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

(10) **DE 601 28 622 T2** 2007.09.13

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 161 052 B1**
(21) Deutsches Aktenzeichen: **601 28 622.7**
(96) Europäisches Aktenzeichen: **01 303 646.2**
(96) Europäischer Anmeldetag: **20.04.2001**
(97) Erstveröffentlichung durch das EPA: **05.12.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **30.05.2007**
(47) Veröffentlichungstag im Patentblatt: **13.09.2007**

(51) Int Cl.[8]: ***G06F 9/46*** (2006.01)
G06F 17/30 (2006.01)
H04L 29/06 (2006.01)

(30) Unionspriorität:
556180 **21.04.2000** **US**

(73) Patentinhaber:
Polarlake Ltd., Dublin, IE

(74) Vertreter:
Wablat, W., Dipl.-Chem. Dr.-Ing. Dr.jur., Pat.-Anw., 14129 Berlin

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

(72) Erfinder:
Parker, Simon, Shankhill, Dublin 18, IE; MacCarthy, Diarmuid Micheal, Raheen, Limerick, IE; McDaid, Georgina, Wicklow Tow, Co Wicklow, IE; Watson, Charles Ian, Monkstown, Co Dublin, IE; Baker, Robert Patrick, Dublin 3, IE; O'Donoghue, Hugh, Killiney, Co Dublin, IE; Buckley, Patrick Anthony Warren, Listowel, Co Kerry, IE

(54) Bezeichnung: **Betriebssystem für strukturierte Verarbeitung von Information**

Beschreibung

HINTERGRUND DER ERFINDUNG

Gebiet der Erfindung

[0001] Die vorliegende Erfindung betrifft allgemein eine erweiterbare und dynamische Software-Betriebs-Umgebung, die Anwendungen unterstützt, welche strukturierte Informationen verarbeiten, und insbesondere eine Umgebung, die XML-Prozessoren unterstützt.

Beschreibung des Standes der Technik

[0002] Ein Computersystem umfasst Hardware und Software, die dazu angeordnet sind, Daten zu speichern und zu verarbeiten. Die Hardware eines typischen Computersystem umfasst eine zentrale Verarbeitungseinheit (CPU), Internspeicher, externen Speicher, Dateneingabegeräte, Datenausgabegeräte und Datenkommunikationsgeräte.

[0003] Die CPU kann Daten manipulieren, die im Internspeicher liegen, Daten zwischen dem Internspeicher und dem externen Speicher bewegen und die anderen Geräte steuern. Instruktionen, welche ebenfalls im Internspeicher liegen, weisen die CPU an, die Handlungen durchzuführen. Die CPU holt Instruktionen vom Internspeicher und führt sie nacheinander aus. Ein Programm ist eine Abfolge von Instruktionen, um eine Aufgabe zu erfüllen, und der Ausdruck Software bezeichnet Programme im allgemeinen.

[0004] In einem Computersystem werden Daten in einer Form kodiert, die der Computer effizient manipulieren kann, gewöhnlich als Gruppen von binären Ziffern. Im externen Speicher werden Daten in größeren Einheiten organisiert, die als Dateien bekannt sind. Programme, welche nicht ausgeführt werden, können im Internspeicher oder im externen Speicher liegen und können als Daten behandelt werden.

[0005] In den frühen Tagen der Computerverarbeitung musste jedes Programm den Computer wie auch die Erledigung der eigenen Arbeit verwalten. Programme mussten ausführliche Instruktionen zum Steuern von Geräten, Behandeln von Fehlern, Kommunikation mit dem Operator und anschließendem Aufräumen enthalten.

[0006] Diese Hauswirtschaftsarbeiten machten Programme unflexibel und fehleranfällig. Der Programmierer musste wissen, welche Geräte mit dem Computer verbunden waren, wo die Daten gespeichert waren, wieviel Internspeicher zur Verfügung stehen würde und wie Nachrichten an den Operator zu senden waren. Das Programm konnte nicht auf einem anderen Computer ablaufen und, wenn die Hardware wechselte, würde das Programm aufhören zu arbeiten. Die Lösung war, die Arbeit zwischen zwei Arten von Software aufzuteilen: Systemprogrammen und Anwendungsprogrammen.

[0007] Systemprogramme werden von Programmieren geschrieben, die die Computerhardware verstehen. Ein Betriebssystem ist eine Sammlung von Systemprogrammen, die einen Computer verwalten. Wenn dem Computer ein neues Gerät hinzugefügt wird, wird dem Betriebssystem ein neues Systemprogramm hinzugefügt.

[0008] Anwendungsprogramme ("Anwendungen") werden von Programmierern geschrieben, die die Aufgabe verstehen, die das Computersystem erfüllen soll. Das Betriebssystem stellt eine vollständige Umgebung für die Anwendung bereit. Wenn die Anwendung einige Hardwareoperationen ausführen muss, wie beispielsweise Lesen oder Schreiben von Daten oder mit dem Operator zu kommunizieren, ruft sie das Betriebssystem zur Hilfe auf.

[0009] Wenn ein Programm ein anderes Programm aufruft, gibt es eine Schnittstelle zwischen ihnen. In vielen Fällen sind die Einzelheiten der Schnittstelle nicht von Bedeutung, aber wenn verschiedene Programmierer einbezogen sind oder, wenn viele Programme einander auf ähnliche Weise aufrufen, ist die Schnittstelle wichtig und muss präzise definiert werden. In diesem Falle wird sie als Anwendungs-Programmierungs-Schnittstelle (API) bezeichnet. Ein Betriebssystem weist eine API auf.

[0010] Es ist oft notwendig, Daten von einem Computer zu einem anderen zu transportieren. Ein einfaches Verfahren verwendet ein tragbares Speichermedium, wie beispielsweise eine Diskette. Der sendende Computer schreibt Daten unter Verwendung eines Ausgabegerätes, die Diskette wird physikalisch zum empfangen-

den Computer transportiert und der empfangende Computer liest die Daten von der Diskette unter Verwendung eines Eingabegerätes.

[0011] Ein praktischeres Verfahren ist, die zwei Computer durch ein Kabel zu verbinden. Der sendende Computer verwendet ein Datenkommunikationsgerät, um Daten auf das Kabel zu schreiben und der empfangende Computer verwendet ein ähnliches Gerät, um Daten vom Kabel zu lesen.

[0012] Solche eine Anordnung arbeitet in beiden Richtungen. Jeder Computer kann senden und empfangen, aber sie müssen sicherstellen, dass, wenn der eine sendet, der andere dann empfängt. Sie bewerkstelligen dies, indem sie darin übereinstimmen, einem Protokoll zu folgen.

[0013] Eine Kommunikation zwischen zwei Computern ist nützlich, aber eine Kommunikation zwischen mehreren Computern ist noch besser. Drei oder mehr Computer können sich ein Kabel teilen und, wenn ein Computer Daten sendet, empfangen sie alle anderen Computer. Dies wird als Broadcasting (Rundsendung) bezeichnet und eine Gruppe von Computern, die in ähnlicher Weise verbunden sind, werden als Broadcast-Netz bezeichnet.

[0014] Für Broadcast-Kommunikation ist ein komplizierteres Protokoll erforderlich. Insbesondere ist es erforderlich, jedem Computer eine eindeutige Adresse zuzuweisen. Wann immer ein Computer Daten an einen anderen Computer sendet, umfassen sie seine eigene Adresse und die Adresse des Empfängers. Alle Computer hören die Übertragung, aber sie alle außer dem Empfänger ignorieren sie.

[0015] Eine Broadcast-Übertragung funktioniert gut, wenn die Anzahl an Teilnehmern begrenzt ist. Wenn zwei Broadcast-Netze miteinander verbunden werden müssen, ist es besser, einen Computern in jedem Netz die externe Kommunikation handhaben zu lassen.

[0016] Ein Computer, der diese Rolle zugunsten eines Netzes ausführt, wird als Router bezeichnet. Die Router werden direkt miteinander verbunden. Wenn ein Sender Daten an ein anderes Netzwerk rundsendet, sendet der Router sie an den anderen Router. Der andere Router sendet sie an den Empfänger.

[0017] Die zwei Broadcast-Netze und die Verbindung zwischen ihnen bilden ein geroutetes Netz, das als ein "Verbundnetz" (internet) bezeichnet wird. Dieses Verbundnetz kann mit anderen Verbundnetzen verbunden sein, so dass ein größeres Netz gebildet wird, und so weiter. Das Internet ist ein Beispiel eines großen öffentlichen Verbundnetzes.

[0018] Geroutete Netze erfordern kompliziertere Transportprotokolle und Adressen als Broadcast-Netze. Das am meisten verwendete Protokoll- und Adressierschema ist das Internet Protocol (IP).

[0019] IP hilft Routern, Daten entlang Netzen zu bewegen und stellt eine Grundlage für eine Familie von Protokollen für eine spezialisierte Kommunikation bereit. Das Transmission-Control-Protocol (TCP) garantiert einen zuverlässigen Kanal zwischen zwei Anwendungen auf verschiedenen Computern. Das Simple-Mail-Transport-Protocol (SMTP-Protokoll) (SMTP) verwendet TCP, um elektronische Post von einem Computer zu einem anderen zu befördern. Das Hypertext-Transport-Protocol (HTTP), ebenfalls auf TCP basiert, bildet die Basis des World-Wide-Web und wird weit unterstützt. Das File-Transfer-Protocol (Dateiübertragungsprotokoll) (FTP) verwendet zwei oder mehr TCP-Verbindungen, um Dateien zwischen Computern zu bewegen.

[0020] Computersysteme können nur mit Daten arbeiten, aber Menschen sind an den Informationen interessiert, welche die Daten darstellen.

[0021] Hier sind einige Daten, dargestellt als Abfolge von Zeichen:
DUB200003220620030000EI123HTWONTIME08001

[0022] Ein menschlicher Beobachter könnte Muster in den Daten erfassen und einige Informationen durch Schlussfolgerungen und Raten erhalten. Hochentwickelte Computersysteme wurden entworfen, um dasselbe zu bewerkstelligen, allerdings nicht so gut.

[0023] Für gewöhnliche Zwecke jedoch benötigen Computer und Menschen einige Hinweise hinsichtlich der Struktur und dem Kontext. Hier sind dieselben Daten, als eine Abfolge von Elementen strukturiert:
DUB,200003220620030000,EI123,HTW,ONTIME,0800,1

[0024] Das Strukturieren der Daten auf diese Weise hilft ein wenig. Es ist zu sehen, dass das letzte Element die Zahl 1 ist, das zweite Element ein Datum sein könnte und "123" zu "EI" im dritten Element gehört.

[0025] Der Kontext, in welchem diese Daten interpretiert werden sollten, ist: "Luftlinien-Flug-Status". Nun sind die Drei-Zeichen-Flughafenbezeichnungen klar sowie die Flugnummer EI123. Dies erklärt immer noch nicht die Bedeutung der letzten zwei Felder.

[0026] Die Information, die diese Daten tatsächlich darstellen, ist: "Aer-Lingus-Flug EI123 von Dublin nach Heathrow um 06:20:03 GMT am 22. März 2000 landete pünktlich und wurde Terminal 1 zugewiesen."

[0027] Diese freie Textdarstellung macht die Information für Menschen sehr klar – wenigstens denen, die ein wenig vom Luftreisen und der englischen Sprache verstehen. Für Computer ist es jedoch nicht leicht, weil die Struktur verloren gegangen ist.

[0028] Markup-Sprachen (Auszeichnungssprachen) bieten einen leistungsfähigen Kompromiss zwischen dem Informationsgehalt von freiem Text und den festgelegten Strukturen, die Computersystem benötigen. Hier ist dieselbe Information, ausgedrückt unter der Verwendung einer Extensible-Markup-Language (erweiterbare Auszeichnungssprache) (XML).

```
<FLIGHT-EVENT>
      <ORIGIN>DUB</ORIGIN>
      <FLIGHT-TIME> Wed Mar 22 06:20:03 GMT 2000</FLIGHT-TIME>
      <FLIGHT-NUMBER>EI123</FLIGHT-NUMBER>
      <DESTINATION>HTW</DESTINATION>
      <ARRIVAL-TIME>08:00</ARRIVAL-TIME>
      <STATUS>ONTIME</STATUS>
      <TERMINAL>1</TERMINAL>
</FLIGHT-EVENT>
```

[0029] Ein XML-Dokument umfasst Elemente und jedes Element wird durch ein Start-Tag (Startidentifizierungskennzeichen) eingeleitet, das einen Namen enthält, und von einem Ende-Tag, das den selben Namen mit einem vorangesetzten Schrägstrich enthält, gefolgt. Tags sind durch spitze Klammern begrenzt. Elemente können Text oder andere Elemente enthalten. In dem Beispiel oben enthält das Element STATUS den Text ON-TIME und das Element FLIGHT-EVENT enthält sieben andere Elemente.

[0030] Eine Anwendung, welche XML verarbeitet, muss vorbereitet sein, viele Aufgaben auszuführen. Sie wird das Dokument als einen Zeichenstrom vom Netz oder von einem externen Speichererhalten müssen, das Dokument hinsichtlich spezieller Zeichen, wie beispielsweise den spitzen Klammern, die Tags begrenzen, abtasten, Elementnamen und die Textinhalte extrahieren, sicherstellen, dass für jedes Ende-Tag ein Start-Tag vorliegt und sicherstellen, dass die Elemente richtig geschachtelt sind. Um ein XML-Dokument zu schreiben, muss eine Anwendung Elemente zusammentragen, die richtig verschachtelt und in der richtigen Reihenfolge sind, die Tags formatieren und Zeichen zum externen Speicher oder das Netz schreiben.

[0031] Wenn die Aufgabe der Anwendung einfach das Identifizieren von Flügen von Dublin oder das Senden einer Nachricht wann immer sich der Status eines Flugs ändert, wird der Programmierer soviel wie möglich von diesem Hauswirtschaften vermeiden wollen. Ein Weg, dies zu bewerkstelligen, ist, die Anwendung in mehrere Programme zu teilen und dann Programme zu verwenden, die bereits geschrieben sind.

[0032] Ein Programm, das einen Strom von Zeichen liest und Tags und Elemente identifiziert, wird als Parser bezeichnet. Es ist üblich, einen Parser mit anderen Programmen zu verwenden und es wurden Standard-Schnittstellen für XML-Parser entworfen.

[0033] Der Parser kann das gesamte Dokument in den Internspeicher lesen und dann Teile des Dokuments in Antwort auf Aufrufe von einem anderen Programm liefern. Dies ist die Document-Object-Model-Schnittstelle (DOM).

[0034] Alternativ kann der Parser ein anderes Programm aufrufen, sobald er etwas Interessantes im Zeichenstrom erkennt. Diese Schnittstelle ist als Simple API für XML (SAX) bekannt. Jeder Ruf ist als ein Ereignis bekannt, eine Abfolge von Aufrufen ist ein Ereignisstrom, und das Programm, welches der Parser aufruft, ist ein Ereignisprozessor. Ein Ereignisprozessor für eine kleine Anwendung kann schnell geschrieben werden, und ein Programmierer kann eine komplexe Anwendung durch Zusammenbau einer Kette einfacher Ereignisprozessoren erstellen.

[0035] Diese Herangehensweise zu einer Anwendungsentwicklung weist Potential auf, ist aber gegenwärtig kostenintensiv, zeitintensiv und fehleranfällig, weil es keine unterstützende Softwareumgebung gibt. Der Programmierer muss immer noch die Kommunikations-, Verwaltungs- und Hauswirtschafts-Einrichtungen bereitstellen, die solche Software benötigt, und die sich ergebende Anwendung ist nicht so portierbar oder flexibel wie sie sein könnte. Was fehlt, ist ein Betriebssystem für Prozessoren für strukturierte Informationen.

[0036] Aus der US 5,983,267 ist bekannt, ein System zur Verarbeitung von Datenstrukturen bereitzustellen, um ein objektorientiertes Modell bereitzustellen, das die Originaldaten beschreibt, umfassend ein Repository und Resourcen, die Metadaten enthalten, die auf die Originaldaten zeigen und diese beschreiben.

[0037] Die WO 99/57837 beschreibt ein Verfahren einen Dienst auf einem Heimnetz auszuführen, worin eine Datenbank einen Satz von Anwendungs-Schnittstellen-Beschreibungs-Daten-Objekten bereitstellt, von denen jedes Informationen in einem strukturierten Format beinhaltet, um ein Heimgerät durch ein anders Heimgerät, das mit dem Netz verbunden ist, anzuweisen und zu steuern.

[0038] Erfindungsgemäße Aspekte sind in den angehängten unabhängigen Ansprüchen dargelegt.

[0039] Kurz gesagt umfasst eine erfindungsgemäße Ausführungsform eine erweiterbare und dynamische Software-Betriebs-Umgebung, die Anwendungen unterstützt, welche strukturierte Informationen verarbeiten und insbesondere eine Umgebung, die XML-Prozessoren unterstützt.

[0040] Die voranstehenden und andere erfindungsgemäße Aufgaben, Merkmale und Vorteile werden aus der folgenden ausführlichen Beschreibung der bevorzugten Ausführungsformen offenbar, welche auf mehrere Figuren der Zeichnung Bezug nehmen.

ZEICHNUNGEN

[0041] Fig. 1 zeigt eine Betriebs-Umgebung in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform.

[0042] Fig. 2 zeigt verschiedene Informationsflüsse von den Transport-Empfänger-Adaptern **20** von Fig. 1 zu den Stromprozessoren **14** von Fig. 1.

[0043] Fig. 3 zeigt ein größeres Detail einer Ausführungsform des Dokumentenprozessors **114**.

[0044] Fig. 4 zeigt Daten, die von einem Stromprozessor aus der Betriebs-Umgebung **10** der Fig. 1 herausfließen.

[0045] Fig. 5–Fig. 9 zeigen Flussdiagramme von einigen der Schritte, die von der Betriebssystem-Umgebung der Fig. 1–Fig. 4 ausgeführt werden.

AUSFÜHRLICHE BESCHREIBUNG DER BEVORZUGTEN AUSFÜHRUNGSFORMEN

[0046] Bezugnehmend auf Fig. 1 wird gezeigt, dass eine Betriebssystem-Umgebung **10** in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform eine Sammlung von Transport-Übermittler-Adaptern **12**, eine Sammlung von Stromprozessoren **14**, eine Sammlung von Ereignisquellen **16**, eine Sammlung von Anforderungsprozessoren **18** und eine Sammlung von Transport-Empfänger-Adaptern **20** umfasst.

[0047] Es ist gezeigt, dass die Sammlung von Transport-Übermittler-Adaptern **12** einen HTTP-Transport-Übermittler-Adapter **24** und einen SMTP-Transport-Übermittler-Adapter **22** umfasst. Die Sammlung von Stromprozessoren **14** enthält einen oder mehrere Stromprozessoren, von denen jeder mit **26** bezeichnet ist. Es ist gezeigt, dass die Sammlung von Ereignisquellen **16** eine Ereignisquelle **28** umfasst. Es ist gezeigt, dass die Sammlung von Anforderungsprozessoren **18** einen Anforderungsprozessor **32** umfasst. Es ist gezeigt,

dass der SAX-Parser **30** sowohl in der Sammlung der Anforderungsprozessoren **18** als auch in der Sammlung von Ereignisquellen **16** enthalten ist. Es ist gezeigt, dass die Sammlung von Transport-Empfänger-Adaptern **20** einen HTTP-Transport-Empfänger-Adapter **34** und einen SMTP-Transport-Empfänger-Adapter **36** umfasst.

[0048] Es sollte bemerkt werden, dass in verschiedenen erfindungsgemäßen Ausführungsformen jeglicher Anforderungsprozessor, wie beispielsweise **32**, erfolgreich mit jeglichem Transport-Empfänger-Adapter, wie beispielsweise Transport-Empfänger-Adapter **34** oder **36**, arbeiten kann. In ähnlicher Weise kann jeglicher Stromprozessor **26** erfolgreich mit jeglicher Ereignisquelle, wie beispielsweise Ereignisquelle **28** oder SAX-Parser **30**, arbeiten.

[0049] In einer erfindungsgemäßen Ausführungsform ist der Anforderungsprozessor **30** ein SAX-Parser zum Parsen von XML-Dokumenten. In anderen erfindungsgemäßen Ausführungsformen können andere Typen strukturierter Informationen, ohne von dem Bereich der Erfindung abzuweichen, verarbeitet werden. Die Funktionen der in Fig. 1 gezeigten Komponenten werden mit der folgenden Diskussion in Bezug auf andere Figuren deutlicher werden. Wie von der folgenden Diskussion offensichtlich wird, ist die Betriebssystem-Umgebung **10** eine dynamische Betriebs-Umgebung, die hinreichend flexibel ist, um verschiedene Protokolle und Prozessoren für strukturierte Informationen zu handhaben. In einer erfindungsgemäßen Ausführungsform wird die Betriebs-Umgebung **10** für XML-Prozessoren verwendet.

[0050] Im Betrieb werden Informationen durch eine Kombination von Prozessoren verarbeitet, die spezifisch ausgewählt und konfiguriert wurden, um die Informationen zu verarbeiten. Sie können von einem Transport-Empfänger-Adapter **34** oder **36** empfangen werden, durch einen Anforderungsprozessor, wie beispielsweise dem SAX-Parser **30** und mehrere Stromprozessoren (**26**) hindurchgehen und schließlich durch einen Transport-Übermittler-Adapter, wie beispielsweise **22** oder **24**, gesendet werden.

[0051] Fig. 2–Fig. 4 zeigen weitere Einzelheiten der Umgebung **10**, indem der Fluss von Informationen durch die Umgebung gezeigt ist. Fig. 2 bis Fig. 4 zeigen lediglich eine der vielen erfindungsgemäßen Ausführungsformen. Es sollte jedoch verstanden werden, dass andere bekannte Verarbeitungstypen durch die vorliegende Erfindung erreicht werden, ohne von dem Bereich der vorliegenden Erfindung abzuweichen.

[0052] In Fig. 2 ist ein Informationsfluss von einem Transport-Empfänger-Adapter **103** zu einem Dokumentenprozessor **110** und ein anderer Fluss vom Anforderungskanal-Behandler **104** zu einem Anforderungsprozessor **119** und ein anderer Fluss von einer Ereignisquelle **118** zu einem Dokumentenprozessor **114** gezeigt.

[0053] In Fig. 2 erzeugt ein XML-Strom-Sender **101** außerhalb der Umgebung ein XML-Dokument oder leitet es an einen Transport-Empfänger-Adapter **101** als einen Zeichenstrom **102** weiter, wobei er ein Transportprotokoll verwendet. Jeder Transport-Empfänger-Adapter implementiert ein oder mehrere Transportprotokolle und kann Zeichenströme von mehreren Quellen gleichzeitig empfangen. Sobald der Transport-Empfänger-Adapter anfängt, einen Zeichenstrom zu empfangen, teilt er einen Anforderungskanal **105** zu, um den Zeichenstrom darzustellen und reicht ihn zum Anforderungskanal-Behandler **104** weiter. Anforderungskanäle gestatten es, dass Anforderungsprozessoren Zeichenströme lesen ohne Einzelheiten der Transportprotokolle zu kennen, die verwendet werden, sie zu empfangen. Ein Anforderungskanal ist kein permanenter Teil des Betriebssystems. Vielmehr stellt er ein eingehendes Dokument dar. Er wird erzeugt, wenn das Dokument eintrifft und wird verworfen, wenn das Dokument verarbeitet wurde. Manchmal gibt es viele Kanäle mit Dokumenten in unterschiedlichen Verarbeitungsstufen. Wenn das System im Leerlauf ist, gibt es keine Anforderungskanäle.

[0054] In Fig. 2 erhält der Anforderungskanal-Verteiler **107** den Anforderungskanal **105** vom Anforderungskanal-Behandler **104** und teilt ihn dem Anforderungsprozessor **109** zu. Der Anforderungskanal-Verteiler **107** konsultiert das Anforderungsprozessor-Wörterbuch **108**, um einen geeigneten Anforderungsprozessor auszuwählen, um jeden Anforderungskanal zu bearbeiten. Der Anforderungsprozessor **109** wurde für den Anforderungskanal **105** ausgewählt und der Anforderungsprozessor **119** wurde für den Anforderungskanal **106** ausgewählt.

[0055] Fig. 5 ist ein Flussdiagramm, das die Schritte in diesem Prozess zeigt. Der Prozess startet beim Schritt **500** und fährt bis zum Schritt **502** fort, an welchem Punkt ein Zeichenstrom erhalten wird oder die Verbindung (zum Transport-Empfänger-Adapter) akzeptiert wird. Als nächstes, bei Schritt **504**, wird ein Anforderungskanal erzeugt, gefolgt vom Schritt **506**, welcher unter Verwendung des Anforderungsprozessor-Wörterbuchs **508** nach einem Anforderungsprozessor sucht. Als nächstes, beim Schritt **510**, wird der Anforderungsprozessor, der im Schritt **506** gefunden wurde, dem Anforderungskanal zugewiesen. Schließlich, beim Schritt **512**, wird der Anforderungsprozessor ausgeführt und die Verarbeitung eines eingehenden Zeichenstroms wird bei **514**

vervollständigt.

[0056] Dieser Zuteilungsmechanismus gestattet es der Umgebung, auf verschiedene Arten von Informationen auf geeignete Weisen zu antworten. Die Umgebung kann eine beliebige Anzahl an Anforderungsprozessoren unterstützen, wobei jeder die Fähigkeit der Umgebung, Anforderungskanäle zu bearbeiten, verbessert. Anforderungsprozessoren könne dynamisch zu jeder Zeit hinzugefügt werden.

[0057] Eine Anwendung, die strukturierte Informationen in der Form eines Zeichenstroms verarbeitet, kann in der Umgebung als ein Anforderungsprozessor installiert werden. Ein Anforderungsprozessor muss nicht Informationen an andere Komponenten der Umgebung für eine weitere Verarbeitung befördern. In Fig. 2 verarbeitet der Anforderungsprozessor **119** die Informationen, die er vom Anforderungskanal **106** empfangen hat.

[0058] Wenn eine weitere Verarbeitung erforderlich ist, konvertiert der Anforderungsprozessor den eingehenden Zeichenstrom (wobei er seinen Input von den Anforderungskanälen des Anforderungskanal-Behandlers **104** erhalten hat) in einen Ereignisstrom.

[0059] Eine Ereignisroutine ist eine Softwareprogrammroutine (Code), die aufgerufen (aktiviert) werden kann, um ein spezielles Konstrukt eines Dokuments (beispielsweise eines XML-Dokumentes) darzustellen. Ein Ereignisstrom ist eine geordnete Abfolge von Routinenaufrufen (ein Routinenaufruf ist die Aktivierung der Routine), welche ein Dokument darstellen. Ein Beispiel eines Standardsatzes von Ereignisroutinen, die verwendet werden, um XML-Dokumente darzustellen, ist durch den SRX-Standard spezifiziert.

[0060] Eine Anwendung, die strukturierte Informationen in der Form eines Ereignisstroms erzeugen kann, ist als eine Ereignisquelle bekannt. Eine Anwendung, die strukturierte Informationen in der Form eines Ereignisstromes verarbeitet, ist als ein Stromprozessor bekannt. In Fig. 2 befördert die Ereignisquelle **118** einen Ereignisstrom zum Stromprozessor **117**.

[0061] Fig. 6 ist ein Flussdiagramm, das zeigt, wie eine Ereignisquelle arbeitet. Im Schritt **518** bestimmt die Ereignisquelle, ob mehr Ereignisse erzeugt werden sollten. Falls nicht, hält die Ereignisquelle an. Sonst erzeugt die Ereignisquelle beim Schritt **522** ein neues Ereignis und verarbeitet das neue Ereignis im Schritt **524** durch Aufrufen eines Stromprozessors. Die Prozedur fährt beim Schritt **518** fort.

[0062] Einer der eingebauten Anforderungsprozessoren im XML-Betriebssystem ist der XML-Zeichenstrom-Anforderungsprozessor **109**. Dieser verarbeitet unter Verwendung eines XML-Parsers den XML-Zeichenstrom von einem Anforderungskanal **105** und erzeugt einen Ereignisstrom, während der XML-Parser parst. Der Ereignisstrom wird dann zum anfänglichen Stromprozessor in einem Dokumentenprozessor **14**, wie beispielsweise dem Stromprozessor **111**, befördert. Dieser Anforderungsprozessor ist daher auch eine Ereignisquelle.

[0063] Ein Stromprozessor kann einen anderen Stromprozessor aufrufen, so dass der Ereignisstrom durch mehrere Stromprozessoren hindurchgehen kann. Um ein Ereignis vom Ereignisstrom zu entfernen, unterlässt ein Stromprozessor es einfach, den nächsten Stromprozessor aufzurufen. Um ein Ereignis in den Ereignisstrom einzufügen, führt ein Stromprozessor einen extra (Routinen-)Aufruf auf den nächsten Stromprozessor aus. Beispielsweise befördert in Fig. 2 der Stromprozessor **111** Ereignisse an einen der Stromprozessoren **111**, **112** und **113**. Jeder der Prozessoren **111**, **112** und **113** kann wiederum Ereignisse zu jedem anderen befördern. Der Stromprozessor **117** befördert Ereignisse zum Stromprozessor **116**, welcher Ereignisse zum Stromprozessor **115** befördert.

[0064] Ein Dokumentenprozessor stellt den Kontext bereit, in welchem Stromprozessoren auf einem Ereignisstrom arbeiten. In Fig. 2 stellt der Dokumentenprozessor **110** einen Kontext für die Stromprozessoren **111**, **112** und **113** bereit, um auf dem Ereignisstrom zu arbeiten, der durch die Ereignisquelle **109** erzeugt wurde. In Abhängigkeit von der Konfiguration des Betriebssystems und den Anforderungen der Ereignisquelle arbeiten ein oder mehr der Stromprozessoren im Kontext jedes Dokumentenprozessors. Im Kontext eines Dokumentenprozessors können Stromprozessoren in verschiedenen Konfigurationen angeordnet sein. Sie können den Ereignisstrom in Reihe oder parallel verarbeiten. Die Umgebung bestimmt einen Stromprozessor im Kontext jedes Dokumentenprozessors als den anfänglichen Stromprozessor. Die Ereignisquelle befördert den Ereignisstrom zum anfänglichen Stromprozessor.

[0065] Fig. 2 zeigt zwei Ereignisströme von den Ereignisquellen **109** und **118**, von denen jeder im Kontext von zwei Dokumentenprozessoren verarbeitet wird. Die Umgebung erzeugt auf die Anforderung einer Ereig-

nisquelle einen Dokumentenprozessor und teilt dem Dokumentenprozessor Stromprozessoren aus der Sammlung von Stromprozessoren **14** zu. In Fig. 2 ist der Stromprozessor **111** der anfängliche Stromprozessor des Dokumentenprozessors **110** und der Stromprozessor **117** ist der anfängliche Stromprozessor des Dokumentenprozessors **114**.

[0066] Es ist zu verstehen, dass Fig. 2 eine einzelne Konfiguration der Umgebung **10** zeigt, welche eine gleichzeitige Bearbeitung von drei Dokumenten ist. Das erste Dokument wird gelesen (**102**), während der Prozessor **109** es parst und mehrere Stromprozessoren (**111**, **112**, **113**) es bearbeiten. Der Dokumentenprozessor **110** und die Instanzen des Stromprozessors können nur für diesen Ereignisstrom erzeugt worden sein, und sie können verworfen werden, wenn das Dokument beendet wird.

[0067] Zur selben Zeit wird nun der Anforderungskanal **106** durch den Anforderungsprozessor **119** bearbeitet.

[0068] Ebenfalls zur selben Zeit erzeugt die Ereignisquelle **118** Ereignisse und die Stromprozessoren **115**, **116** und **117** verarbeiten den Ereignisstrom. Der Dokumentenprozessor **114** und die Instanzen des Stromprozessors können nur für diesen Ereignisstrom erzeugt worden sein, und sie können verworfen werden, wenn das Dokument beendet wird.

[0069] Es sollte bemerkt werden, dass **112** und **116** verschiedene Instanzen desselben Stromprozessors sein können.

[0070] In Abhängigkeit von den Ressourcen, die dem Betriebssystem zur Verfügung stehen sind, und dem Arbeitsaufkommen, welches es zu bewältigen hat, kann eine beliebige Zahl von Dokumentenprozessoren vorliegen. Ein bestimmter Dokumentenprozessor wird nicht mehr als ein Dokument zur Zeit behandeln und jedes Dokument wird nur durch einen Dokumentenprozessor hindurchgehen.

[0071] Die Wörterbücher enthalten Muster oder Referenzen auf Muster, um Instanzen von verschiedenen Stromprozessoren, Anforderungsprozessoren und Instruktionen zu erstellen. Instanzen können auf Anforderung erzeugt werden. Beispielsweise können, wenn fünf identische Dokumente auf einmal empfangen werden, fünf Instanzen von einem Stromprozessor-Muster erzeugt werden, um die Dokumente zu handhaben.

[0072] Die Fähigkeit, dynamische Kontexte aufzubauen, in welchen Stromprozessoren arbeiten, stellen eine leistungsfähige und flexible Infrastruktur für Anwendungen, die strukturierte Informationen verarbeiten, bereit.

[0073] Ein Stromprozessor kann auf viele Weisen auf Ereignisse in einem Ereignisstrom antworten, die folgenden sind einige dieser Wege:

- Erfassen von Strukturen, Mustern, Namen oder Schlüsselwörtern in einem Satz von Ereignissen.
- Einfügen neuer Ereignisse in den Ereignisstrom
- Entfernen von Ereignissen vom Ereignisstrom
- Replizieren des Ereignisstroms oder von Teilen davon, wobei jedes Ereignis zu verschiedenen Stromprozessoren geleitet wird
- Kommunizieren mit oder Teilnehmen an einem externen System; beispielsweise:
- Erzeugen einer Ausgabe in ein Fenster, eine Datei, ein Netzziel oder ein anderes System
- Aufruf der Programmierschnittstelle eines externen Systems
- Zugriff auf eine Datenbank zum Speichern oder Wiederauffinden
- Ändern der Konfiguration des Dokumentenprozessors oder von Komponenten des XML-Betriebssystems selbst
- Nachschlagen und Aufrufen von eingebauten oder benutzerdefinierten Kommandos.

[0074] Wenn Stromprozessoren zusammenarbeiten, um eine komplexe Aufgabe auszuführen, kann es erforderlich sein, dass sie miteinander kommunizieren. Sie können den Ereignisstrom verwenden, um private Informationen zu tragen. Der sendende Stromprozessor (UpStream-Prozessor) drückt die privaten Informationen als Ereignis-Routinenaufrufe aus. Der empfangende Stromprozessor (DownStream-Prozessor) extrahiert die Informationen von den Ereignis-Routinenaufrufen.

[0075] Diese Technik kann verwendet werden, um Stromprozessoren Instruktionen und Konfigurationsinformationen von außerhalb der Umgebung als auch von UpStream-Prozessoren bereitzustellen.

[0076] In einer typischen Konfiguration enthält die Umgebung einen Satz von eingebauten Stromprozessoren. Diese stellen Basiseinrichtungen bereit, einschließlich der Fähigkeit, das XML-Betriebssystem mit neuen

Stromprozessoren zu erweitern.

[0077] Fig. 3 zeigt detaillierter eine Ausführungsform eines Dokumentenprozessors, wie beispielsweise **114** in Fig. 2. Es ist gezeigt, dass der Dokumentenprozessor **114** auf ein Instruktions-Wörterbuch **210**, ein Stromprozessor-Wörterbuch **211** und ein Anforderungsprozessor-Wörterbuch **108** verweist, und, dass er einen Anforderungsprozessorlader **209**, einen Stromprozessorlader **208**, einen Instruktionslader **207**, einen Instruktionsinterpreter-Stromprozessor **206**, einen benutzerdefinierten Stromprozessor **205**, einen Transportstromprozessor **204** und einen Interpreter-Stromprozessor **203** umfasst. Es ist gezeigt, dass der Dokumentenprozessor **114** mit der Ereignisquelle **118** gekoppelt ist.

[0078] Die Fig. 7, Fig. 8 und Fig. 9 zeigen, wie Stromprozessoren arbeiten. Die Stromprozessoren empfangen ein Ereignis zur Zeit, daher zeigt jedes Flussdiagramm, wie ein Stromprozessor ein einzelnes Ereignis handhabt. Fig. 7 zeigt die allgemeine Logik eines Stromprozessors, und Fig. 8 und Fig. 9 zeigen die Details von zwei der eingebauten Stromprozessoren.

[0079] In Fig. 7 entscheidet der Stromprozessor im Schritt **528**, ob das Ereignis relevant für diesen Stromprozessor ist. Falls dies der Fall ist, verarbeitet er im Schritt **530** das Ereignis entsprechend dem Entwurf des Stromprozessors. Der Schritt **532** bestimmt, ob dieses Ereignis vom Ereignisstrom entfernt werden soll, und falls dies der Fall ist, endet der Prozess. Falls der Schritt **528** bestimmt, dass das Ereignis nicht relevant war, oder der Schritt **532** bestimmte, dass es nach der Bearbeitung weiter befördert werden sollte, dann bestimmt der Schritt **534**, ob im Dokumentenprozessor ein anderer Stromprozessor diesem Stromprozessor folgt. Falls dies der Fall st, ruft der Stromprozessor im Schritt **536** den nächsten Stromprozessor auf, bevor der Prozess endet.

[0080] In einer typischen Konfiguration kann das Stromprozessor-Wörterbuch **211** Referenzen auf die folgenden (Komponenten) enthalten:

- Stromprozessorlader **208**
- Instruktionslader **207**
- Anforderungsprozessorlader **209**
- Instruktionsinterpreter **206**
- Transportstromprozessor **204**
- Interpreterstromprozessor **203**

[0081] Das Stromprozessor-Wörterbuch **211** bedient das gesamte XML-Betriebssystem. Es kann auf einem externen Speicher gehalten werden, so dass es verfügbar bleibt, nachdem das Betriebssystem heruntergefahren und wieder hochgefahren wurde. Diese Fähigkeit wird als "Persistenz" bezeichnet.

[0082] In einer typischen Konfiguration enthält das Instruktions-Wörterbuch **210** anfänglich Referenzen auf verschiedene Gruppen von Instruktionen:

- Konfigurations-Instruktionen verwalten den Status des XML-Betriebssystems, insbesondere zur Zeit des Hochfahrens.
- Stromprozessorsteuerungs-Instruktionen warten das Stromprozessor-Wörterbuch.
- Instruktionssteuerungs-Instruktionen warten das Instruktions-Wörterbuch.
- Interpretersteuerungs-Instruktionen verwalten den Status des Interpreterstromprozessors.
- Anforderungsprozessorsteuerungs-Instruktionen warten das Anforderungsprozessor-Wörterbuch.

[0083] Das Instruktions-Wörterbuch bedient das gesamte XML-Betriebssystem und kann persistent sein.

[0084] Die Funktionen und Wechselwirkungen der Lader **209**, **208** und **207** und der Prozessoren **206**, **205**, **204** und **203** von Fig. 3 sind hier untenstehend dargestellt.

[0085] Ein Interpreter-Stromprozessor **203** erfasst Kommandos und Parameter im Ereignisstrom, initiiert die geeigneten Aktionen und kann Ereignisse vom Strom entfernen.

[0086] Ein Interpreter-Stromprozessor erkennt Konstrukte, die einer bestimmten Syntax entsprechen. Wenn er solch ein Konstrukt antrifft, schlägt er einen Eintrag in einem Stromprozessor-Wörterbuch **211** nach. Falls eine Übereinstimmung vorliegt, richtet er den Ereignisstrom zum entsprechenden Stromprozessor. Beispiel 5, hier untenstehend dargestellt, ist ein Beispiel, wie der Interpreter Ereignisse entfernt. Der Instruktionsinterpreter-Stromprozessor **206** erkennt Konstrukte, die der Syntax für Instruktionen entsprechen. Wenn er eine mögliche Instruktion erkennt, schlägt er einen Eintrag im Instruktions-Wörterbuch **210** nach. Wenn eine Überein-

stimmung vorliegt, initiiert er die entsprechende Aktion. Im Beispiel 4, hier untenstehend dargestellt, werden zwei Instruktionen aufgerufen.

[0087] Fig. 8 zeigt die Schritte zur Verarbeitung eines Ereignisses durch die Interpreter **203** und **206**. Im Schritt **540** startet der Prozess, und danach wird eine Bestimmung bei **542** durchgeführt, ob oder ob nicht das gegenwärtige Ereignis ein Kommando ist, um den Stromprozessor zu initiieren. Falls dies der Fall ist, wird im Schritt **544** durch das Stromprozessor-Wörterbuch eine Suche nach einem Stromprozessor durchgeführt.

[0088] Danach wird der aufgefundene Stromprozessor im Schritt **548** mit der Kette von Stromprozessoren verbunden und der Prozess hält bei **562** an.

[0089] Wenn der Schritt **542** bestimmte, dass das Ereignis kein Stromprozessorkommando war, dann bestimmt der Schritt **550**, ob das Ereignis eine Instruktion ist. Falls dies der Fall ist, lädt der Interpreter den Instruktionsinterpreter-Stromprozessor. Der Instruktionsinterpreter durchsucht im Schritt **522** das Instruktions-Wörterbuch und die aufgefundene Instruktion vom Wörterbuch wird im Schritt **556** aufgerufen und der Prozess hält an. Ansonsten wird das Ereignis zum nächsten Stromprozessor in der Kette weitergereicht, falls vorhanden (Schritte **558** und **560**).

[0090] Der Instruktionslader **207** liest eine Instruktion vom Ereignisstrom und installiert dieselbe im Instruktions-Wörterbuch **210**.

[0091] Der eingebaute Stromprozessorlader **208** liest Informationen, die vom Stromprozessormodul vom Ereignisstrom bereitgestellt sind und installiert diese im Stromprozessor-Wörterbuch **211**.

[0092] Der Anforderungsprozessorlader **209** liest einen Anforderungsprozessor vom Ereignisstrom ein und installiert diesen im Anforderungsprozessor-Wörterbuch **108** (gezeigt in Fig. 2).

[0093] Instruktionen, Stromprozessoren und Anforderungsprozessoren sind Muster für ausführbare Module, die veröffentlichten Schnittstellen entsprechen. Jedes Muster ist zur Verwendung durch den entsprechenden Lader **207**, **208** oder **209** in einem XML-Dokument kodiert.

[0094] Ein oder mehrere benutzerdefinierte Stromprozessoren, wie beispielsweise der benutzerdefinierte Stromprozessor **205**, die vom Stromprozessorlader installiert sind, sind zur unmittelbaren Verwendung vom Strom-Wörterbuch **211** verfügbar.

[0095] In Fig. 3 sind die drei "Lader"-Stromprozessoren (Lader **209**, **208** und **207**) Teil des Erweiterungsmechanismus', welcher es dem Betriebssystem gestattet, zu wachsen und sich verschiedenen Aufgaben anzupassen.

[0096] Eine Analogie kann den Mechanismus erklären. Betrachtet sei der Fall, wo ein Angestellter in einem Büro eingehende Papierdokumente entsprechend einem Satz gedruckter Instruktionen, die an der Wand angebracht sind, bearbeitet. Ein Satz von Instruktionen erklärt, was mit Papierdokumenten zu tun ist, die neue Instruktionen enthalten. Von Zeit zu Zeit kommt ein Papierdokument an, das neue Instruktionen enthält. Der Angestellte bringt das Instruktionsblatt an der Wand zusammen mit vorhergehenden Instruktionen an und fährt dann fort, Papierdokumente zu bearbeiten, wobei er vielleicht die neuen Instruktionen verwendet.

[0097] Gleichermaßen wird ein Lader aufgerufen, wenn der Interpreter eine/n neue/n Stromprozessor, Anforderungsprozessor oder Instruktion im Strom erfasst. Der Lader sichert das Muster im geeigneten Wörterbuch.

[0098] Der eingebaute Transportstromprozessor **204** konvertiert den Ereignisstrom in einen Zeichenstrom. Jeglicher Stromprozessor kann die Transportfähigkeiten des XML-Betriebssystem verwenden. Der Transportstromprozessor ist ein Allzweckstromprozessor, welcher durch Einbetten von Kommandos im Ereignisstrom gesteuert werden kann. Er ist als eine praktische Methode für Stromprozessoren bereitgestellt, um die Transportfähigkeiten zu verwenden.

[0099] In Fig. 4 ist ein Datenfluss von einem Stromprozessor in der Betriebsumgebung **10** (von Fig. 1), nämlich Stromprozessor **204**, an einen Empfänger außerhalb der Umgebung, nämlich den Empfänger **308**, gezeigt. Jeglicher Stromprozessor kann mit dem Antwortkanallieferanten (Response Channel provider) **303** zusammenarbeiten, um einen Zeichenstrom an ein oder mehrere Ziele außerhalb der Umgebung zu senden.

[0100] Der Transportstromprozessor **204** erfasst Kommandos, die in den Ereignisstrom durch andere Stromprozessoren eingefügt sind. Wenn er ein Öffnen-Kommando erfasst, fordert der Transportstromprozessor einen Antwortkanal **305** vom Antwortkanallieferanten **303**. Es gibt auch Kommandos, um Output an einen oder mehrere auszusetzen (suspend) und wiederaufzunehmen (resume), sowie einen Kanal zu schließen (close). Dies gestattet es, dass überlappende oder nicht-überlappende Segmente des Zeichenstroms an mehr als ein Ziel verteilt werden.

[0101] Beispiel 7, hier untenstehend dargestellt, dient als ein Beispiel für den Transportstromprozessor, der auf zwei Kanälen auf einmal schreibt.

[0102] Die Funktionen der in Fig. 4 gezeigten Blöcke sind unten aufgeführt:
Der Antwortkanallieferant **303** erzeugt einen Antwortkanal **305** für eine Einheit, die durch eine Adresse identifiziert wird. Der Antwortkanal kann dann verwendet werden, um Informationen an die Einheit zu senden.

[0103] Die Adresse, die verwendet wird, um den Antwortkanal vom Antwortkanallieferanten anzufordern, kann eine transportspezifische Adresse sein. Eine transportspezifische Adresse identifiziert ein Transportprotokoll und enthält Verbindungsinformationen in einem protokollspezifischen Format. Ein Beispiel einer SMTP-Protokolladresse ist: "SMTP://router@xiam.com" und ein Beispiel einer HTTP-Protokolladresse ist "HTTP://router.xiam.com:81".

[0104] Es ist nützlich, eine Einheit durch eine Adresse zu identifizieren, die nicht transportspezifisch ist, um es zu gestatten, dass der Transportmechanismus ausgewechselt wird, ohne die Adresse der Einheit zu ändern. Solch eine Adresse muss zu einer oder mehreren transportspezifischen Adressen aufgelöst werden, bevor ein Antwortkanal erzeugt werden kann.

[0105] In dieser erfindungsgemäßen Ausführungsform ist eine Adresse im Format "XMLR://xiam.com" nicht transportspezifisch, und der Adressenauflöser **302** wandelt diese Adresse in eine transportspezifische Adresse um. Jeder Antwortkanal ist mit einem Übermittler-Transport-Adapter (**306**) verbunden. Jeder Transport-Übermittler-Adapter kann ein oder mehrere Protokolle implementieren.

[0106] Der Transport-Übermittler-Adapter verbindet sich mit dem Empfänger **308** (welches ein Verbraucher sein kann) und übermittelt den Zeichenstrom **307**. Der Transport-Übermittler-Adapter **306** kann anfangen, einen XML-Strom zu senden, während ein Dokumentenprozessor immer noch den Ereignisstrom bearbeitet und, während ein Transport-Empfänger-Adapter immer noch einen Zeichenstrom empfängt.

[0107] In dieser erfindungsgemäßen Ausführungsform sehen Adressen, die nicht transportspezifisch sind, wie folgt aus:
XMLR://<Domain>

[0108] Der Adressenauflöser befragt den bezeichneten Domain-Namenserver der XML-Maschine nach TXT-Einträgen nach der Domain in der Adresse. Alle TXT-Rückantworten, die dem Adressantwortformat entsprechen, werden geparst, um Protokolladressen zu extrahieren. Das Adressenantwortformat ist
XMLR:<Präferenzzahl>:<Protokolladresse>

[0109] Die Präferenzzahl zeigt die Präferenzordnung für jede Protokolladresse an. Je niedriger die Präferenzzahl desto höher die Präferenz.

[0110] Der Antwortkanallieferant **303** stellt einen Antwortkanal **305** bereit, der in der Lage ist, mit der höchsten Präferenz, die vom verfügbaren Transport-Übermittler-Adapter **306** – falls vorhanden – unterstützt wird, an die Protokolladresse zu liefern.

[0111] Fig. 9 zeigt die Schritte zum Verarbeiten eines Ereignisses durch den Transportstromprozessor. Im Schritt **570** beginnt der Prozess und setzt zu **572** fort, an welchem Punkt eine Bestimmung durchgeführt wird, ob das vorliegende Ereignis ein "Öffnen"-Kommando ist, und, falls dies der Fall ist, setzt der Prozess zum Schritt **574** fort, wo ein Antwortkanal erhalten wird. Der Kanal wird zur Kanalliste (Schritt **576**) und zur Aktivliste (Schritt **578**) hinzugefügt, und der Prozess endet.

[0112] Wenn das vorliegende Ereignis nicht ein "Öffnen"-Kommando ist, entscheidet der Schritt **580**, ob es ein "Schließen"-Kommando ist. Falls dies der Fall ist, schließt der Stromprozessor den Antwortkanal und entfernt ihn von der Aktivliste und von der Kanalliste (Schritte **582**, **584**, bzw. **586**), und der Prozess endet.

[0113] Wenn ein Kommando noch nicht erkannt wurde, bestimmt der Schritt **588** ob das Kommando ein "Pause"-Kommando ist. Falls dies der Fall ist, entfernt der Schritt **590** den Kanal, der durch das Kommando spezifiziert wurde, von der Aktivliste, und der Prozess endet. Ansonsten entscheidet der Schritt **592**, ob das Ereignis ein "Wiederaufnahme"-Kommando ist. Falls dies der Fall ist, fügt der Schritt **594** den spezifizierten Kanal zur Aktivliste hinzu.

[0114] Wenn kein Kommando erkannt wurde, schreibt der Transportstromadapter auf den Zeichenstrom in jedem der Kanäle in der Aktivliste (Schritt **596**).

[0115] Schließlich wird das Ereignis zum nächsten Stromprozessor in der Kette, falls vorhanden, befördert (Schritte **598** und **600**).

Beispiele

[0116] Eine Anzahl von Szenarien wird unten dargestellt, um, wie folgt, als Beispiele einiger der hierin obenstehend diskutierten Blöcke und Funktionen zu dienen:
Wie zuvor hierin angemerkt, unter Rückbezug auf Fig. 2, erzeugen ein oder mehrere Stromsender **101** (außerhalb der Umgebung) XML-Dokumente oder leiten diese an das XML-Betriebssystem als einen Zeichenstrom **102** unter Verwendung eines Transportprotokolls weiter. Ein Beispiel solch eines Zeichenstroms ist durch Beispiel 1 vorgestellt, worin der Status eines Flugs überprüft wird, der aus Dublin zu einer Zeit startet, die im Beispiel mit anderen entsprechenden Parametern notiert ist.

Beispiel 1

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="14-4.xsl"?>
  <FLIGHT-EVENT>
    <ORIGIN>DUB</ORIGIN>
    <FLIGHT-TIME>Wed Mar 22 06:20:03 GMT 2000</FLIGHT-TIME>
    <FLIGHT-NUMBER>EI123</FLIGHT-NUMBER>
    <DESTINATION>HTW</DESTINATION>
    <STATUS>ONTIME</STATUS>
    <ARRIVAL-TIME>08:00</ARRIVAL-TIME>
    <TERMINAL>1</TERMINAL>
  </FLIGHT-EVENT>
```

[0117] Wie zuvor angemerkt, ist ein Ereignisstrom eine geordnete Abfolge von Ereignisroutinenaufrufen, welche ein Dokument darstellen. Ein Beispiel eines Ereignisstroms, der auf den vom Beispiel 1 bezogen ist, ist durch Beispiel 2 dargestellt.

Beispiel 2

```
setDocumentLocator
startDocument
processingInstruction: xml-stylesheet, type="text/xsl" href="14-4.xsl"
startElement: FLIGHT-EVENT
characters (5):

startElement: ORIGIN
characters (3): DUB
endElement: ORIGIN
characters (5):

startElement: FLIGHT-TIME
characters (28): Wed Mar 22 06:20:03 GMT 2000
endElement: FLIGHT-TIME
characters (5):

startElement: FLIGHT-NUMBER
characters (5): EI123
endElement: FLIGHT-NUMBER
characters (5):

startElement: DESTINATION
characters (3): HTW
endElement: DESTINATION
characters (5):

startElement: STATUS
characters (6): ONTIME
endElement: STATUS
characters (5):

startElement: ARRIVAL-TIME
characters (5): 08:00
endElement: ARRIVAL-TIME
characters (5):

startElement: TERMINAL
characters (1): 1
endElement: TERMINAL
characters (1):

endElement: FLIGHT-EVENT
endDocument
```

(characters – Zeichen)

[0118] Im Beispiel 3 wird ein privates Kommando mit zwei Argumenten in den Ereignisstrom eingefügt. Die private Information ist ein Kommando an den Transportstromprozessor, den Kanal (channel) "A" pausieren zu lassen (pause). Ein Interpreterstromprozessor schlägt ""xiam:channel" im Stromprozessor-Wörterbuch nach und befördert den Ereignisstrom als ein Ergebnis an den Transportstromprozessor

Beispiel 3

```
startElement: xiam:channel, attributes (2)...
...attribute: command, CDATA, pause
...attribute: key, CDATA, A
endElement: xiam:channel
```

[0119] Beispiel 4 enthält ein XML-Dokument in der Form eines Zeichenstroms. Es enthält zwei auszuführen-

de Kommandos; eines mit Parametern und eines ohne. Der Zeichenstrom wird in einen geeigneten Ereignisstrom konvertiert, welcher vom Instruktionsinterpreterstromprozessor interpretiert wird. Dieser Stromprozessor wird den relevanten Instruktionsnamen und Parameter extrahieren und die Instruktion aufrufen. Das erste Kommando konfiguriert den Adressenauflöser (XARP) unter Verwendung der Instruktion namens "Transport.XARP.Config", den DNS-Server 10.20.1.1 zu verwenden. Dieser Instruktionsname erscheint im Instruktions-Wörterbuch.

Beispiel 4

```
<xiam:action name="Transport.XARP.Config">
      <parameter name="Type" value="DNS"/>
      <parameter name="DNSServer" value="10.20.1.1"/>
      <parameter name="Comment">
            Configure XARP
      </parameter>
</xiam:action>
<xiam:action name="Transport.StartAdapter.Transmitter.FILE"/>
```

(action – Handlung)

[0120] Beispiel 5 ist ein Beispiel, wie der Interpreter Ereignisse entfernt.

Beispiel 5

```
setDocumentLocator
startElement: ALPHABET
startElement: xiam:channel, attributes
(2)...
...attribute: command, CDATA, pause
...attribute: key, CDATA, A
endElement: xiam:channel
characters (0):
startElement: LETTER
characters (0):
endElement: LETTER
```

```
setDocumentLocator
startElement: ALPHABET
characters (0):
startElement: LETTER
characters (0):
endElement: LETTER
```

[0121] Beispiel 6 zeigt eine Adressauflösung und eine SMTP-Verbindung.

Beispiel 6

[0122] Ein Stromprozessor möchte Informationen an die Organisation Xiam senden und weiss, dass die Adresse dieser Organisation XMLR://xiam.com ist. Er fragt nach einem Antwortkanal, wobei er die Adresse XMLR://xiam.com angibt.

[0123] Die DNS (Domain Name Service, Domainnamendienst)-Datenbank für die Zone "xiam.com" enthält diese Ressourcendatensätze:

```
in   txt   "XMLR:10:http://xmlr1.xiam.com"
in   txt   "XMLR:20:http://xmlr2.xiam.com"
in   txt   "XMLR:20:http://xmlr3.xiam.com"
in   txt   "XMLR:30:smtp://xmlr@xiam.com"
in   mx    10 mail.xiam.com.
```

in mx 20 backup.isp.net

[0124] Angenommen sei, dass keiner der im XML-Betriebssystem installierten Transport-Übermittler-Adapter unter Verwendung von HTTP senden kann, aber einer da ist, der unter Verwendung von SMTP senden kann.

[0125] Der Adressenauflöser umgeht die HTTP-Adressen im DNS und gibt die Protokolladresse zurück: SMTP://xmlr@xiam.com.

[0126] Der Adapter verbindet sich mit Port **25** (der Standardport für SMTP) auf dem Host mail.xiam.com und initiiert eine Nachricht zur Mailbox "xmlr@xiam.com".

[0127] Der Antwortkanallieferant gibt einen Antwortkanal zurück, der diese Verbindung zum Stromprozessor darstellt.

[0128] Jegliche Informationen, die vom Stromprozessor zu diesem Kanal gesendet werden, werden unter Verwendung von SMTP gesendet.

[0129] Wenn der Stromprozessor den Kanal schließt, beendet der Adapter die SMTP-Nachricht und schließt die Verbindung.

[0130] Beispiel 7 zeigt die Ereignisstrommanipulation.

Beispiel 7

Note	Interpreter Stream Processor	Distributor Stream Processor	Transport Stream Processor	Response Channel 1	Response Channel 2
1	start: xiam:distributor				
2	start: memo	start: memo	start: open John end: open start: open Mary end: open start: memo characters: Memo!	(open, to John) Memo!	(open, to Mary) Memo!
3	start: priority characters: Urgent end: priority	start: priority characters: Urgent end: priority			
4	start: body characters: Lunch? end: body	start: body characters: Lunch? end: body	start: body characters: Lunch? end: body	Lunch?	Lunch?
5	end: memo	end: memo	end: memo start: close John end: close start: close Mary end:close	(close)	(close)
6	end: xiam:distributor				

1 Der Interpreter antwortet auf dieses Ereignis, indem er den Ereignisstrom zum Verteilerstromprozessor (Distributor Stream Processor) richtet, welchen er im Stromprozessor-Wörterbuch findet. Er verbraucht das Start-Ereignis.

2 Der Verteiler antwortet auf den Start eines Memos, indem er den Ereignisstrom zum Transportstrompro-

zessor (Transport Stream Processor) richtet und Kommandos im Ereignisstrom einfügt. Er fügt auch ein Titelereignis ein. Der Transport öffnet zwei Antwortkanäle (Response Channels) und schreibt den Titel in beide.
3 Die Priorität (priority) ist nicht relevant, daher entfernt der Verteiler diese Ereignisse vom Strom.
4 Der Body des Memos tritt durch alle Stromprozessoren hindurch. Der Transport schreibt den Text in beide Kanäle.
5 Am Ende des Memos schließt der Transport die Kanäle.
6 Der Interpreter erkennt, dass der Verteiler nicht mehr erforderlich ist und stoppt das Senden von Ereignissen an diesen.

[0131] Die in den Fig. 1–Fig. 9 gezeigten Komponenten sind in einer bevorzugten erfindungsgemäßen Ausführungsform in Softwarecode implementiert. Dieselben können jedoch auch in Hardware implementiert sein, ohne vom Bereich der Erfindung abzuweichen.

[0132] Obwohl die vorliegende Erfindung hinsichtlich besonderer Ausführungsformen beschrieben wurde, wird vorweggenommen, dass Änderungen und Abweichungen hiervon dem Fachmann zweifellos offensichtlich werden. Es ist daher beabsichtigt, dass die folgenden Ansprüche dahingehend interpretiert werden, dass sie all solche Änderungen und Abweichungen, die in den wahren Umfang der Erfindung fallen, abdecken.

[0133] Die vorliegende Erfindung kann durch ein Computerprogramm implementiert werden, das auf einem Computer arbeitet. Ein erfindungsgemäßer Aspekt stellt daher ein Speichermedium bereit, das implementierbare Prozessorinstruktionen speichert, um einen Prozessor zum Ausführen des hierin oben beschriebenen Verfahrens zu steuern.

[0134] Ferner kann das Computerprogramm in elektronischer Form erhalten werden, indem der Code beispielsweise über ein Netz, wie beispielsweise das Internet, heruntergeladen wird. Entsprechend wird in Übereinstimmung mit einem anderen erfindungsgemäßen Aspekt ein elektrisches Signal, das implementierbare Prozessorinstruktionen zum Steuern eines Prozessors enthält, bereitgestellt, um das hierin oben beschriebene Verfahren auszuführen.

Patentansprüche

1. Betriebssystem (**10**) zur Verwendung in Netzumgebungen, um Informationen in der Form von Dokumenten zu verarbeiten, umfassend:
einen oder mehrere Transport-Empfänger-Adapter (**20**), um die Dokumente als empfangene Zeichenströme zu empfangen und, um ein oder mehrere Transportprotokolle zu implementieren, wobei die Transport-Empfänger-Adapter Mittel zum gleichzeitigen Empfangen von Zeichenströmen von mehreren Quellen umfassen;
einen Parser (**30**), um Informationen von jeder Zeichenkette zu extrahieren;
und einen Zuteilungsmechanismus, um dynamisch Instanzen von Elementen des Betriebssystems zu erzeugen und die Elemente zum Verarbeiten der Zeichenkette auf der Grundlage der extrahierten Informationen zu verbinden,
worin die Instanzen der Elemente für jedes Dokument umfassen:
eine Ereignisquelle (**16**), um auf der Grundlage der empfangenen Informationen strukturierte Informationen in der Form eines Ereignisstroms zu erzeugen;
einen oder mehrere Stromprozessoren (**14**), um den ein oder mehrere Ereignisse umfassenden Ereignisstrom zu verarbeiten; und
einen Transport-Übermittler-Adapter (**12**) zum Verbinden mit einem Empfänger, um die strukturierten Informationen in der Form eines Übermittlungszeichenstroms zu übermitteln.

2. Betriebssystem nach Anspruch 1, worin der Transport-Empfänger dazu ausgebildet ist, Informationen in der Form eines Extensible-Markup-Language-(erweiterbare Auszeichnungs-Sprache-) (XML-) Dokuments zu empfangen.

3. Betriebssystem nach Anspruch 2, umfassend einen XML-Sender (**101**), der dazu ausgebildet ist, das XML-Dokument zum Empfang durch den ein oder mehreren Transport-Empfänger-Adapter unter Verwendung eines Transportprotokolls als einen Zeichenstrom zu übermitteln.

4. Betriebssystem nach Anspruch 3, worin der eine oder mehrere Transport-Empfänger-Adapter dazu ausgebildet ist (sind), den Zeichenstrom zu empfangen und bei Beginn des Empfangs davon einen Anforderungskanal (**105**) zuzuteilen, um den Zeichenstrom darzustellen.

5. Betriebssystem nach Anspruch 4, worin der Anforderungskanal dazu ausgebildet ist, es Anforderungsprozessoren (**119**) zu gestatten, den Zeichenstrom ohne Kenntnis von Einzelheiten des Transportprotokolls zu lesen.

6. Betriebssystem nach Anspruch 4, umfassend Mittel zum Erzeugen des Anforderungskanals, wenn das Dokument durch den ein oder mehreren Transport-Empfänger-Adapter empfangen wird.

7. Betriebssystem nach Anspruch 4, das dazu ausgebildet ist, den Anforderungskanal zu verwerfen, wenn das Dokument verarbeitet wurde.

8. Betriebssystem nach Anspruch 6, umfassend Mittel zum Darstellen einer Vielzahl von Zeichenströmen durch eine Vielzahl von Anforderungskanälen, wobei jeder Zeichenstrom mit einem eindeutigen Dokument verbunden ist und die Dokumente in verschiedenen Stufen verarbeitet werden.

9. Betriebssystem nach Anspruch 4, worin jeder der ein oder mehreren Transport-Empfänger-Adapter dazu ausgebildet ist, Zeichenströme von verschiedenen Quellen gleichzeitig zu empfangen.

10. Betriebssystem nach Anspruch 4, worin der ein oder mehrere Transport-Empfänger-Adapter dazu ausgebildet ist, den dargestellten Zeichenstrom einem Anforderungskanal-Behandler zu übergeben.

11. Betriebssystem nach Anspruch 10, worin ein Anforderungsprozessor-Behandler dazu ausgebildet ist, einen Anforderungsprozessor auszuwählen, der einem bestimmten Anforderungskanal zuzuteilen ist, wobei der Anforderungsprozessor dazu ausgebildet ist, dann den dargestellten Zeichenstrom zu verarbeiten.

12. Betriebssystem nach Anspruch 11, worin der Anforderungsprozessor eine Ereignisquelle ist, die dazu ausgebildet ist, den verarbeiteten Zeichenstrom zur weiteren Verarbeitung in einen Ereignisstrom umzuwandeln.

13. Betriebssystem nach Anspruch 4, worin ein oder mehrere bestimmte der Stromprozessoren dazu ausgebildet sind, einen oder mehrere andere der Stromprozessoren aufzurufen.

14. Betriebssystem nach Anspruch 1, umfassend zwei oder mehr Ereignisquellen, wobei jede zur Erzeugung eines Ereignisstroms vorgesehen ist, um zwei oder mehr Dokumente gleichzeitig zu verarbeiten.

15. Betriebssystem nach Anspruch 14, das dazu ausgebildet ist, jedes Dokument durch einen anderen Stromprozessor zu verarbeiten.

16. Betriebssystem nach Anspruch 14, worin jeweilige Instanzen desselben Stromprozessors dazu ausgebildet sind, jedes Dokument zu verarbeiten.

17. Betriebssystem nach Anspruch 1, worin der Transport-Übermittler-Adapter dazu ausgebildet ist, einen ersten Zeichenstrom zu übertragen, während einer der ein oder mehreren Stromprozessoren einen mit einem zweiten Zeichenstrom verbundenen Ereignisstrom verarbeitet und, während ferner der Transport-Empfänger-Adapter einen dritten Zeichenstrom empfängt.

18. Betriebssystem nach Anspruch 1, umfassend Wörterbücher (**108**) mit Mustern oder Verweisen auf Muster, um Instanzen (**111**-**117**) des ein oder mehreren Stromprozessors für jedes der empfangenen Dokumente zu erzeugen.

19. Betriebssystem nach Anspruch 18, das dazu ausgebildet ist, die Instanzen auf Anforderung zu erzeugen.

20. Betriebssystem nach Anspruch 18, worin eines der Wörterbücher ein außerhalb des Betriebssystems gespeichertes Stromprozessor-Wörterbuch ist.

21. Betriebssystem nach Anspruch 1, worin:
jeder der ein oder mehreren Stromprozessoren Mittel zum Antworten auf ein Ereignis in einem Ereignisstrom durch Erfassen von Strukturen, Mustern, Namen oder Schlüsselwörtern im Ereignis;
zum Einfügen neuer Ereignisse in den Ereignisstrom;
zum Entfernen von Ereignissen aus dem Ereignisstrom;

zum Replizieren des Ereignisstroms oder Teilen davon, wobei jedes Ereignis an verschiedene Stromprozessoren gerichtet wird;
zum Kommunizieren mit oder Teilnehmen an einem externen System;
zum Ändern der Konfiguration des Dokumentenprozessors; oder
zum Aufrufen von eingebauten oder benutzerdefinierten Kommandos umfasst.

22. Verfahren zum Betrieb eines Betriebssystems (**10**) zur Verwendung in Netzumgebungen zum Verarbeiten von Informationen in der Form von Dokumenten, umfassend:
einen oder mehrere Transport-Empfänger-Adapter (**20**), der die Dokumente als empfangene Zeichenströme empfängt und ein oder mehrere Transportprototkolle implementiert, wobei die Transport-Empfänger-Adapter Mittel zum gleichzeitigen Empfangen von Zeichenströmen von mehreren Quellen umfassen;
einen Parser (**30**), der Informationen von jeder Zeichenkette extrahiert; und
einen Zuteilungsmechanismus, der dynamisch Instanzen von Elementen des Betriebssystems erzeugt und die Elemente zum Verarbeiten der Zeichenkette auf der Grundlage der extrahierten Informationen verbindet,
worin die Instanzen der Elemente für jedes Dokument umfassen:
eine Ereignisquelle (**16**), die auf der Grundlage der empfangenen Informationen strukturierte Informationen in der Form eines Ereignisstroms erzeugt;
einen oder mehrere Stromprozessoren (**14**), die den Ereignisstrom, der ein oder mehrere Ereignisse umfasst, verarbeiten; und
einen mit einem Empfänger verbundenen Transport-Übermittler-Adapter (**12**), der die Strukturinformationen in der Form eines Zeichenstroms übermittelt.

23. Verfahren nach Anspruch 22, worin die durch den Transport-Empfänger empfangenen Informationen ein Extensible-Markup-Language-Dokument sind.

24. Verfahren nach Anspruch 23, worin ein XML-Sender unter Verwendung eines Transportprotokolls das XML-Dokument als einen Zeichenstrom zum Empfang durch den einen oder mehreren Transport-Empfänger-Adapter übermittelt.

25. Verfahren nach Anspruch 24, worin der eine oder mehrere Transport-Empfänger-Adapter den Zeichenstrom empfängt und beim Beginn des Empfangs davon einen Anforderungskanal zuteilt, um den Zeichenstrom darzustellen.

26. Verfahren nach Anspruch 25, worin der Anforderungskanal es Anforderungsprozessoren ermöglicht, den Zeichenstrom ohne Kenntnis von Einzelheiten des Transportprotokolls zu lesen.

27. Verfahren nach Anspruch 25, worin ein Anforderungskanal erzeugt wird, wenn das Dokument durch den ein oder mehreren Transport-Empfänger-Adapter empfangen wird.

28. Verfahren nach Anspruch 25, worin der Anforderungskanal verworfen wird, wenn das Dokument verarbeitet wurde.

29. Verfahren nach Anspruch 27, worin eine Vielzahl von Anforderungskanälen eine Vielzahl von Zeichenströmen darstellen, wobei jeder Zeichenstrom mit einem eindeutigen Dokument verbunden ist und die Dokumente in verschiedenen Stufen verarbeitet werden.

30. Verfahren nach Anspruch 25, worin jeder der ein oder mehreren Transport-Empfänger-Adapter Zeichenströme von mehreren Quellen gleichzeitig empfängt.

31. Verfahren nach Anspruch 25, worin der ein oder mehrere Transport-Empfänger-Adapter den dargestellten Zeichenstrom einem Anforderungskanal-Behandler übergibt.

32. Verfahren nach Anspruch 31, worin ein Anforderungsprozessor-Behandler einen Anforderungsprozessor auswählt, der einem bestimmten Anforderungskanal zuzuteilen ist, wobei der Anforderungsprozessor den dargestellten Zeichenstrom verarbeitet.

33. Verfahren nach Anspruch 32, worin der Anforderungsprozessor eine Ereignisquelle ist, die den verarbeiteten Zeichenstrom zur weiteren Verarbeitung in einen Ereignisstrom umwandelt.

34. Verfahren nach Anspruch 25, worin ein oder mehrere bestimmte der Stromprozessoren einen oder

mehrere andere der Stromprozessoren aufruft.

35. Verfahren nach Anspruch 22, umfassend zwei oder mehr Ereignisquellen, von denen jede für das Erzeugen eines Ereignisstroms vorgesehen ist, um zwei oder mehr Dokumente gleichzeitig zu verarbeiten.

36. Verfahren nach Anspruch 35, worin jedes Dokument durch einen anderen Stromprozessor verarbeitet wird.

37. Verfahren nach Anspruch 35, worin jedes Dokument durch eine jeweilige Instanz desselben Stromprozessors verarbeitet wird.

38. Verfahren nach Anspruch 22, worin der Transport-Übermittler-Adapter einen ersten Zeichenstrom übermittelt, während einer der ein oder mehreren Stromprozessoren einen Ereignisstrom verarbeitet, der mit einem zweiten Zeichenstrom verbunden ist und während ferner der Transport-Empfänger-Adapter einen dritten Zeichenstrom empfängt.

39. Verfahren nach Anspruch 22, umfassend Wörterbücher mit Mustern oder Verweisen auf Muster, um Instanzen von den ein oder mehreren Stromprozessoren für jedes der empfangenen Dokumente zu erzeugen.

40. Verfahren nach Anspruch 39, worin die Instanzen auf Anforderung erzeugt werden.

41. Verfahren nach Anspruch 39, worin eines der Wörterbücher ein Stromprozessor-Wörterbuch ist, das außerhalb des Betriebssystems gespeichert ist.

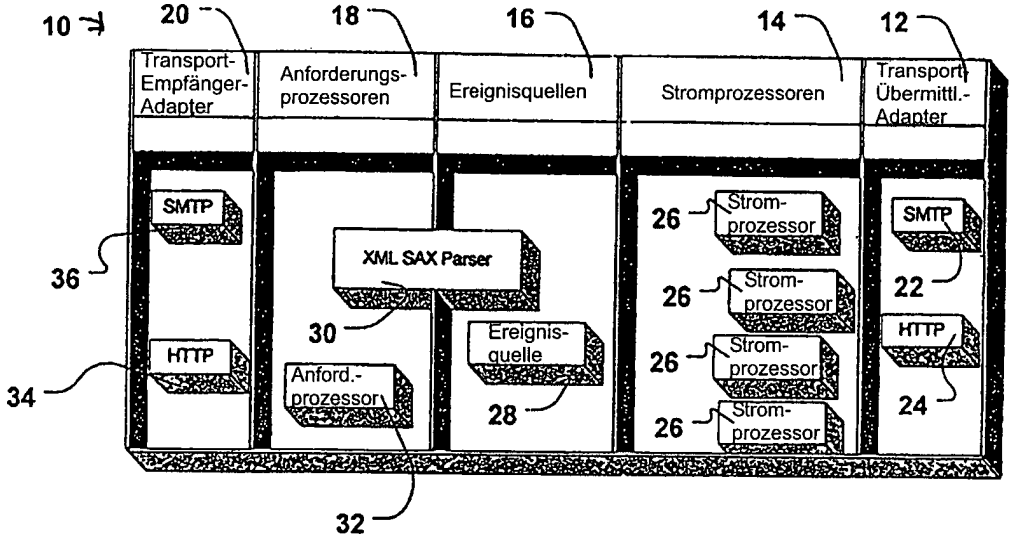
42. Verfahren nach Anspruch 22, worin:
jeder der ein oder mehreren Stromprozessoren auf ein Ereignis in einem Ereignisstrom durch Erfassen von Strukturen, Mustern, Namen oder Schlüsselwörtern in dem Ereignis antwortet;
neue Ereignisse in den Ereignisstrom einfügt;
Ereignisse von dem Ereignisstrom entfernt;
den Ereignisstrom oder Teile davon repliziert und jedes Ereignis an verschiedene Stromprozessoren richtet;
mit einem externen System kommuniziert oder an dem externen System teilnimmt;
die Konfiguration des Dokumentenprozessors ändert; oder eingebaute oder benutzerdefinierte Kommandos aufruft.

43. Speichermedium zum Speichern von implementierbaren Prozessorinstruktionen, um einen Prozessor zum Ausführen des Verfahrens nach einem der Ansprüche 22 bis 42 zu steuern.

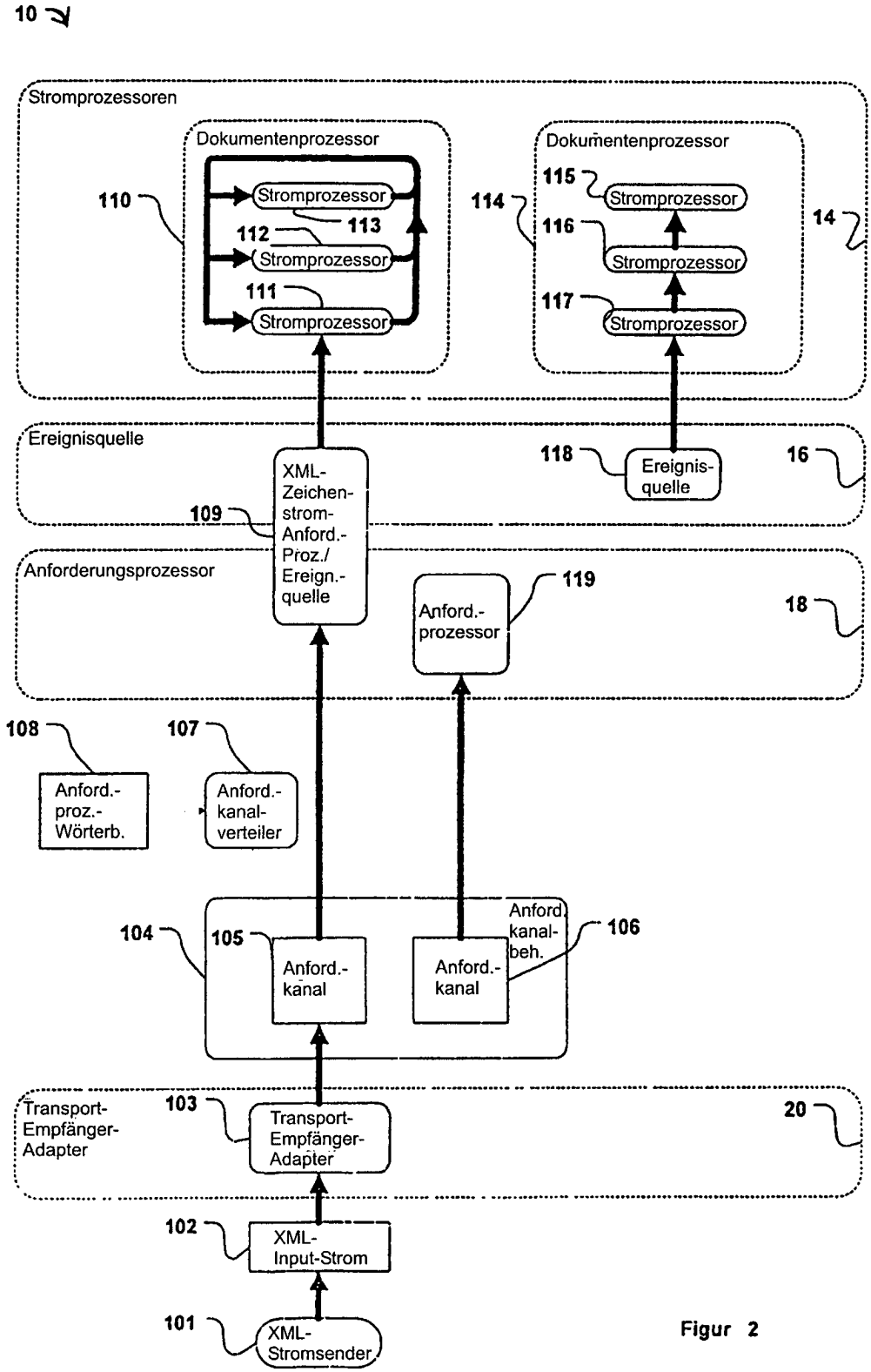
44. Elektrisches Signal zum Übertragen von implementierbaren Prozessorinstruktionen, um einen Prozessor zum Ausführen des Verfahrens nach einem der Ansprüche 22 bis 42 zu steuern.

Es folgen 9 Blatt Zeichnungen

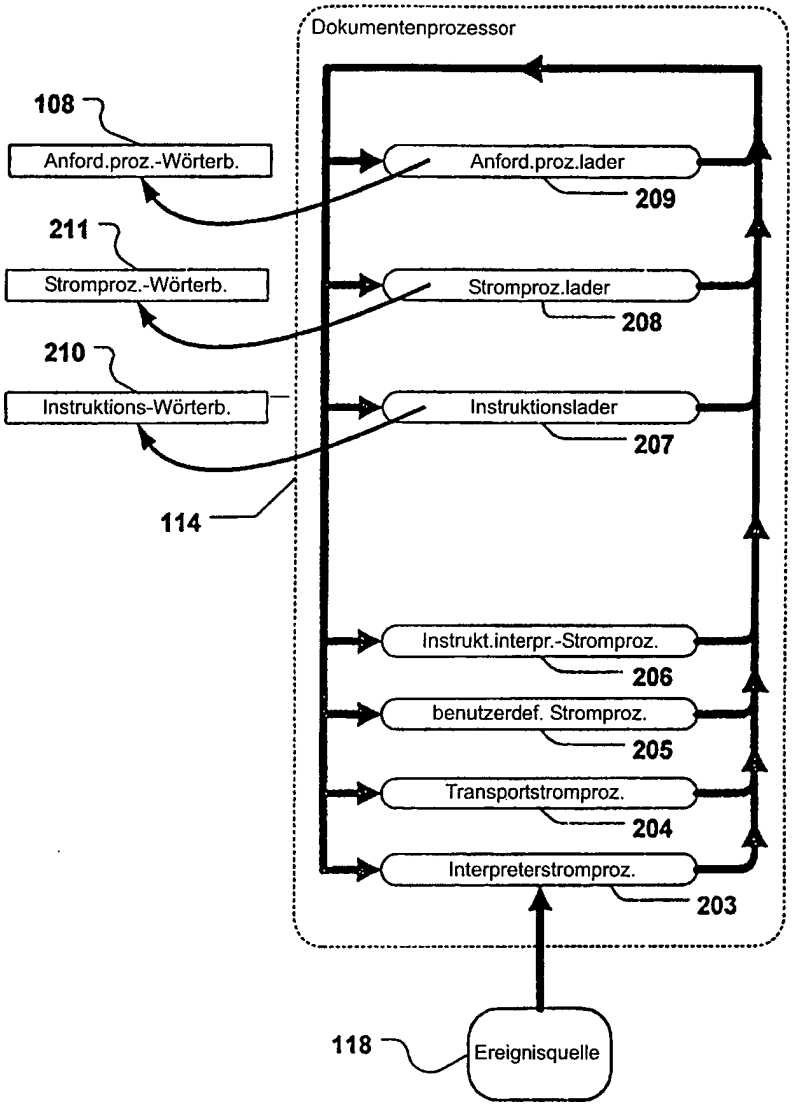
Anhängende Zeichnungen



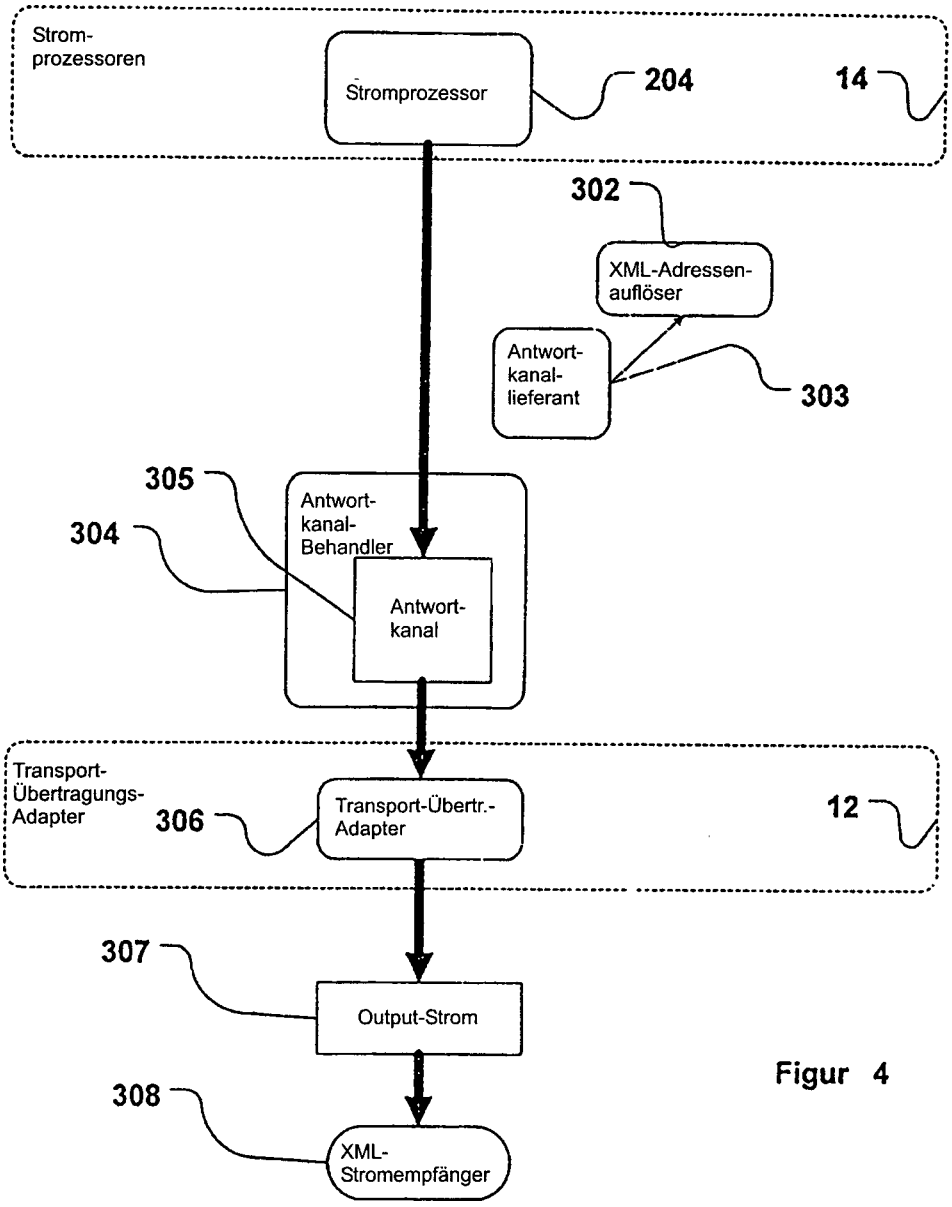
Figur 1


10
Stromprozessoren
Dokumentenprozessor
Dokumentenprozessor
110
Stromprozessor
113
112
Stromprozessor
111
Stromprozessor
114
115
Stromprozessor
116
Stromprozessor
117
Stromprozessor
14
Ereignisquelle
118
Ereignis-quelle
16
109
XML-Zeichen-strom-Anford.-Proz./ Ereign.-quelle
Anforderungsprozessor
Anford.-prozessor
119
18
108
Anford.-proz.-Wörterb.
107
Anford.-kanal-verteiler
104
105
Anford.-kanal
Anford.-kanal
Anford. kanal-beh.
106
Transport-Empfänger-Adapter
103
Transport-Empfänger-Adapter
20
102
XML-Input-Strom
101
XML-Stromsender

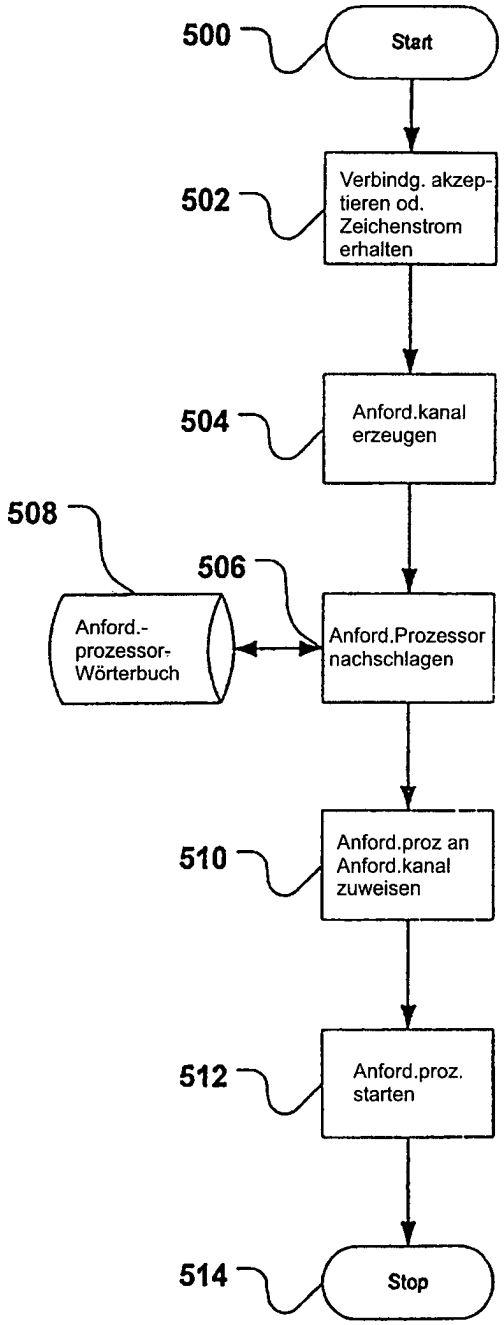
Figur 2


Dokumentenprozessor
108
Anford.proz.-Wörterb.
Anford.proz.lader
209
211
Stromproz.-Wörterb.
Stromproz.lader
208
210
Instruktions-Wörterb.
Instruktionslader
207
114
Instrukt.interpr.-Stromproz.
206
benutzerdef. Stromproz.
205
Transportstromproz.
204
Interpreterstromproz.
203
118
Ereignisquelle

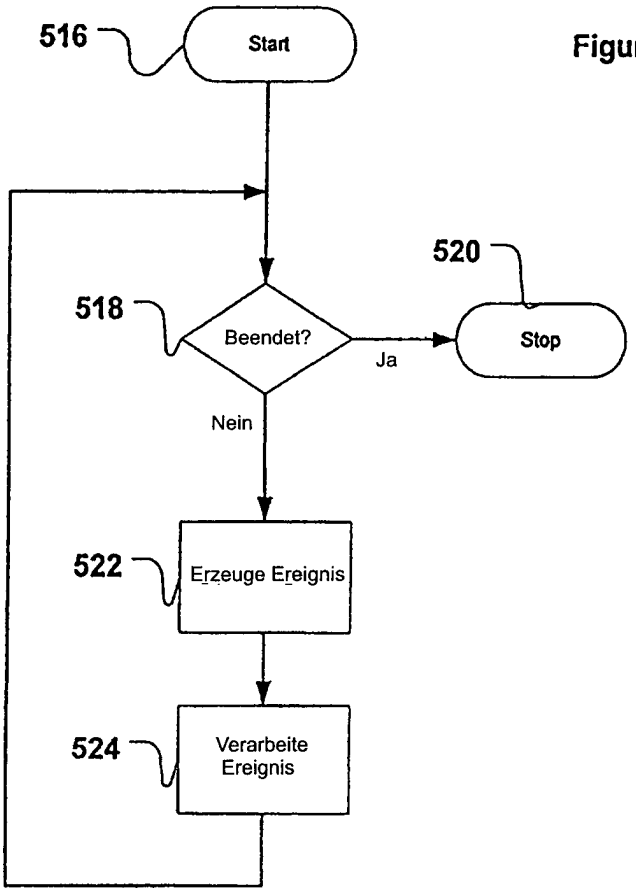
Figur 3


Strom-
prozessoren
Stromprozessor
204
14
302
XML-Adressen-
auflöser
Antwort-
kanal-
lieferant
303
305
304
Antwort-
kanal-
Behandler
Antwort-
kanal
Transport-
Übertragungs-
Adapter
306
Transport-Übertr.-
Adapter
12
307
Output-Strom
308
XML-
Stromempfänger

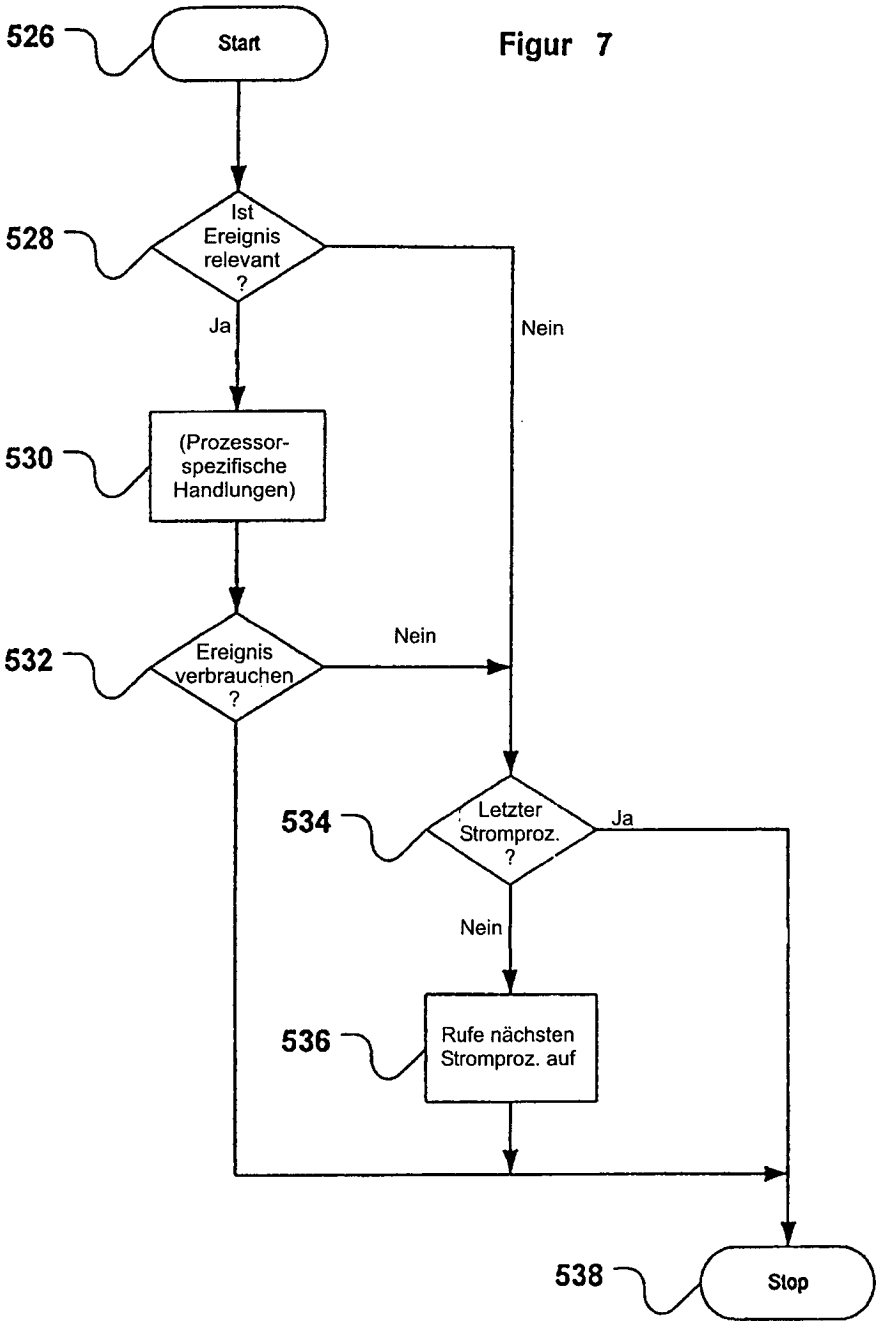
Figur 4


500
Start
502
Verbindg. akzeptieren od. Zeichenstrom erhalten
504
Anford.kanal erzeugen
508
506
Anford.-prozessor-Wörterbuch
Anford.Prozessor nachschlagen
510
Anford.proz an Anford.kanal zuweisen
512
Anford.proz. starten
514
Stop

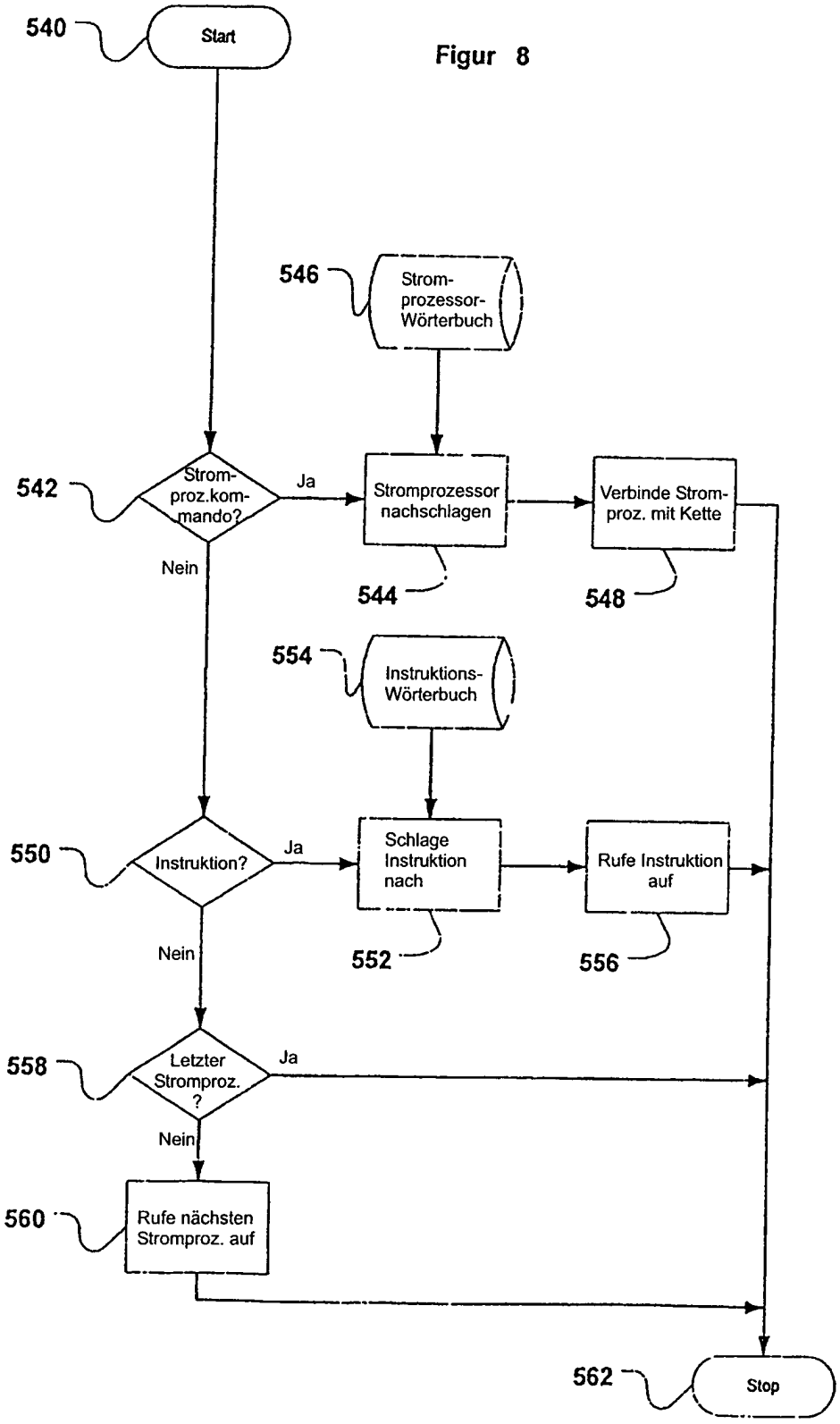
Figur 5


516
Start
520
518
Beendet?
Stop
Ja
Nein
522
Erzeuge Ereignis
524
Verarbeite Ereignis

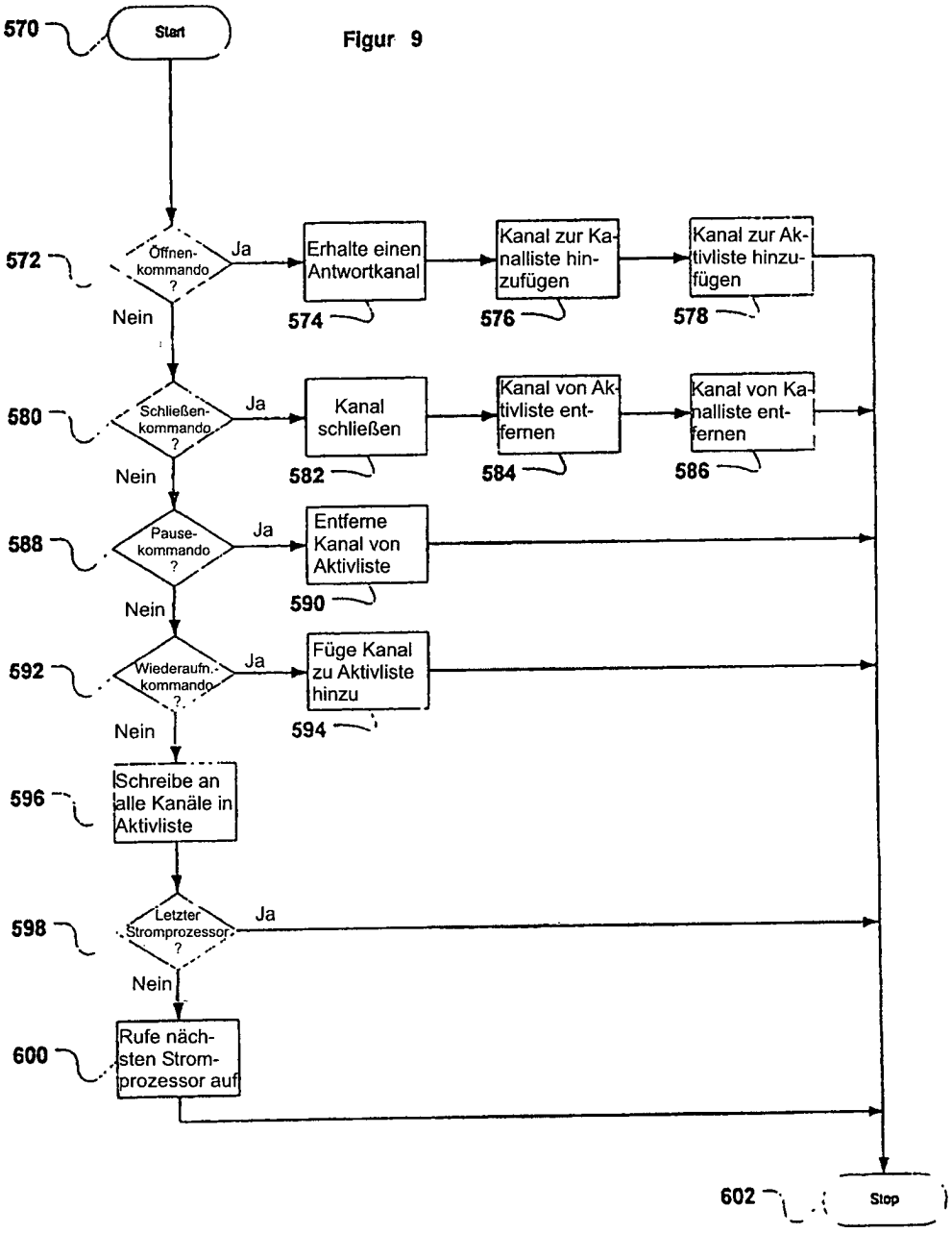
Figur 6


526
Start
528
Ist Ereignis relevant ?
Ja
Nein
530
(Prozessor-spezifische Handlungen)
532
Ereignis verbrauchen ?
Nein
534
Letzter Stromproz. ?
Ja
Nein
536
Rufe nächsten Stromproz. auf
538
Stop

Figur 7


540
Start
546
Strom-prozessor-Wörterbuch
542
Strom-proz.kom-mando?
Ja
Stromprozessor nachschlagen
Verbinde Strom-proz. mit Kette
Nein
544
548
554
Instruktions-Wörterbuch
550
Instruktion?
Ja
Schlage Instruktion nach
Rufe Instruktion auf
Nein
552
556
558
Letzter Stromproz. ?
Ja
Nein
560
Rufe nächsten Stromproz. auf
562
Stop

Figur 8


570
Start
572
Öffnen-kommando ?
Ja
Nein
Erhalte einen Antwortkanal
574
Kanal zur Kanalliste hinzufügen
576
Kanal zur Aktivliste hinzufügen
578
580
Schließen-kommando ?
Ja
Nein
Kanal schließen
582
Kanal von Aktivliste entfernen
584
Kanal von Kanalliste entfernen
586
588
Pause-kommando ?
Ja
Nein
Entferne Kanal von Aktivliste
590
592
Wiederaufn.-kommando ?
Ja
Nein
Füge Kanal zu Aktivliste hinzu
594
596
Schreibe an alle Kanäle in Aktivliste
598
Letzter Stromprozessor ?
Ja
Nein
600
Rufe nächsten Stromprozessor auf
602
Stop

Figur 9