäne zusammenhängenden Funktion; die Unterklasse Linie **252** zeichnet eine Linie für die Resultate einer über der Domäne zusammenhängenden Funktion. Vom Einzeldomänen-Element **242** zeichnet die Unterklasse Intervall **246** eine Form zwischen zwei oder mehreren Koordinaten, wobei die Form des Intervalls vom Wert des Ästhetik-Formparameters abhängt, der durch die Constraints im Punkt definiert ist; der Punkt **248** zeichnet einen einzigen Punkt für jede Koordinate, wobei die Form des Punkts vom Wert der Ästhetik-Formparameter abhängt, die durch Constraints definiert sind, die Folgendes umfassen: Schriftzeichen, Punkt, Horizontalstrich, Hexagon, Bild, Plus, Rechteck, Breitenausdehnung, Dreieck und Vertikalstrich.

[0146] Annotationselemente **254** werden auch bereitgestellt. Annotationselemente **254** dienen Referenzzwecken und basieren nicht auf Daten aus der Datenansicht-Schnittstelle **14**. Sie können alle in einen Graphrahmen **134** (von Abb. 11) platziert werden unter Verwendung von Datenkoordinaten oder Koordinaten (–1 bis 1) aus der Renderer-Schnittstelle **152** (in Abb. 12 gezeigt). Die Unterklassen des Annotationselements **254** sind Referenzlinie **256**, Referenzbild **258**, Referenztext **260** und Referenzfläche **262**, die alle zur Annotation eines Graphen **10** benutzt werden.

[0147] Die finale Unterklasse des Elements **46** ist der Parkettstein **264**.

[0148] Eine andere erfindungsgemäße Ausführungsart ist ein Verfahren zum Erzeugen von quantitativer ästhetischer Graphik aus Daten. Diese Ausführungsart nutzt die Softwarekomponenten des erfindungsgemäßen Systems, wie oben beschrieben ist. Das Verfahren umfasst das Bereitstellen eine Variablenliste. Die Datenansicht-Schnittstelle **14** stellt eine solche Liste bereit. Die Daten **12** werden nach der Indexierung zuerst in Variable Datenstrukturen umgewandelt und dann in Mengen von Variablen **22** durch Anwendung der Algebra **24**, was das Benutzen von mindestens einer der Funktionen Nest, Blend und Cross involviert, die oben beschrieben sind.

[0149] Die Variablen sind wie vorher beschrieben und können stetig sein oder als Alternative kategorisch. Wenn die Variablen kategorisch sind, dann ist die numerische Skale eine ganze Zahl mit einem Wertebereich von null bis zur Anzahl der Kategorien minus eins. Wenn die Variablen stetig sind, dann läuft die numerische Skale vom Minimumwert für die Daten bis zum Maximumwert. Die Datenansicht-Schnittstelle **14** verarbeitet im Rahmenmodell **94** enthaltene Spezifikationen, die ausgeführt werden müssen, um die Elemente **46** in einem Graphen zu zeichnen. In die Spezifikationen eingeschlossen sind Algebraische Spezifikation **48** (von **Abb. 2**), Elementspezifikation **96** (von **Abb. 5**), Variable Transformationsspezifikation **52** (von **Abb. 2**) und Koordinatenspezifikation **50** (von **Abb. 2**), wie oben erörtert ist.

[0150] Der nächste Schritt ist das Bereitstellen einer Liste der Punktrepräsentationen. Solche Repräsentationen können ein jedes der oben beschriebenen Attribute umfassen, definiert durch den Typ der Graphdimension **100**, wie oben erörtert ist. So können die Repräsentationen solche wie beispielsweise die Folgenden umfassen: Position, Größe, Form, Farbe, Farbton, Helligkeit, Sättigung, Rotation, Unschärfe, Transparenz, Textes, Muster, Orientierung oder Körnung. Wie oben beschrieben ist und gemäß der Erfindung, benutzt die Glyphe **32** eine Graphfunktion, um einen Primitiven Graphen **34** aus den Variablen zu erzeugen und bezieht die Variablen auf Dimensionen eines geometrischen Raums, in den der Graph eingebettet wird.

[0151] Ein anderer Schritt des erfindungsgemäßen Verfahrens ist das Bereitstellen einer Liste von Koordinatenpunkten, auf einem Koordinatensystem basierend, wie vorher beschrieben ist. Jedes passende Koordinatensystem kann benutzt werden, einschließlich beispielsweise des kartesischen, rechtwinkligen, polaren, sphärischen oder anderen Systems. Das bestimmte Koordinatensystem **138**, das für den Graphrahmen **134** installiert ist, bestimmt die Anzahl der erzeugten Achsen **114**.

[0152] Das Verfahren umfasst auch das Bereitstellen einer Liste von ästhetischen Repräsentationen. Die Ästhetikklasse **38** enthält alle ästhetischen Attribute zum Zeichnen eines Primitiven Graphen **34**. Diese Attribute können visuell sein oder als Alternative nichtvisuell wie beispielsweise Sound.

[0153] Zusätzliche Schritte des erfindungsgemäßen Verfahrens enthalten das Auswählen von mindestens einer Variablen aus der Variablenliste, wobei mindestens eine Repräsentation für Punkte aus der Liste der Repräsentationen für Punkte ausgewählt wird, mindestens ein Koordinatensystem aus der List der Koordinatensysteme ausgewählt wird und mindestens eine ästhetische Repräsentation aus der Liste der ästhetischen Repräsentationen ausgewählt wird.

[0154] Nachdem die obigen Auswahlen getroffen worden sind, involvieren die nächsten Schritte des erfindungsgemäßen Verfahrens das Bewegen der mindestens einen Variablen an eine vorbestimmte Position und das Anzeigen einer sichtbaren Graphik, die Folgendes repräsentiert: die mindestens eine Variable, die mindestens eine Repräsentation für Punkte, das mindestens eine Koordinatensystem und die mindestens eine ästhetische Repräsentation. Der Graphrahmen **134** zeichnet eine Hintergrundbox, ein Gitternetz, einen Kopftitel, einen Fußtitel und enthält Achsen **114**, eine Legende **112** und Elemente **46**. Die Item-Rahmen-Schnittstelle **18** enthält die zum Zeichnen der Elemente erforderliche Geometrie. Die Transformation **36** ortet einen Punkt im Raum, dessen Koordinatensystem gegeben ist. Unter Verwendung ihrer jeweiligen Mathematik, transformiert die Transformation **36** Koordinaten in andere Koordinaten und modifiziert direkt – d. h. transformiert – die abstrakte Klasse für Primitive **180**, wie oben beschrieben ist. Dadurch transformiert die Transformation **36** den Primitiven Graphen **34** und repräsentieret ihn im ausgewählten Koordinatensystem. Die Transformationen **36** sind auf Transformationen beschränkt, die die funktionale Relation zwischen der Domäne und dem Bereich eines bestimmten Primitiven Graphen **34** aufrechterhalten. Vorzugsweise ist die Transformation **36** beim Graphrahmen **134** registriert. Die Renderer-Schnittstelle **152** (von **Abb. 12**) wendet die Ästhetik **38** an, wenn der Primitive Graph gezeichnet wird. Zeichenobjekte für Primitive Graphen werden bereitgestellt; wie in **Abb. 12** gezeigt ist, sind darin beispielsweise Folgende enthalten: Polygon **154**, Linien **156**, Symbollinie **158**, Linie **160**, Bild **162**, Rechteck **164**, Text **166**, wovon Textbeschreibung **168** ein Teil ist, Symbol **170**, Kreis **172**, Hexagon **174**, Ellipse **175**, Schicht **177** (nicht gezeigt), Symbollinie **179** (nicht gezeigt) und andere Primitive **181** (nicht gezeigt). In der graphischen Repräsentation kann die Punktmenge beispielsweise durch ein Intervall, einen

Punkt, eine Fläche oder eine Linie repräsentiert sein.

[0155] In noch einer anderen erfindungsgemäßen Ausführungsart wird ein Datenverarbeitungssystem bereitgestellt, das Graphen mathematisch aus Daten konstruiert und die Graphen ästhetisch als Graphik repräsentiert. Das System umfasst einen Computerprozessor und einen an die Computerverarbeitung gekoppelten Speicher. Der Speicher enthält einen Satz Computerbefehle zum:

(a) Indexieren der Daten **12**, um eine Datenmenge **16** zu bilden;

(b) Umwandeln der Datenmenge in eine Variable Datenstruktur, die wie folgt definiert ist:

Varset $[X_1, ..., X_n] = [I_m, X_1, ..., X_n, f]$, wo

$X_1, ..., X_n$ n Mengen repräsentiert und eine endliche Menge oder eine Menge reeller Zahlen oder markierter reeller Zahlen ist.

I ist eine Indexmenge {1, 2, ..., m} ist, die über alle natürlichen Zahlen N variiert,

$f: I \rightarrow X_1 \times X_2 \times ... \times X_n$ und f über alle möglichen Funktionen dieser Form variiert.

(c) Umwandeln der Variablen Daten Struktur in eine Menge von Variablen **22** der Form V = varset[X] unter Verwendung von mindestens einer der Funktionen Blend-Funktion, Cross-Funktion und Nest-Funktion;

(d) Abbilden der Menge von Variablen **22** in eine Menge mathematischer Punkte; und

(e) Abbilden der Menge mathematischer Punkte in eine ästhetische Repräsentation.

[0156] Der Speicher kann auch Befehle zum Implementieren der oben beschriebenen verschiedenen Funktionen und Schnittstellen enthalten.

[0157] Eine andere erfindungsgemäße Ausführungsart ist ein nichtflüchtiges Speichermedium, das in einem maschinenlesbaren Format codierte Computersoftware zum Erzeugen von quantitativer Graphik enthält. Das nichtflüchtige Speichermedium enthält: einen Satz von Computerbefehlen zum Indexieren von Daten, um eine Datenmenge zu bilden; einen Satz von Computerbefehlen zum Umwandeln der Datenmenge in eine variable Datenstruktur, wobei die variable Datenstruktur eine Indexmenge, einen Bereich und eine Funktion hat; einen Satz von Computerbefehlen zum Umwandeln der variablen Datenstruktur in eine Menge von Variablen durch Verwendung mindestens einer Funktion, die aus der Folgendes umfassenden Gruppe ausgewählt ist: eine Blend-Funktion, eine Cross-Funktion und eine Nest-Funktion; einen Satz von Computerbefehlen zum Abbilden der Menge der Variablen in eine Punktmenge; und einen Satz von Computerbefehlen zum Abbilden der Punktmenge in eine ästhetische Repräsentation. Das nichtflüchtige Speichermedium kann zusätzlich einen Satz von Befehlen enthalten, die die Variablen zu Dimensionen eines geometrischen Raums in Beziehung setzen, in den die ästhetische Repräsentation eingebettet wird.

[0158] Die oben beschriebenen Merkmale, Funktionen und Ereignisse werden auf ähnliche Weise in diese Ausführungsart eingegliedert.

[0159] Abb. 17 ist ein Objektdiagramm, das die Wechselbeziehung zwischen den folgenden vorher erörterten Softwarekomponenten zeigt:

(a) die Datenansicht-Schnittstelle **14** (in Abb. 3 gezeigt) einschließlich der Entfernten Datenansicht **88** (in Abb. 4 gezeigt) und der JDBC-Datenansicht **66** (in Abb. 3 gezeigt);

(b) die Item-Rahmen-Schnittstelle **18** (in Abb. 1 gezeigt) einschließlich des Rahmenmodells **94** (in Abb. 5 gezeigt), den Baum **98** und die Graphdimensionen **100** umfassend (beide in Abb. 5 gezeigt);

(c) die Controller-Schnittstelle (in Abb. 14 und Abb. 15 gezeigt); und

(d) die Rahmenlayout-Schnittstelle **42** einschließlich Graphrahmen **134**, Legende **112**, Achsen **114** und Glyphen-Schnittstelle **32** (die alle in Abb. 11 gezeigt sind).

[0160] Die oben beschriebenen erfindungsgemäßen Ausführungsarten sind nur als Veranschaulichungen gedacht und sollten nicht als Einschränkung der in den nachfolgenden Patentansprüchen dargelegten Erfindung angesehen werden. Es wird deutlich sein, dass die Erfindung zu einer Vielzahl von Änderungen, Modifikationen und Umgruppierungen befähigt ist und dass beabsichtigt ist, dass solche Änderungen, Modifikationen und Umgruppierungen in den Schutzbereich der nachfolgenden Patentansprüche fallen.

[0161] Elemente und Schritte in den Patentansprüchen wurden nur deshalb numerisch und/oder alphabetisch bezeichnet, um die Lesbarkeit und Verständlichkeit zu fördern. Als solches wird nicht beabsichtigt und soll nicht angenommen werden, dass allein die numerische und/oder alphabetische Bezeichnung die Anordnung der Elemente und/oder Schritte in den Patentansprüchen angibt.

Patentansprüche

1. Verfahren zum Computergenerieren von quantitativer ästhetischer Graphik aus Daten, folgende Schritte umfassend:
Indexieren der Daten, um eine Datenmenge zu bilden;
Umwandeln der Datenmenge in eine variable Datenstruktur, wobei die variable Datenstruktur eine Indexmenge, einen Bereich und eine Funktion umfasst;
Umwandeln der variablen Datenstruktur in eine Menge von Variablen unter Verwendung von mindestens einem der Schritte Blend-Schritt, Cross-Schritt und Nest-Schritt, worin die Umwandlung einer variablen Datenmenge in eine variable Datenstruktur außerdem das Überwachen des Bereichs einer jeden Variablen umfasst, um in der Datenmenge einen Wert zu detektieren, der außerhalb des Bereichs liegt, sodass Fehler in der Datenmenge abgefangen werden;
Abbilden der Menge der Variablen in eine Punktmenge; und
Abbilden der Punktmenge in eine ästhetische Repräsentation.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass, wenn die Datenmenge einen Wert außerhalb des Bereichs enthält, mindesten einer der Schritte aus der folgenden Gruppe ausgeführt wird: der Wert wird auf geeignete Weise zugewiesen oder dem Benutzer wird mitgeteilt, dass der Wert nicht verarbeitet werden kann.

3. Verfahren nach Anspruch 1, worin aus dem Cross-Schritt ein kartesisches Produkt im Bereich der variablen Datenstruktur resultiert.

4. Verfahren nach Anspruch 1, worin aus dem Nest-Schritt eine Schichtung der Werte der Mengen resultiert.

5. Verfahren nach Anspruch 1, worin aus dem Blend-Schritt eine Vereinigung im Bereich der variablen Datenstruktur resultiert.

6. Verfahren nach Anspruch 1, außerdem folgende Schritte umfassend:
a. Bereitstellen einer Variablenliste;
b. Bereitstellen einer Liste der Punktrepräsentationen;
c. Bereitstellen einer Liste der Koordinatensysteme;
d. Bereitstellen einer Liste der ästhetischen Repräsentationen;
worin der Schritt des Abbildens einer Punktmenge in eine ästhetische Repräsentation Folgendes umfasst:
e. Auswählen mindestens einer Variablen aus der Variablenliste;
f. Auswählen mindestens einer Punktrepräsentation aus der Liste der Punktrepräsentationen;
g. Auswählen mindestens eines Koordinatensystems aus der Liste der Koordinatensysteme;
h. Auswählen mindestens einer ästhetischen Repräsentation aus der Liste der ästhetischen Repräsentationen;
i. Bewegen der mindestens einen Variablen an eine vorbestimmte Stelle; und
j. Anzeigen einer sichtbaren Graphik, die die mindestens eine Variable, die mindestens eine Punktrepräsentation, das mindestens eine Koordinatensystem und die mindestens eine ästhetische Repräsentation reflektiert.

7. Verfahren nach Anspruch 6, dadurch gekennzeichnet, dass die mindestens eine Variable, die an eine vorbestimmte Stelle bewegt wird, von mindestens einer Funktion manipuliert worden ist, wobei die Funktion aus der Gruppe ausgewählt wurde, die aus Blend-Funktion, Nest-Funktion und Cross-Funktion besteht.

8. Verfahren nach Anspruch 7, dadurch gekennzeichnet, dass aus der Blend-Funktion eine Vereinigung im Bereich der variablen Datenstruktur resultiert.

9. Verfahren nach Anspruch 7, dadurch gekennzeichnet, dass aus der Cross-Funktion ein kartesisches Produkt im Bereich der variablen Datenstruktur resultiert.

10. Verfahren nach Anspruch 7, dadurch gekennzeichnet, dass der Schritt des Bewegens der mindestens einen Variablen an eine vorbestimmte Stelle die Verknüpfung der mindestens einen Variablen an die Dimensionen eines geometrischen Raums umfasst, in den die ästhetische Repräsentation eingebettet wird.

11. Datenverarbeitungssystem zur mathematischen Konstruktion von Graphen aus Daten und zur ästhetischen Repräsentation der Graphen als Graphik, bestehend aus:
einem Computerprozessor; und

einem Speicher, ansprechbar an den Computerprozessor gekoppelt, der einen Satz von Computerbefehlen enthält zum Ausführen der Verfahrensschritte nach irgendeinem der vorhergehenden Ansprüche, wenn die Befehle vom Computerprozessor ausgeführt werden.

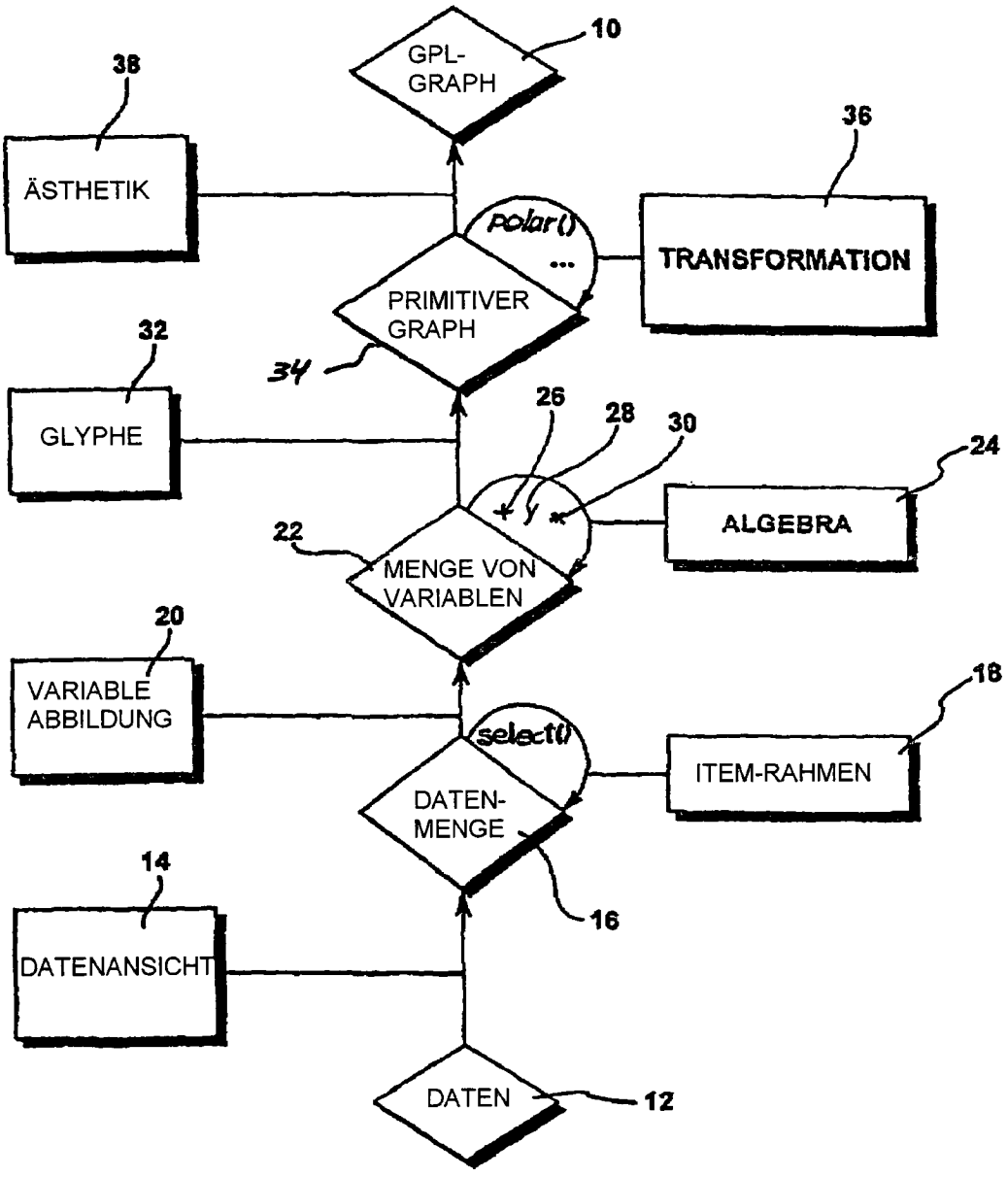
12. System nach Anspruch 11, dadurch gekennzeichnet, dass die Computerbefehle zum Umwandeln der Datenmenge in eine variable Datenstruktur außerdem einen Befehl enthalten, der der Datenmenge einen geeigneten Wert zuweist oder anzeigt, dass der Wert nicht verarbeitet werden kann, wenn die Datenmenge einen Wert außerhalb des Bereichs enthält.

13. Nichtflüchtiges Speichermedium mit in maschinenlesbarem Format codierter Computersoftware, die einen Computer anweist, nach dem Verfahren eines der Ansprüche 1 bis 10 ästhetische Graphik aus Daten zu erzeugen.

Es folgen 18 Blatt Zeichnungen

Anhängende Zeichnungen

FIG.1



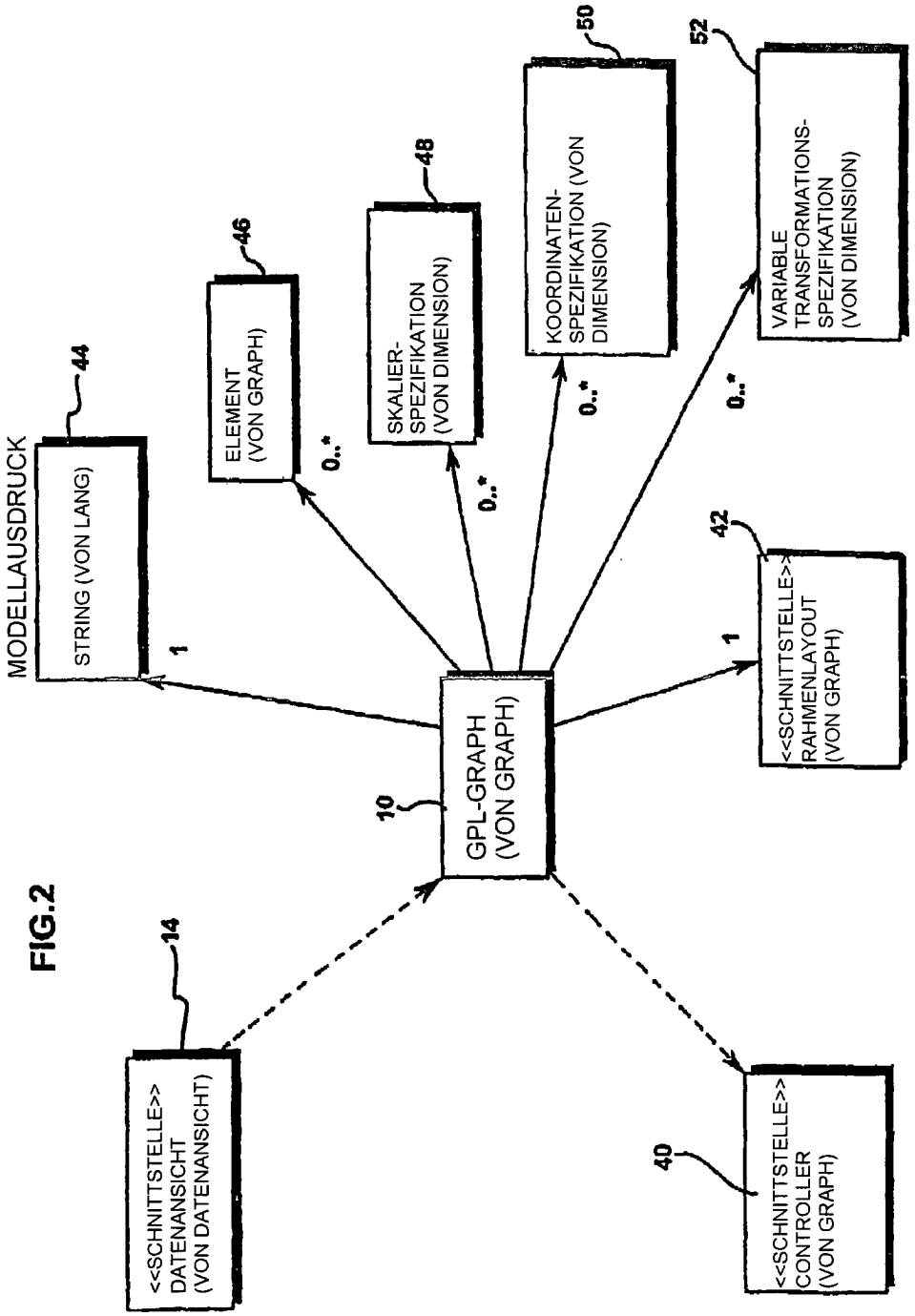

MODELLAUSDRUCK
STRING (VON LANG)
44
ELEMENT (VON GRAPH)
46
SKALIER-SPEZIFIKATION (VON DIMENSION)
48
KOORDINATEN-SPEZIFIKATION (VON DIMENSION)
50
VARIABLE TRANSFORMATIONS-SPEZIFIKATION (VON DIMENSION)
52
<<SCHNITTSTELLE>> RAHMENLAYOUT (VON GRAPH)
42
GPL-GRAPH (VON GRAPH)
10
<<SCHNITTSTELLE>> DATENANSICHT (VON DATENANSICHT)
14
<<SCHNITTSTELLE>> CONTROLLER (VON GRAPH)
40
1
0..*
0..*
0..*
1

FIG.2

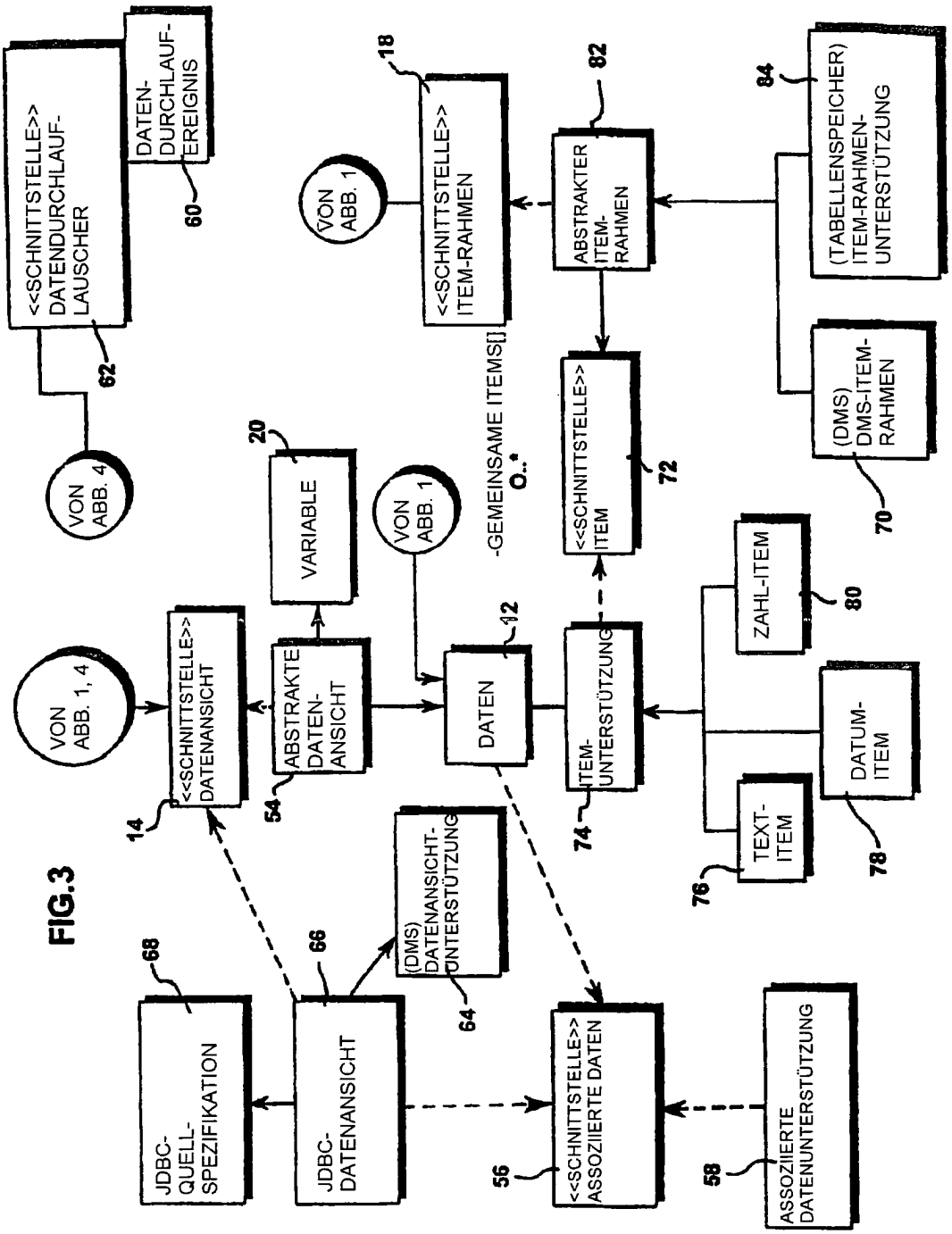

<<SCHNITTSTELLE>>
DATENDURCHLAUF-
LAUSCHER
DATEN-
DURCHLAUF-
EREIGNIS
60
62
VON
ABB. 4
VON
ABB. 1, 4
<<SCHNITTSTELLE>>
DATENANSICHT
14
ABSTRAKTE
DATEN-
ANSICHT
54
20
VARIABLE
VON
ABB. 1
DATEN
12
VON
ABB. 1
<<SCHNITTSTELLE>>
ITEM-RAHMEN
18
-GEMEINSAME ITEMS[]
0..*
ABSTRAKTER
ITEM-
RAHMEN
82
<<SCHNITTSTELLE>>
ITEM
72
ITEM-
UNTERSTÜTZUNG
74
(TABELLENSPEICHER)
ITEM-RAHMEN-
UNTERSTÜTZUNG
84
(DMS)
DMS-ITEM-
RAHMEN
70
ZAHL-ITEM
80
DATUM-
ITEM
78
TEXT-
ITEM
76
(DMS)
DATENANSICHT-
UNTERSTÜTZUNG
64
JDBC-
QUELL-
SPEZIFIKATION
68
JDBC-
DATENANSICHT
66
<<SCHNITTSTELLE>>
ASSOZIIERTE DATEN
56
ASSOZIIERTE
DATENUNTERSTÜTZUNG
58

FIG.3

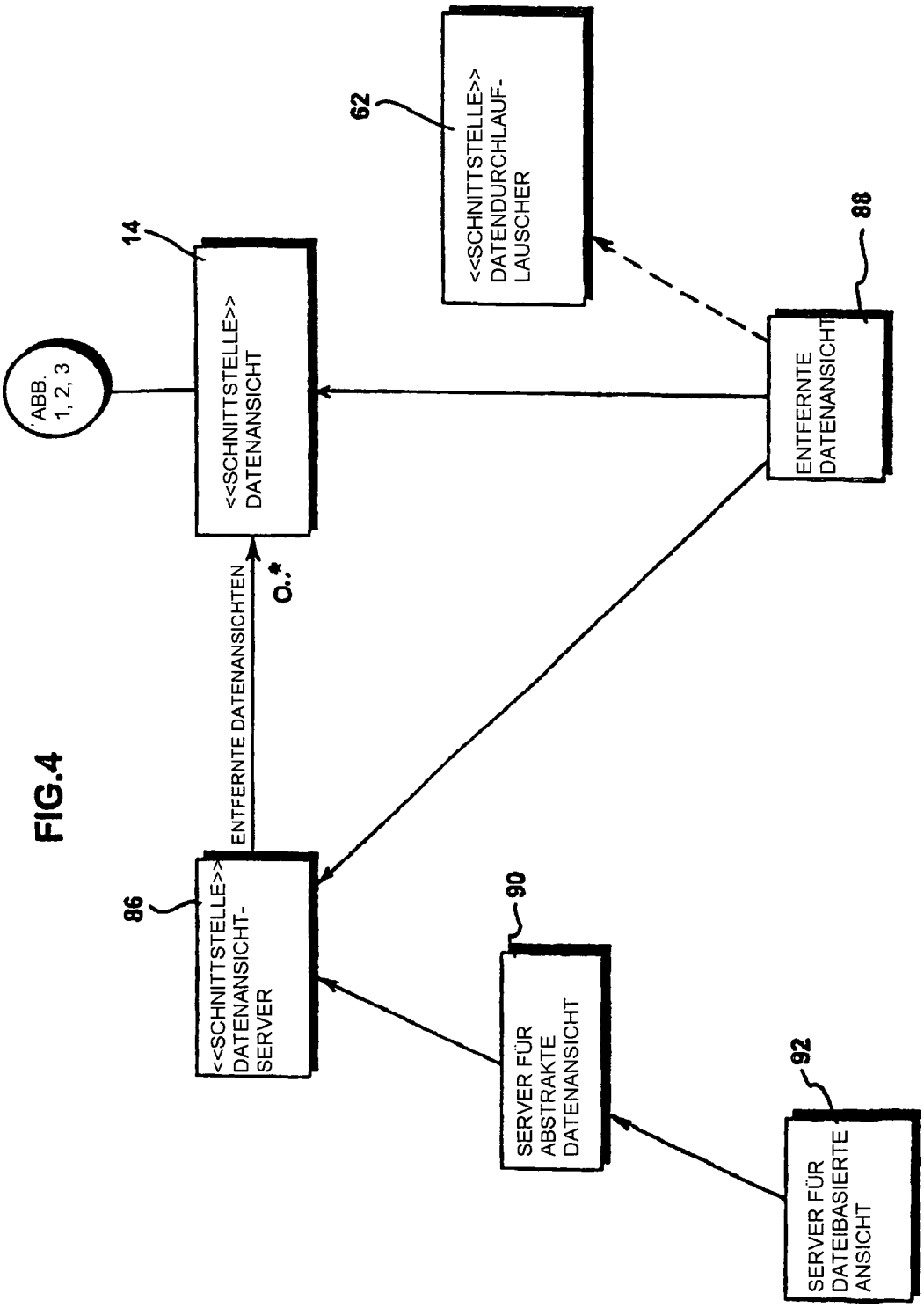

62
<<SCHNITTSTELLE>>
DATENDURCHLAUF-
LAUSCHER
14
<<SCHNITTSTELLE>>
DATENANSICHT
88
ENTFERNTE
DATENANSICHT
ABB.
1, 2, 3
ENTFERNTE DATENANSICHTEN
0..*
86
<<SCHNITTSTELLE>>
DATENANSICHT-
SERVER
90
SERVER FÜR
ABSTRAKTE
DATENANSICHT
92
SERVER FÜR
DATEIBASIERTE
ANSICHT

FIG.4

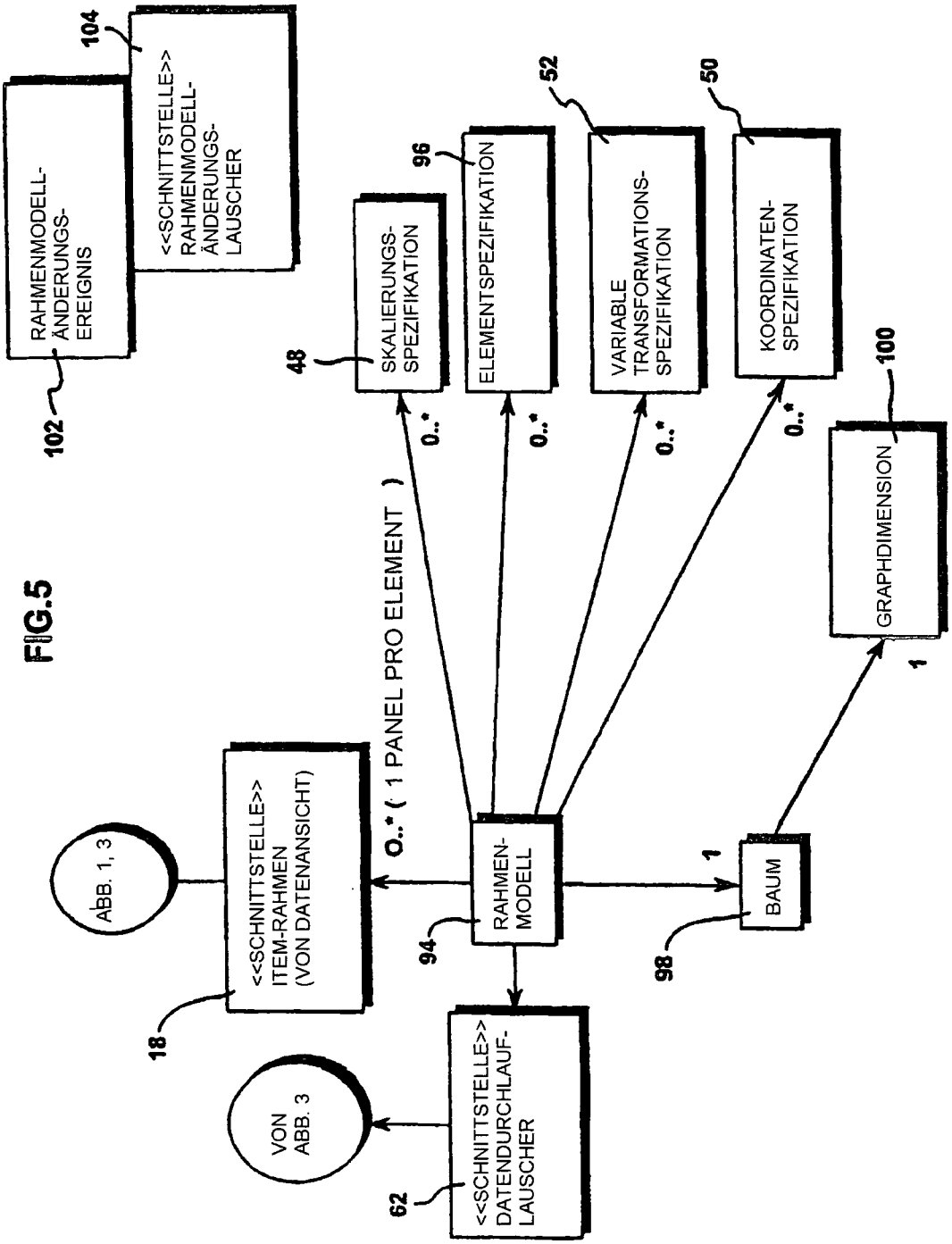

RAHMENMODELL-ÄNDERUNGS-EREIGNIS
102
<<SCHNITTSTELLE>> RAHMENMODELL-ÄNDERUNGS-LAUSCHER
104
SKALIERUNGS-SPEZIFIKATION
48
ELEMENTSPEZIFIKATION
96
VARIABLE TRANSFORMATIONS-SPEZIFIKATION
52
KOORDINATEN-SPEZIFIKATION
50
GRAPHDIMENSION
100
1
0..* (1 PANEL PRO ELEMENT)
0..*
0..*
0..*
0..*
RAHMEN-MODELL
94
1
BAUM
98
<<SCHNITTSTELLE>> ITEM-RAHMEN (VON DATENANSICHT)
18
ABB. 1, 3
<<SCHNITTSTELLE>> DATENDURCHLAUF-LAUSCHER
62
VON ABB. 3

FIG.5

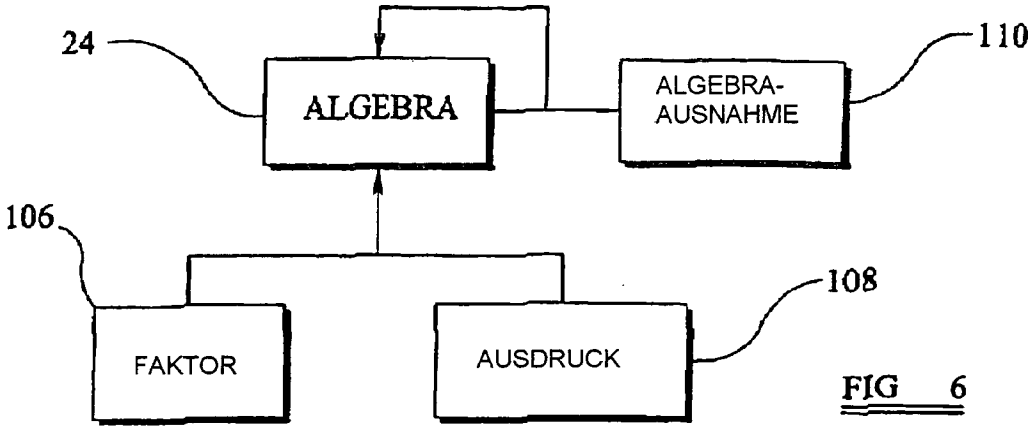


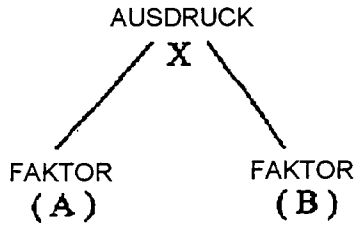
FIG 6

BEISPIEL 1

AUSDRUCK (EINFACHE CROSS-OPERATION MIT ZWEI VARIABLEN):(A*B)

EXPANSION NICHT ERFORDERLICH, ALSO IST DER BAUM EINFACH:

OBJEKTDIAGRAMM:



HANDCODIERUNG:

AUSDRUCK = NEUER AUSDRUCK (NEUER FAKTOR("A"),
ALGEBRA.CROSS, NEUER FAKTOR ("B"))

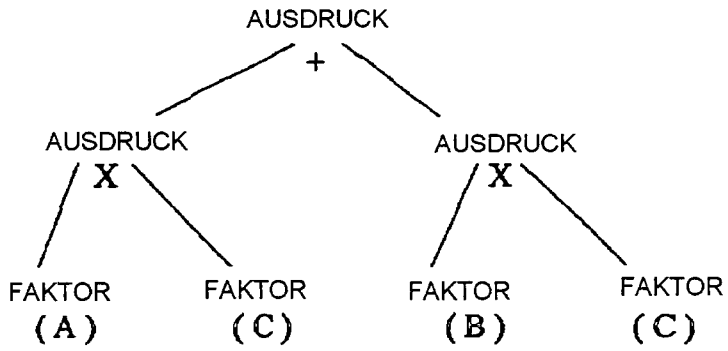
FIG 7

BEISPIEL 2

AUSDRUCK: (A+B)*C

DER AUSDRUCK WIRD IN MONOME EXPANDIERT: A*C+B*C

OBJEKTDIAGRAMM:



HANDCODIERUNG:

```
FAKTOR A=NEUER FAKTOR("A");

FAKTOR B=NEUER FAKTOR("B");

FAKTOR C=NEUER FAKTOR("C");


AUSDRUCK EX1=NEUER AUSDRUCK ( A, ALGEBRA.CROSS, C);

AUSDRUCK EX2=NEUER AUSDRUCK ( B, ALGEBRA.CROSS, C);

AUSDRUCK HAUPTAUSRUCK=NEUER AUSDRUCK
(EX1, ALGEBRA.BLEND, EX2);
```

FIG 8

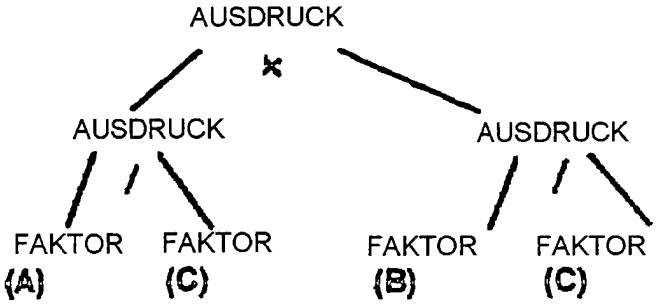
FIG.9

BEISPIEL 3

AUSDRUCK: (A*B)/C

DER AUSDRUCK WIRD IN MONOME EXPANDIERT: A/C*B/C

OBJEKTDIAGRAMM:



HANDCODIERUNG:

FAKTOR A=NEUER FAKTOR("A");

FAKTOR B=NEUER FAKTOR("B");

FAKTOR C=NEUER FAKTOR("C");

AUSDRUCK EX1=NEUER AUSDRUCK (A, ALGEBRA.NEST, C);

AUSDRUCK EX2=NEUER AUSDRUCK (B, ALGEBRA.NEST, C);

AUSDRUCK HAUPTAUSRUCK=NEUER AUSDRUCK
(EX1, ALGEBRA.CROSS, EX2);

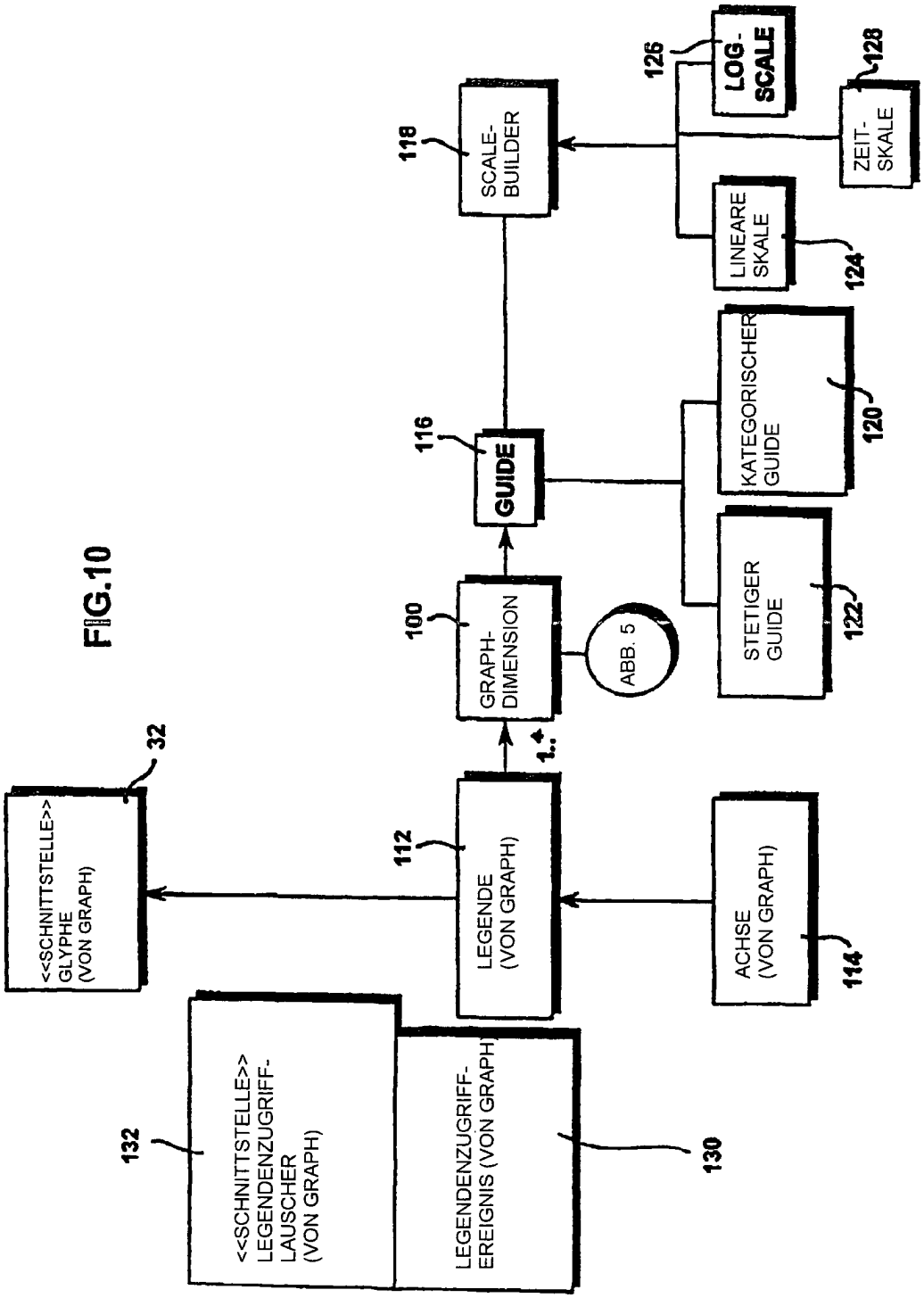
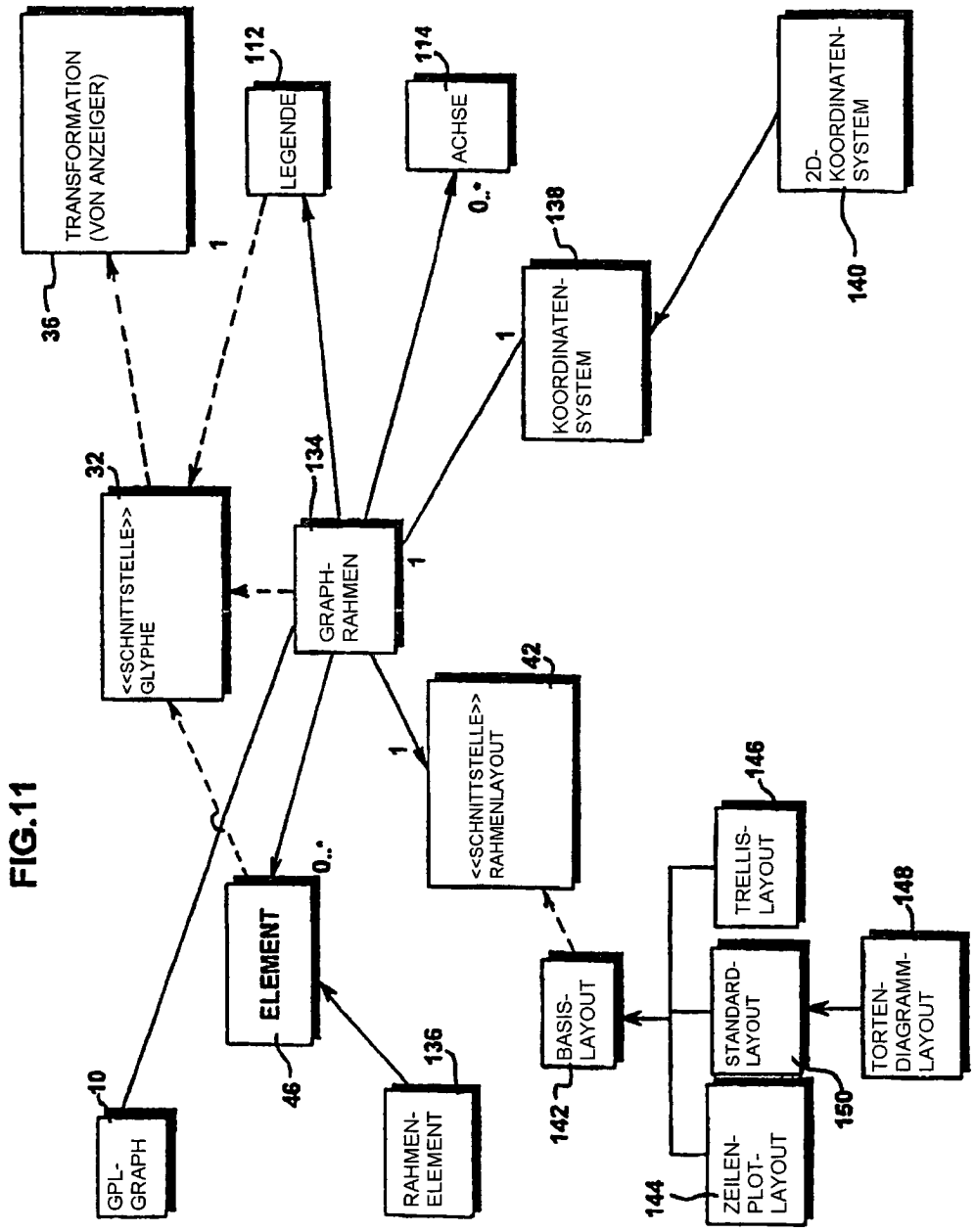

<<SCHNITTSTELLE>>
GLYPHE
(VON GRAPH)
32
<<SCHNITTSTELLE>>
LEGENDENZUGRIFF-
LAUSCHER
(VON GRAPH)
132
LEGENDENZUGRIFF-
EREIGNIS (VON GRAPH)
130
LEGENDE
(VON GRAPH)
112
ACHSE
(VON GRAPH)
114
GRAPH-
DIMENSION
100
1..*
ABB. 5
GUIDE
116
SCALE-
BUILDER
118
KATEGORISCHER
GUIDE
120
STETIGER
GUIDE
122
LINEARE
SKALE
124
LOG-
SCALE
126
ZEIT-
SKALE
128

FIG.10

FIG.11


TRANSFORMATION (VON ANZEIGER)
36
LEGENDE
112
ACHSE
114
0..*
2D-KOORDINATEN-SYSTEM
140
KOORDINATEN-SYSTEM
138
1
<<SCHNITTSTELLE>> GLYPHE
32
GRAPH-RAHMEN
134
<<SCHNITTSTELLE>> RAHMENLAYOUT
42
ELEMENT
0..*
46
GPL-GRAPH
10
RAHMEN-ELEMENT
136
BASIS-LAYOUT
142
TRELLIS-LAYOUT
146
STANDARD-LAYOUT
TORTEN-DIAGRAMM-LAYOUT
148
150
ZEILEN-PLOT-LAYOUT
144

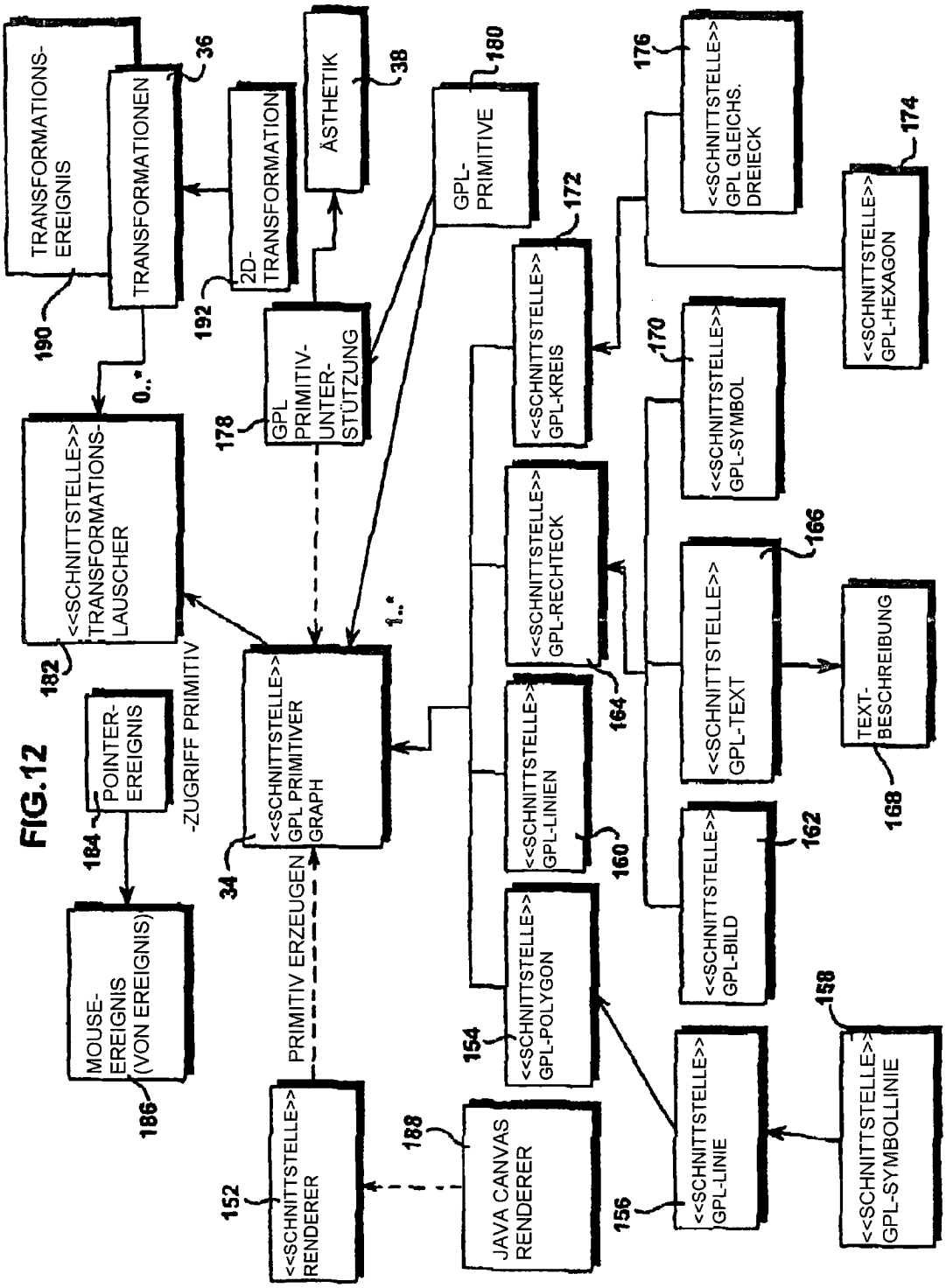
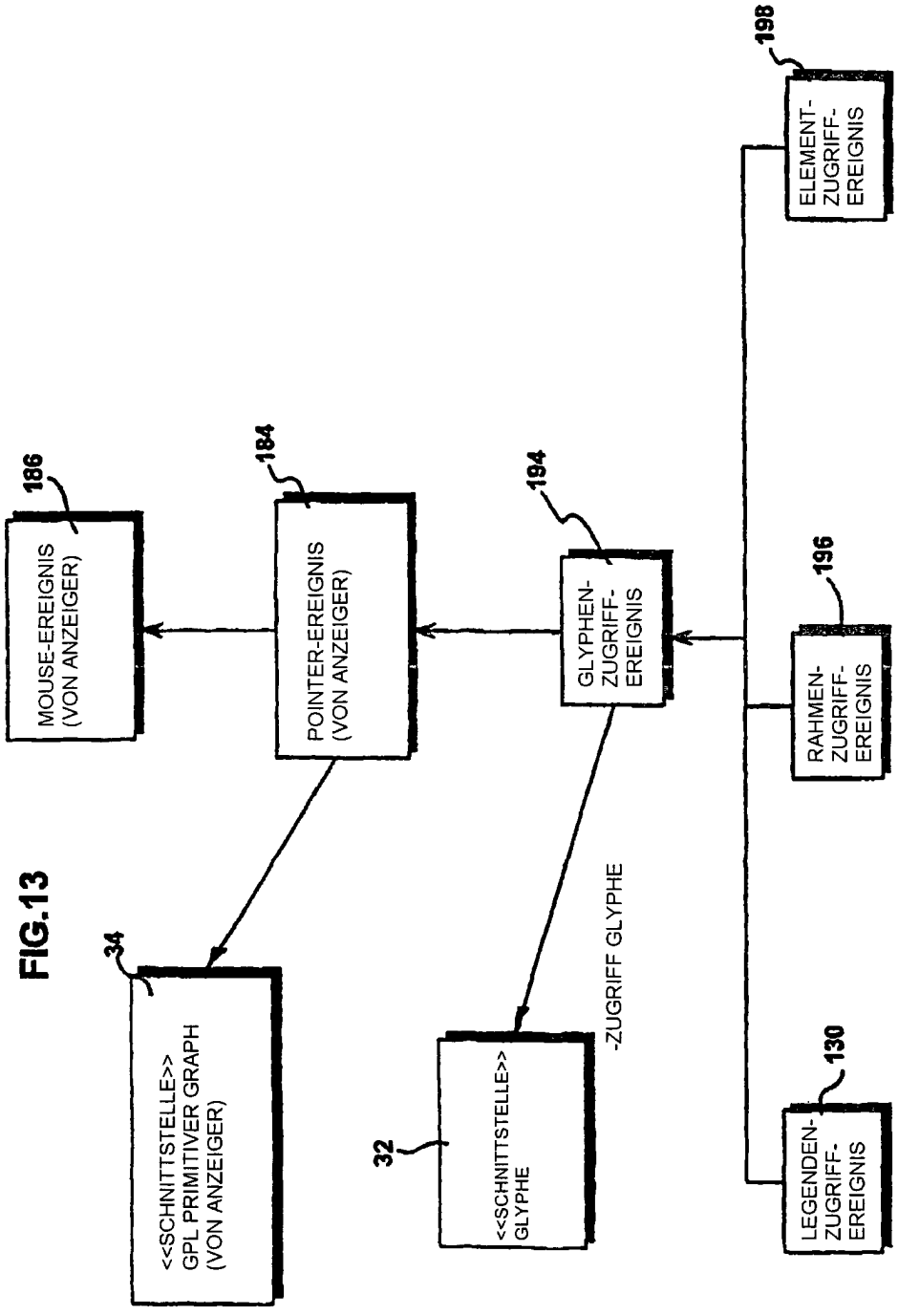

TRANSFORMATIONS-EREIGNIS
TRANSFORMATIONEN
36
2D-TRANSFORMATION
ÄSTHETIK
38
GPL-PRIMITIVE
180
190
192
0..*
<<SCHNITTSTELLE>> TRANSFORMATIONS-LAUSCHER
GPL PRIMITIV-UNTER-STÜTZUNG
178
<<SCHNITTSTELLE>> GPL-KREIS
172
<<SCHNITTSTELLE>> GPL GLEICHS. DREIECK
176
<<SCHNITTSTELLE>> GPL-HEXAGON
174
<<SCHNITTSTELLE>> GPL-SYMBOL
170
182
POINTER-EREIGNIS
184
-ZUGRIFF PRIMITIV
<<SCHNITTSTELLE>> GPL PRIMITIVER GRAPH
1..*
<<SCHNITTSTELLE>> GPL-RECHTECK
164
<<SCHNITTSTELLE>> GPL-TEXT
166
TEXT-BESCHREIBUNG
168
34
<<SCHNITTSTELLE>> GPL-LINIEN
160
<<SCHNITTSTELLE>> GPL-BILD
162
MOUSE-EREIGNIS (VON EREIGNIS)
186
PRIMITIV ERZEUGEN
<<SCHNITTSTELLE>> GPL-POLYGON
154
<<SCHNITTSTELLE>> GPL-LINIE
156
<<SCHNITTSTELLE>> GPL-SYMBOLLINIE
158
152
<<SCHNITTSTELLE>> RENDERER
188
JAVA CANVAS RENDERER

FIG.12

FIG.13


198
ELEMENT-ZUGRIFF-EREIGNIS
186
MOUSE-EREIGNIS (VON ANZEIGER)
184
POINTER-EREIGNIS (VON ANZEIGER)
194
GLYPHEN-ZUGRIFF-EREIGNIS
196
RAHMEN-ZUGRIFF-EREIGNIS
34
<<SCHNITTSTELLE>> GPL PRIMITIVER GRAPH (VON ANZEIGER)
32
<<SCHNITTSTELLE>> GLYPHE
-ZUGRIFF GLYPHE
130
LEGENDEN-ZUGRIFF-EREIGNIS

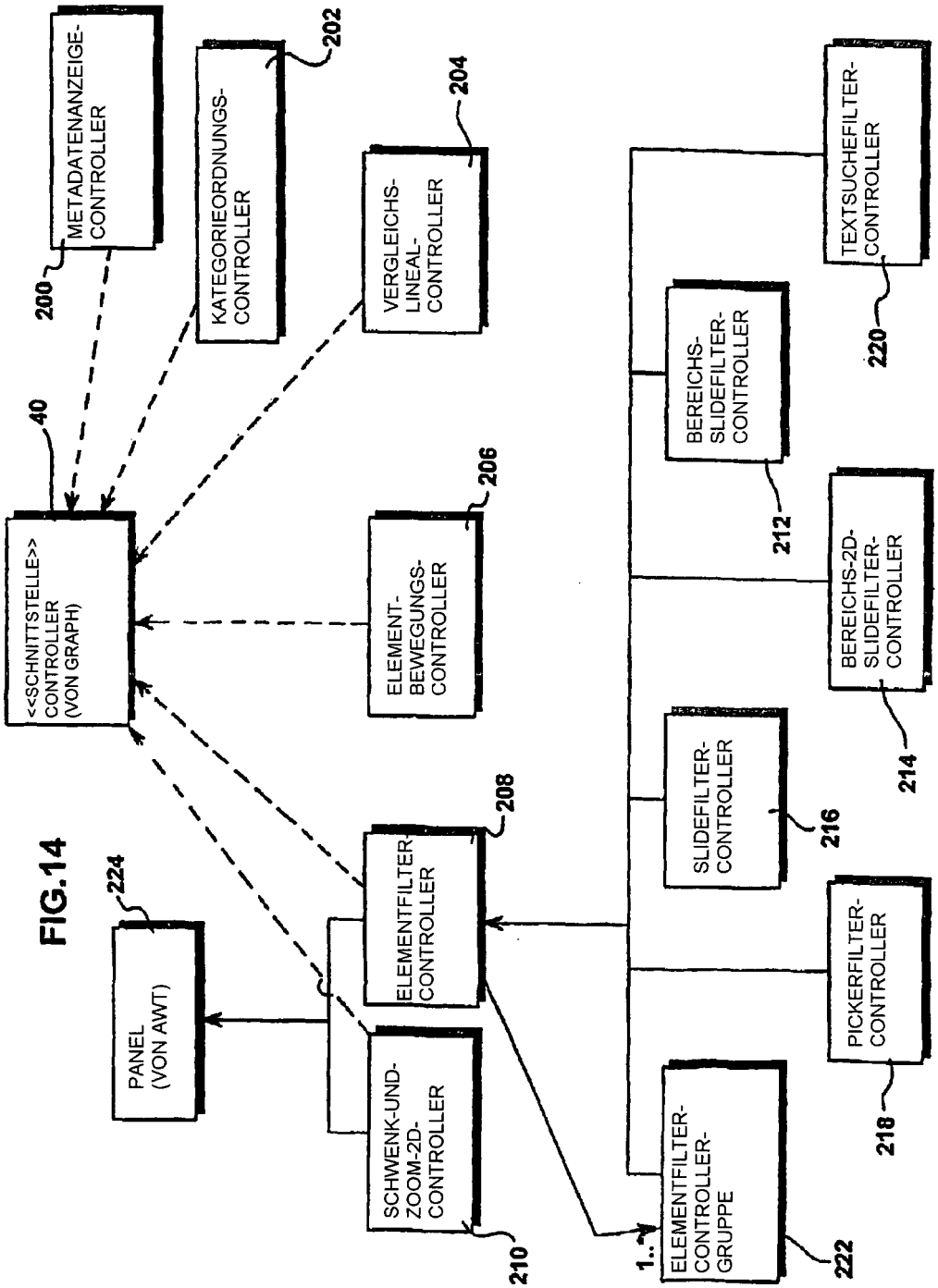

METADATENANZEIGE-CONTROLLER
200
KATEGORIEORDNUNGS-CONTROLLER
202
VERGLEICHS-LINEAL-CONTROLLER
204
<<SCHNITTSTELLE>> CONTROLLER (VON GRAPH)
40
ELEMENT-BEWEGUNGS-CONTROLLER
206
ELEMENTFILTER-CONTROLLER
208
SCHWENK-UND-ZOOM-2D-CONTROLLER
210
PANEL (VON AWT)
224
ELEMENTFILTER-CONTROLLER-GRUPPE
1..*
222
BEREICHS-SLIDEFILTER-CONTROLLER
212
SLIDEFILTER-CONTROLLER
216
PICKERFILTER-CONTROLLER
218
BEREICHS-2D-SLIDEFILTER-CONTROLLER
214
TEXTSUCHEFILTER-CONTROLLER
220

FIG.14

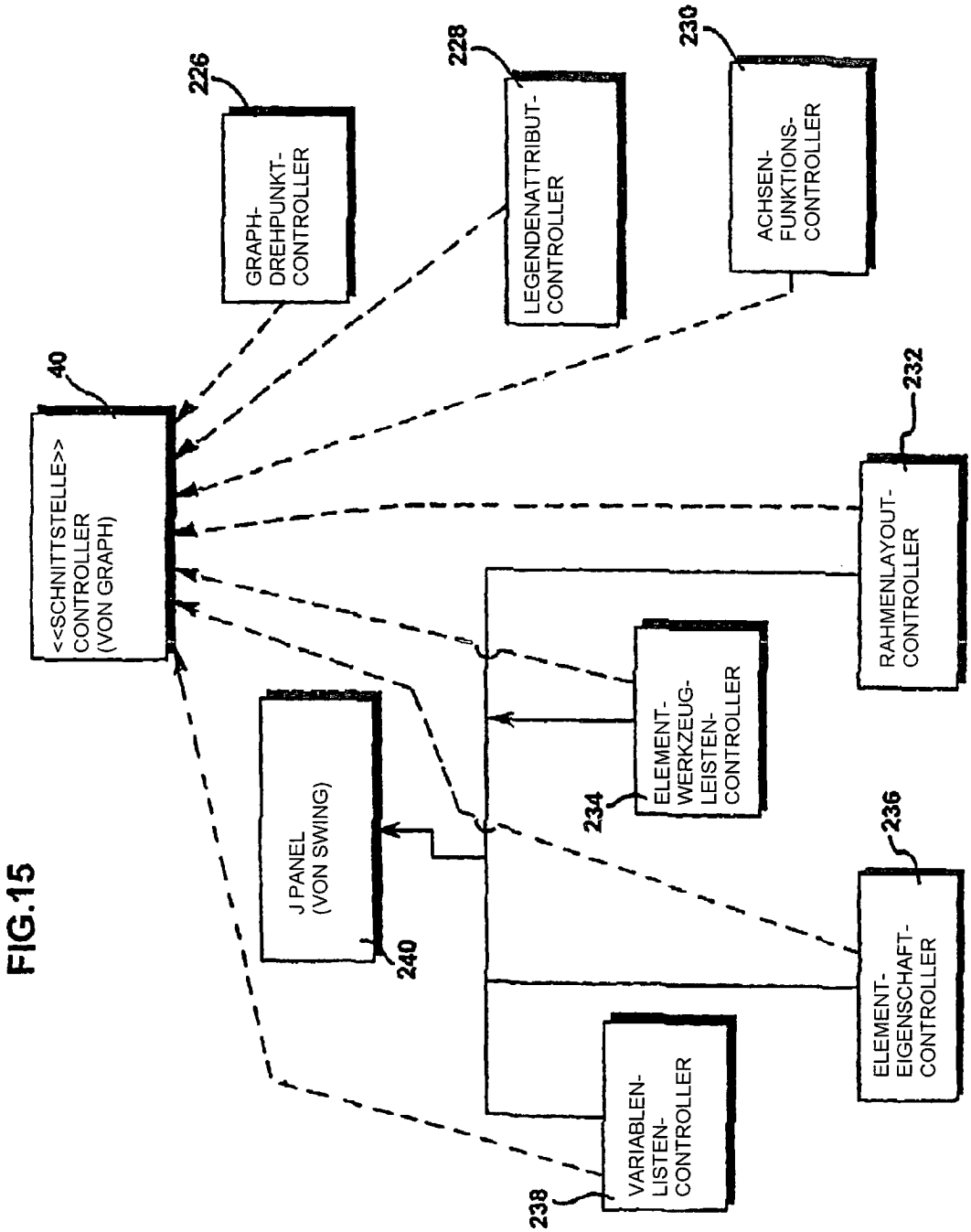
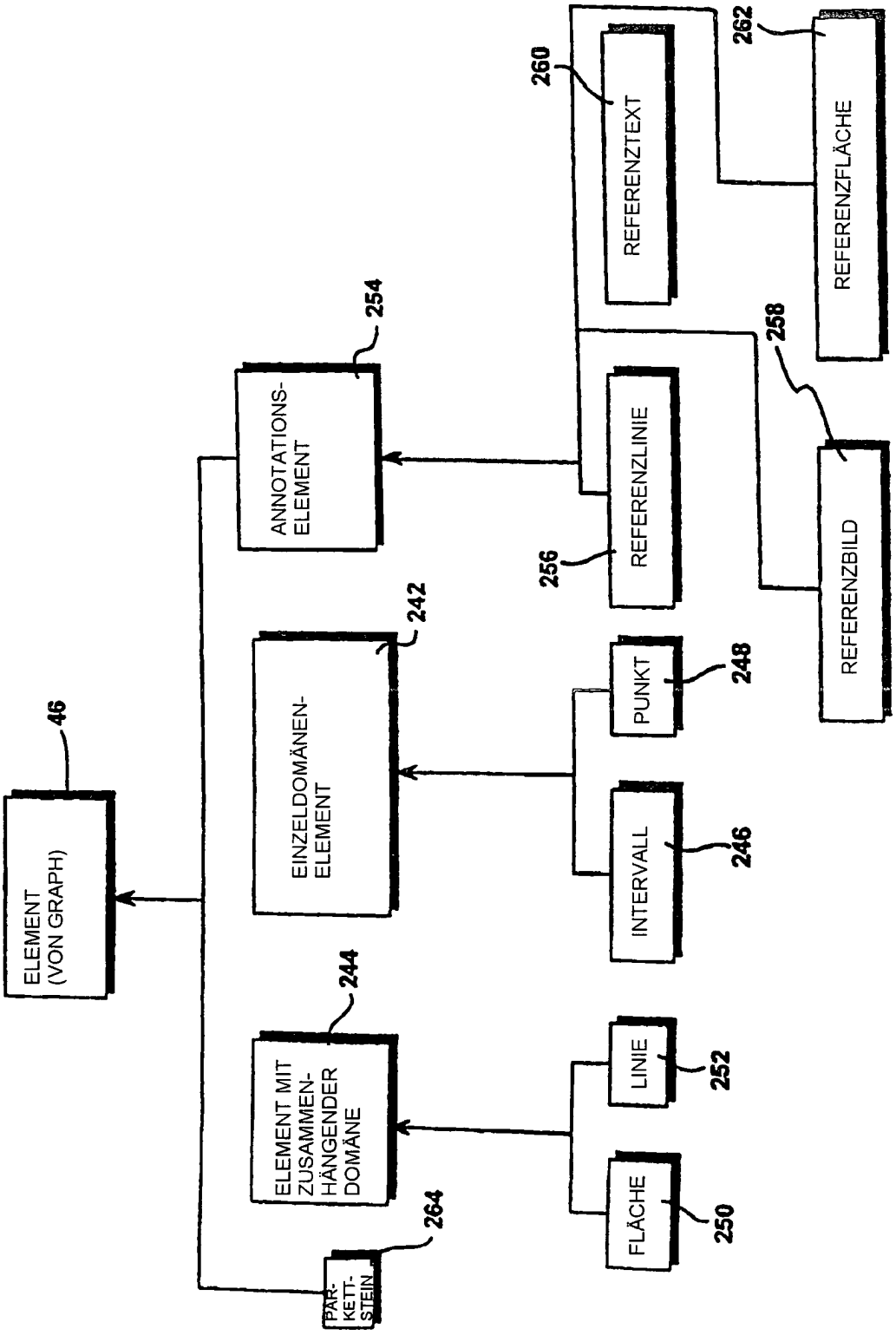

226
GRAPH-DREHPUNKT-CONTROLLER
228
LEGENDENATTRIBUT-CONTROLLER
230
ACHSEN-FUNKTIONS-CONTROLLER
40
<<SCHNITTSTELLE>> CONTROLLER (VON GRAPH)
232
RAHMENLAYOUT-CONTROLLER
ELEMENT-WERKZEUG-LEISTEN-CONTROLLER
234
J PANEL (VON SWING)
240
236
ELEMENT-EIGENSCHAFT-CONTROLLER
VARIABLEN-LISTEN-CONTROLLER
238

FIG.15

FIG.16


ELEMENT (VON GRAPH)
46
ANNOTATIONS-ELEMENT
254
EINZELDOMÄNEN-ELEMENT
242
ELEMENT MIT ZUSAMMEN-HÄNGENDER DOMÄNE
244
PAR-KETT-STEIN
264
REFERENZTEXT
260
REFERENZLINIE
256
PUNKT
248
INTERVALL
246
LINIE
252
FLÄCHE
250
REFERENZFLÄCHE
262
REFERENZBILD
258

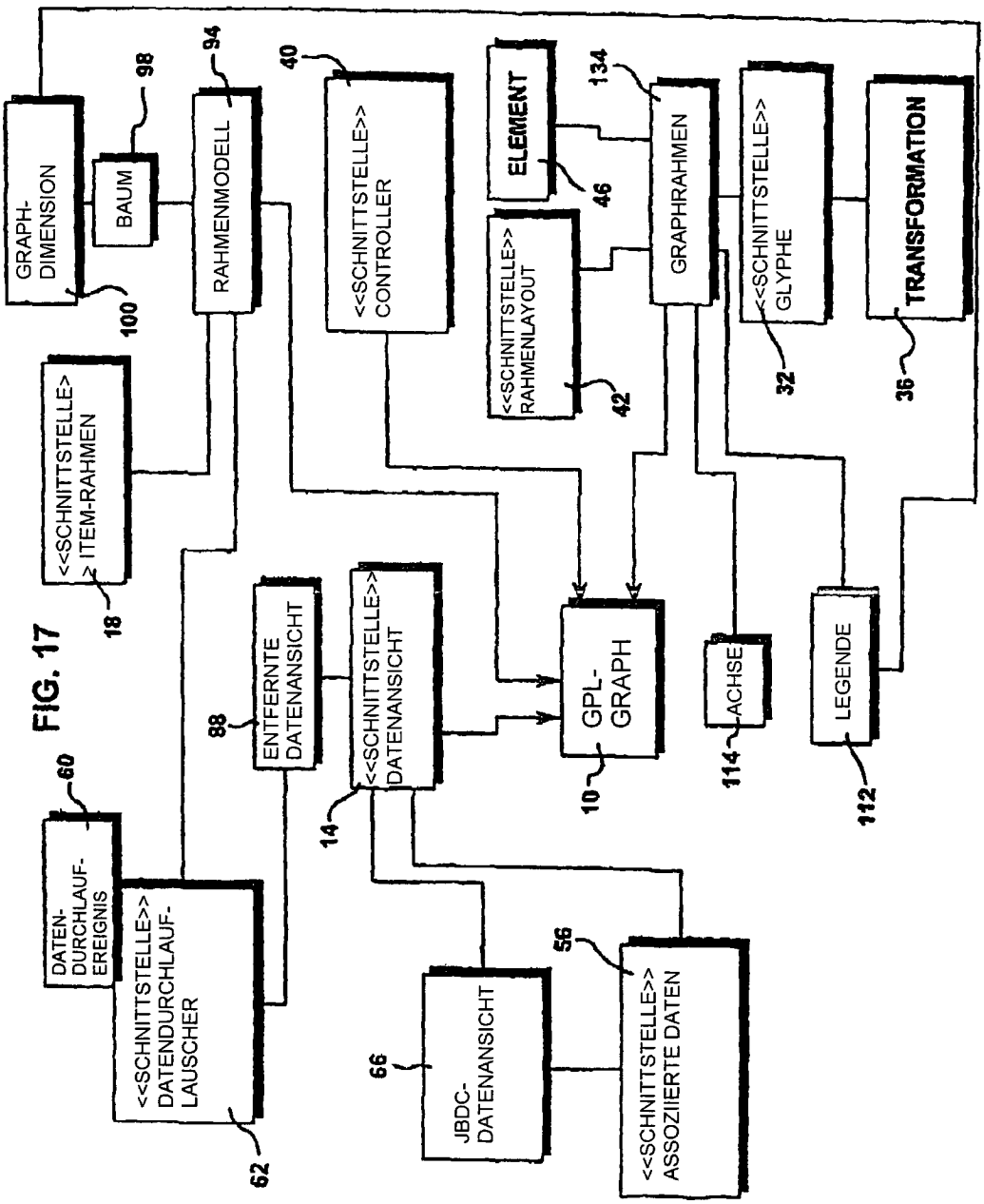

GRAPH-DIMENSION
BAUM
RAHMENMODELL
<<SCHNITTSTELLE>> CONTROLLER
ELEMENT
GRAPHRAHMEN
<<SCHNITTSTELLE>> GLYPHE
TRANSFORMATION
<<SCHNITTSTELLE>> RAHMENLAYOUT
<<SCHNITTSTELLE>> ITEM-RAHMEN
ENTFERNTE DATENANSICHT
<<SCHNITTSTELLE>> DATENANSICHT
GPL-GRAPH
ACHSE
LEGENDE
DATEN-DURCHLAUF-EREIGNIS
<<SCHNITTSTELLE>> DATENDURCHLAUF-LAUSCHER
JBDC-DATENANSICHT
<<SCHNITTSTELLE>> ASSOZIIERTE DATEN
98
94
40
134
46
100
42
32
36
18
88
14
10
114
112
60
56
66
62

FIG. 17

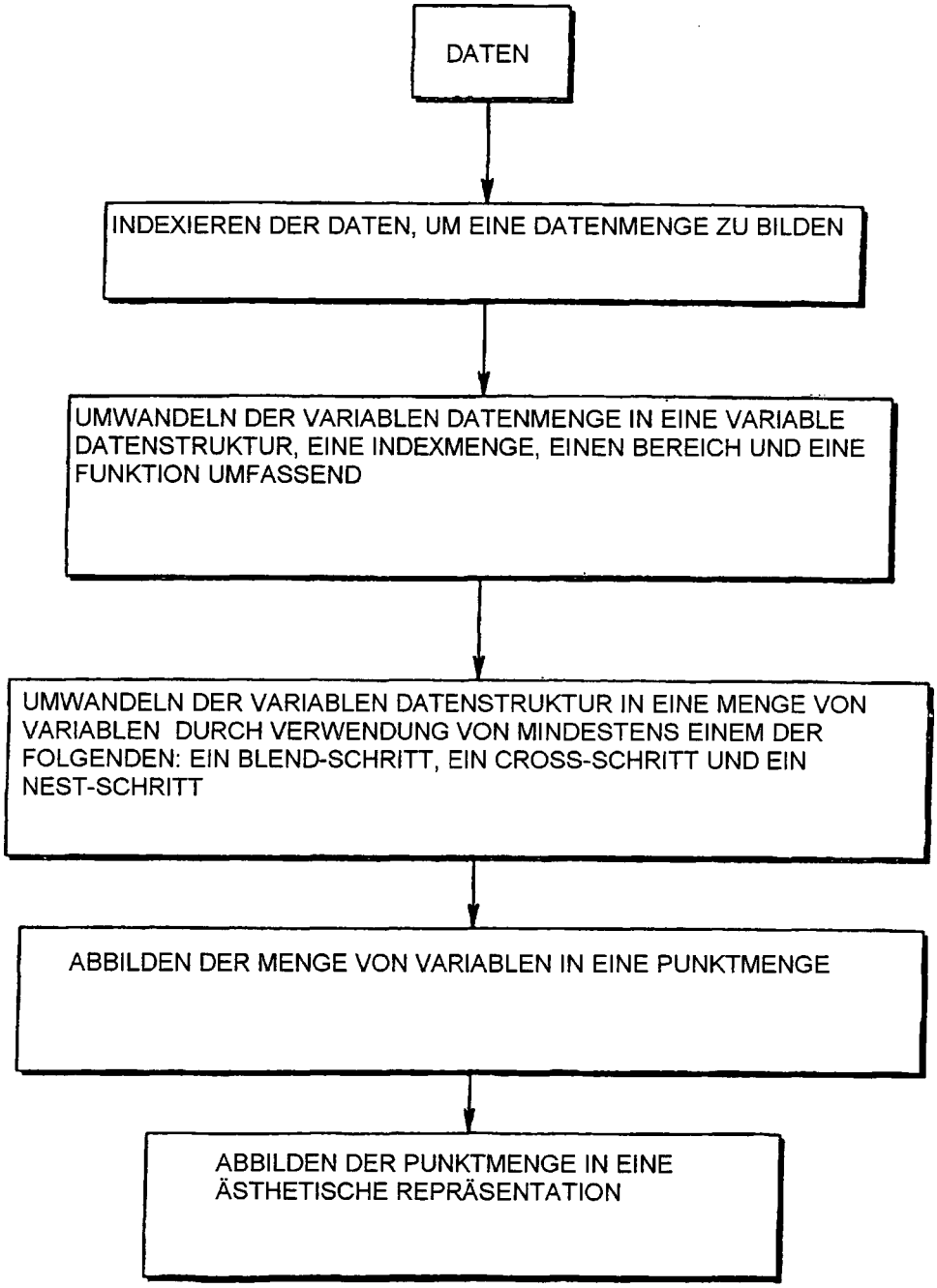

DATEN
INDEXIEREN DER DATEN, UM EINE DATENMENGE ZU BILDEN
UMWANDELN DER VARIABLEN DATENMENGE IN EINE VARIABLE DATENSTRUKTUR, EINE INDEXMENGE, EINEN BEREICH UND EINE FUNKTION UMFASSEND
UMWANDELN DER VARIABLEN DATENSTRUKTUR IN EINE MENGE VON VARIABLEN DURCH VERWENDUNG VON MINDESTENS EINEM DER FOLGENDEN: EIN BLEND-SCHRITT, EIN CROSS-SCHRITT UND EIN NEST-SCHRITT
ABBILDEN DER MENGE VON VARIABLEN IN EINE PUNKTMENGE
ABBILDEN DER PUNKTMENGE IN EINE ÄSTHETISCHE REPRÄSENTATION

FIG 18

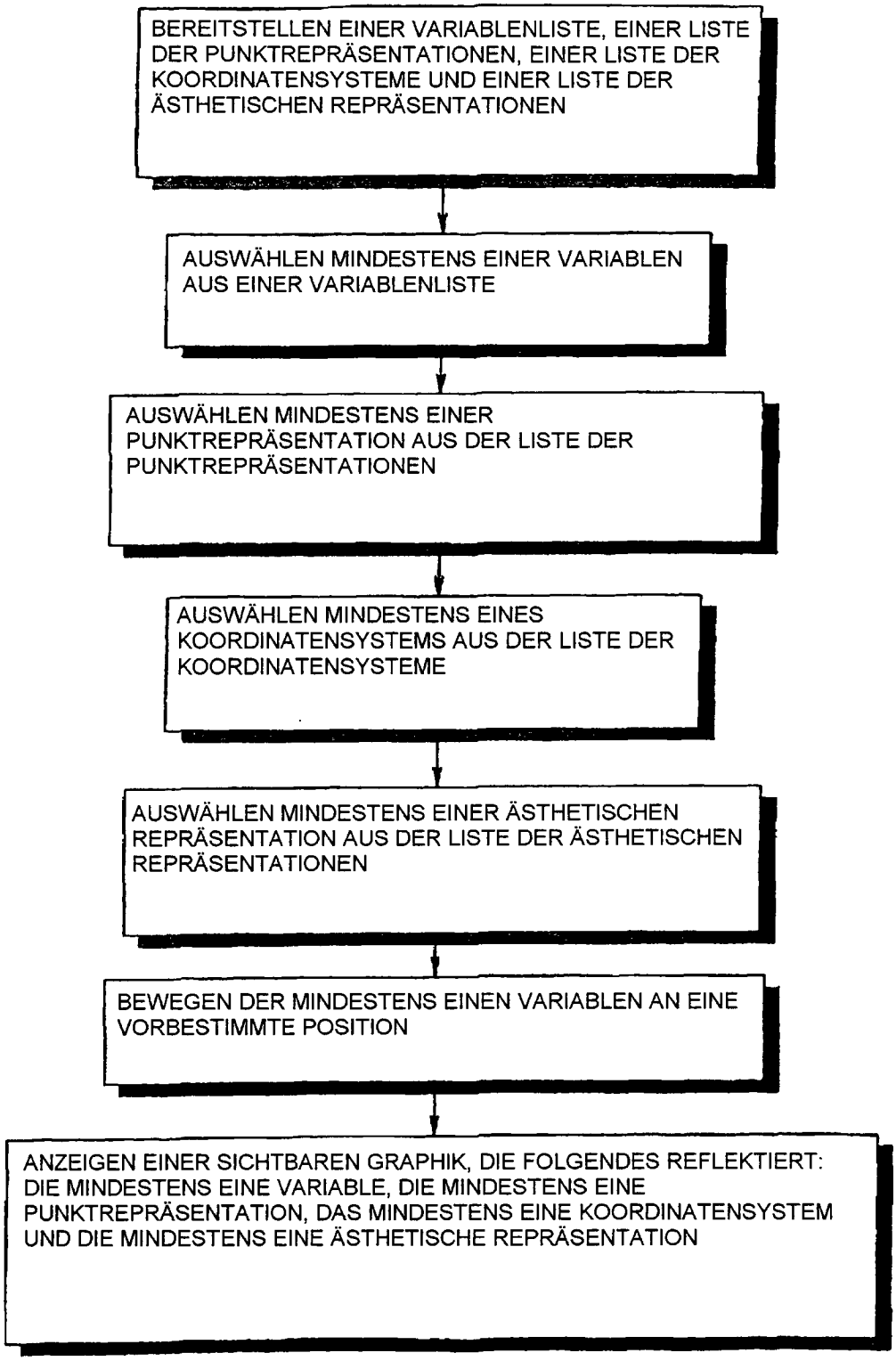

BEREITSTELLEN EINER VARIABLENLISTE, EINER LISTE DER PUNKTREPRÄSENTATIONEN, EINER LISTE DER KOORDINATENSYSTEME UND EINER LISTE DER ÄSTHETISCHEN REPRÄSENTATIONEN
AUSWÄHLEN MINDESTENS EINER VARIABLEN AUS EINER VARIABLENLISTE
AUSWÄHLEN MINDESTENS EINER PUNKTREPRÄSENTATION AUS DER LISTE DER PUNKTREPRÄSENTATIONEN
AUSWÄHLEN MINDESTENS EINES KOORDINATENSYSTEMS AUS DER LISTE DER KOORDINATENSYSTEME
AUSWÄHLEN MINDESTENS EINER ÄSTHETISCHEN REPRÄSENTATION AUS DER LISTE DER ÄSTHETISCHEN REPRÄSENTATIONEN
BEWEGEN DER MINDESTENS EINEN VARIABLEN AN EINE VORBESTIMMTE POSITION
ANZEIGEN EINER SICHTBAREN GRAPHIK, DIE FOLGENDES REFLEKTIERT: DIE MINDESTENS EINE VARIABLE, DIE MINDESTENS EINE PUNKTREPRÄSENTATION, DAS MINDESTENS EINE KOORDINATENSYSTEM UND DIE MINDESTENS EINE ÄSTHETISCHE REPRÄSENTATION

FIG 19