(19) Deutsches Patent- und Markenamt

(10) **DE 11 2018 002 984 T5** 2020.02.27

(12) # Veröffentlichung

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2019/016628**
in der deutschen Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: **11 2018 002 984.0**
(86) PCT-Aktenzeichen: **PCT/IB2018/054458**
(86) PCT-Anmeldetag: **18.06.2018**
(87) PCT-Veröffentlichungstag: **24.01.2019**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **27.02.2020**

(51) Int Cl.: ***G06F 11/30*** (2006.01)
G06F 15/177 (2006.01)

(30) Unionspriorität:
15/653,676 **19.07.2017** **US**

(71) Anmelder:
International Business Machines Corporation, Armonk, N.Y., US

(74) Vertreter:
Richardt Patentanwälte PartG mbB, 65185 Wiesbaden, DE

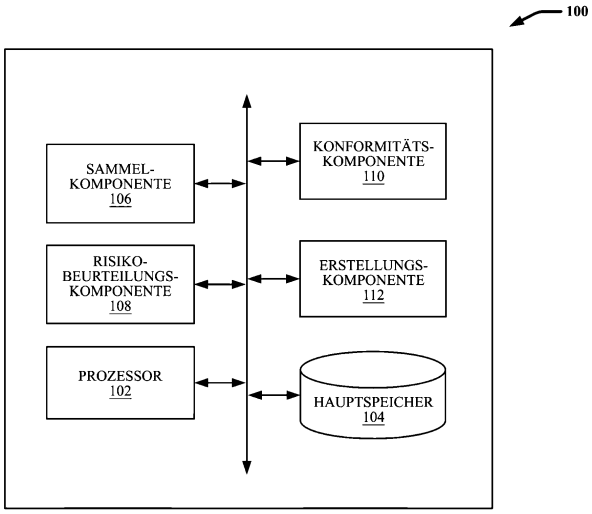
(72) Erfinder:
Adam, Constantin Mircea, Yorktown Heights, NY, US; Vukovic, Maja, Yorktown Heights, N.Y., US; Hwang, Jinho, Yorktown Heights, NY, US; Nadgowda, Shripad, Yorktown Heights, NY, US; Anerousis, Nikolaos, San Jose, CA, US

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Konformitätsbewusste Laufzeiterzeugung auf Grundlage von Anwendungsmustern und Risikobeurteilung**

(57) Zusammenfassung: Bereitgestellt werden Systeme, durch einen Computer umgesetzte Verfahren und/oder Computerprogrammprodukte, die konformitätsbewusste Laufzeiterzeugung von Containern ermöglichen. In einer Ausführungsform umfasst ein durch einen Computer umgesetztes Verfahren Folgendes: durch ein funktionsmäßig mit einem Prozessor verbundenes System erfolgendes Erkennen von Informationen, die von einer in Container einzubindenden Zielanwendung verwendet werden; Ermitteln, ob für die Informationen innerhalb eines oder mehrerer definierter Grenzwerte ein oder mehrere Risikoverstöße vorliegen; Ermitteln, ob in den Informationen ein Konformitäts- oder ein Sicherheitsverstoß vorliegt, wobei das Ermitteln, ob der Konformitäts- oder Sicherheitsverstoß vorliegt, auf Grundlage einer Feststellung eines Nichtvorliegens eines oder mehrerer Risikoverstöße durch die Risikobeurteilungskomponente erfolgt; und Erzeugen eines neuen Containers definierten Komponenten der Zielanwendung entsprechender Komponenten, die es der Zielanwendung ermöglichen, ohne ein zugrunde liegendes Betriebssystem ausgeführt zu werden, wobei das Erzeugen auf Grundlage einer Feststellung erfolgt, dass in den Informationen kein Konformitäts- oder Sicherheitsverstoß vorliegt.



Beschreibung

HINTERGRUND

[0001] Die vorliegende Erfindung betrifft ein Ermöglichen konformitätsbewusster Laufzeiterzeugung von Containern.

KURZDARSTELLUNG

[0002] Es folgt eine Kurzdarstellung, um ein Grundverständnis einer oder mehrerer Ausführungsformen der Erfindung zu vermitteln. Diese Kurzdarstellung soll keine Schlüsselelemente bzw. entscheidenden Elemente bezeichnen oder irgendeinen Schutzumfang der konkreten Ausführungsformen der Erfindung oder einen Schutzumfang der Ansprüche skizzieren. Ihr alleiniger Zweck besteht darin, in Vorausschau auf die nachstehend gegebene ausführlichere Beschreibung Konzepte in vereinfachter Form darzustellen. In einer oder mehreren Ausführungsformen werden Einheiten, Systeme, durch einen Computer umgesetzte Verfahren, Vorrichtungen und/oder Computerprogrammprodukte beschrieben, welche die konformitätsbewusste Laufzeiterzeugung von Containern ermöglichen.

[0003] Gemäß einer Ausführungsform der Erfindung wird ein System bereitgestellt. Das System kann einen Hauptspeicher aufweisen, der durch einen Computer ausführbare Komponenten speichert. Das System kann zudem einen Prozessor aufweisen, der funktionsmäßig mit dem Hauptspeicher verbunden ist und der im Hauptspeicher gespeicherte durch einen Computer ausführbare Komponenten ausführen kann. Die durch einen Computer ausführbaren Komponenten können eine Sammelkomponente aufweisen, die Informationen erkennen kann, die von einer in Container einzubindenden Zielanwendung verwendet werden. Die durch einen Computer ausführbaren Komponenten können ferner eine Risikobeurteilungskomponente aufweisen, die ermitteln kann, ob für die Informationen innerhalb eines oder mehrerer definierter Grenzwerte ein oder mehrere Risikoverstöße vorliegen. Die durch einen Computer ausführbaren Komponenten können ferner eine Konformitätskomponente aufweisen, die ermitteln kann, ob in den Informationen ein Konformitäts- oder ein Sicherheitsverstoß vorliegt, wobei die Feststellung durch die Konformitätskomponente auf Grundlage einer Feststellung eines Nichtvorliegens eines oder mehrerer Risikoverstöße durch die Risikobeurteilungskomponente erfolgt. Die durch einen Computer ausführbaren Komponenten können ferner eine Erstellungskomponente aufweisen, die auf Grundlage einer Feststellung, dass in den Informationen kein Konformitäts- oder Sicherheitsverstoß vorliegt, einen neuen Container erzeugen kann, der definierten Komponenten der Zielanwendung entspricht, die es der Zielanwendung ermöglichen, ohne ein zugrunde liegendes Betriebssystem ausgeführt zu werden.

[0004] Gemäß einer weiteren Ausführungsform der Erfindung wird ein durch einen Computer umgesetztes Verfahren bereitgestellt. Das durch einen Computer umgesetzte Verfahren kann durch ein funktionsmäßig mit einem Prozessor verbundenes System erfolgendes Erkennen von Informationen umfassen, die von einer in Container einzubindenden Zielanwendung verwendet werden. Das durch einen Computer umgesetzte Verfahren kann ferner durch das System erfolgendes Feststellen umfassen, ob für die Informationen innerhalb eines oder mehrerer definierter Grenzwerte ein oder mehrere Risikoverstöße vorliegen. Das durch einen Computer umgesetzte Verfahren kann ferner durch das System erfolgendes Ermitteln umfassen, ob in den Informationen ein Konformitäts- oder ein Sicherheitsverstoß vorliegt, wobei das Ermitteln, ob der Konformitäts- oder Sicherheitsverstoß vorliegt, auf Grundlage einer Feststellung eines Nichtvorliegens eines oder mehrerer Risikoverstöße durch die Risikobeurteilungskomponente erfolgt. Das durch einen Computer umgesetzte Verfahren kann ferner durch das System erfolgendes Erzeugen eines neuen Containers definierten Komponenten der Zielanwendung entsprechender Komponenten umfassen, die es der Zielanwendung ermöglichen, ohne ein zugrunde liegendes Betriebssystem ausgeführt zu werden, wobei das Erzeugen auf Grundlage einer Feststellung erfolgt, dass in den Informationen kein Konformitäts- oder Sicherheitsverstoß vorliegt.

[0005] Gemäß einer weiteren Ausführungsform der Erfindung wird ein Computerprogrammprodukt zum Ermöglichen einer konformitätsbewussten Laufzeiterzeugung von Containern bereitgestellt. Das Computerprogrammprodukt kann ein durch einen Computer lesbares Speichermedium mit auf diesem enthaltenen Programmanweisungen aufweisen. Die Programmanweisungen können durch einen Prozessor ausführbar sein, um den Prozessor zu veranlassen, Informationen zu erkennen, die von einer in Container einzubindenden Zielanwendung verwendet werden. Die Programmanweisungen können ferner durch einen Prozessor ausführbar sein, um den Prozessor zu veranlassen, zu ermitteln, ob für die Informationen innerhalb eines oder mehrerer definierter Grenzwerte ein oder mehrere Risikoverstöße vorliegen. Die Programmanweisungen können ferner durch einen Prozessor ausführbar sein, um den Prozessor zu veranlassen zu ermitteln, ob in den Informationen ein Konformitäts- oder ein Sicherheitsverstoß vorliegt, wobei die den Konformitäts- oder Sicherheitsverstoß betreffende Ermittlung auf Grundlage einer Feststellung eines Nichtvorliegens eines oder mehrerer Risikoverstöße durch die Risikobeurteilungskomponente erfolgt. Die Programmanweisungen können ferner durch einen Prozessor ausführbar sein, um den Pro-

zessor zu veranlassen, einen neuen Container definierten Komponenten der Zielanwendung entsprechender Komponenten zu erzeugen, die es der Zielanwendung ermöglichen, ohne ein zugrunde liegendes Betriebssystem ausgeführt zu werden.

Figurenliste

Fig. 1 ist ein Blockschaubild eines beispielhaften Systems, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglicht, gemäß einer oder mehreren Ausführungsformen der Erfindung;

Fig. 2 veranschaulicht ein Blockschaubild eines beispielhaften, nicht-einschränkenden Systems, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglicht, mit einer Zuordnungskomponente, gemäß einer oder mehreren vorliegend beschriebenen Ausführungsformen.

Fig. 3 veranschaulicht ein Blockschaubild eines beispielhaften, nicht-einschränkenden Systems, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglicht, mit einer Musterkomponente, gemäß einer oder mehreren vorliegend beschriebenen Ausführungsformen.

Fig. 4 veranschaulicht ein Blockschaubild eines beispielhaften, nicht-einschränkenden Systems, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglicht, mit einer Betriebsablaufkomponente, gemäß einer oder mehreren vorliegend beschriebenen Ausführungsformen.

Fig. 5 veranschaulicht ein beispielhaftes, nicht-einschränkendes durch einen Computer umgesetztes Verfahren, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglicht, gemäß einer oder mehreren vorliegend beschriebenen Ausführungsformen.

Fig. 6 veranschaulicht ein beispielhaftes, nicht-einschränkendes, durch einen Computer umgesetztes Verfahren, das Konformitäts- und Sicherheitsklassifizierung ermöglicht, gemäß einer oder mehreren vorliegend beschriebenen Ausführungsformen.

Fig. 7, **Fig. 8**, **Fig. 9** und **Fig. 10** veranschaulichen beispielhafte, nicht-einschränkende, durch einen Computer umgesetzte Verfahren, die konformitätsbewusste Laufzeiterzeugung von Containern ermöglichen, gemäß einer oder mehreren vorliegend beschriebenen Ausführungsformen.

Fig. 11 veranschaulicht ein Blockschaubild einer beispielhaften, nicht-einschränkenden Betriebsumgebung, in der eine oder mehrere vorliegend beschriebene Ausführungsformen ermöglicht werden können.

Fig. 12 zeigt eine Cloud-Computing-Umgebung, in der eine oder mehrere vorliegend beschriebene Ausführungsformen ermöglicht werden können.

Fig. 13 zeigt Abstraktionsmodellschichten, in denen eine oder mehrere vorliegend beschriebene Ausführungsformen ermöglicht werden können.

AUSFÜHRLICHE BESCHREIBUNG

[0006] Die nachfolgende ausführliche Beschreibung ist rein veranschaulichend und soll nicht Ausführungsformen der Erfindung und/oder Anwendung oder Verwendungen von Ausführungsformen der Erfindung einschränken. Des Weiteren besteht nicht die Absicht einer Bindung an jegliche explizit oder implizit in den vorangehenden Abschnitten zum Hintergrund und der Kurzdarstellung oder im Abschnitt der ausführlichen Beschreibung angegebene Informationen.

[0007] Nachfolgend werden eine oder mehrere Ausführungsformen der Erfindung unter Bezugnahme auf die Zeichnungen beschrieben, wobei gleiche Bezugszeichen durchgehend zur Bezeichnung gleicher Elemente verwendet werden. In der nachfolgenden Beschreibung werden zu Erläuterungszwecken zahlreiche konkrete Einzelheiten dargelegt, um ein umfassenderes Verständnis der Erfindung zu vermitteln. Offenkundig kann jedoch in verschiedenen Fällen die Erfindung ohne diese konkreten Einzelheiten umgesetzt werden.

[0008] Eine oder mehrere Ausführungsformen der vorliegend beschriebenen Erfindung können automatisch einen konformitätsbewussten Laufzeit-Container erzeugen, der es einer Anwendung (z.B. ausführbare Datei, Dienst, Dienstinstanz, Anwendungsinstanz usw.) ermöglichen kann, ohne ein zugrunde liegendes Betriebssystem zu laufen. Die Bezeichnungen „Anwendung“, „ausführbare Datei“, „Dienst“, „Dienstinstanz“, „Prozess“ und/oder „Anwendungsinstanz“ wie vorliegend verwendet können vorliegend synonym verwendet werden.

[0009] **Fig. 1** veranschaulicht ein Blockschaubild eines Systems **100**, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglichen kann. Aspekte von in dieser Offenbarung erläuterten Systemen (z.B. das System **100** und dergleichen), Vorrichtungen oder Prozessen können eine oder mehrere durch eine Maschine ausführbare Komponenten darstellen, die in einer oder mehreren Maschinen gespeichert sind, z.B. in einem oder mehreren durch einen Computer lesbaren Medien gespeichert sind, die zu einer oder mehreren Maschinen gehören. Solche Komponenten können, wenn sie durch die eine oder die mehreren Maschinen wie z.B. Computer, Datenverarbeitungseinheiten, virtuelle Maschinen usw. ausgeführt werden, die Maschinen veran-

lassen, die beschriebenen Arbeitsschritte durchzuführen.

[0010] In verschiedenen Ausführungsformen der Erfindung kann es sich bei dem System **100** um eine beliebige Art von Komponente, Maschine, Einheit, Einrichtung, Vorrichtung und/oder Apparat handeln, die bzw. der einen Prozessor aufweist. In einigen Ausführungsformen ist das System **100** zu wirksamer und/oder funktionsmäßiger Datenübertragung mit einem drahtgebundenen und/oder drahtlosen Netzwerk in der Lage. Zu Komponenten, Maschinen, Vorrichtungen, Einheiten, Einrichtungen und/oder Apparatschaften, die das System **100** aufweisen können, zählen, ohne jedoch hierauf beschränkt zu sein, Tablet-Datenverarbeitungseinheiten, Handgeräte, Server-Datenverarbeitungsmaschinen und/oder Datenbanken, Laptop-Computer, Notebook-Computer, Desktop-Computer, Mobiltelefone, Smartphones, Nutzergeräte und/oder -apparatschaften, industrielle und/oder kommerzielle Einheiten, digitale Assistenten, Multimedia-Internet-fähige Telefone, Multimedia-Spieler und dergleichen.

[0011] Wie in **Fig. 1** veranschaulicht, kann das System **100** einen Prozessor **102**, einen Hauptspeicher **104**, eine Sammelkomponente **106**, eine Risikobeurteilungskomponente **108**, eine Konformitätskomponente **110** und/oder eine Erstellungskomponente **112** aufweisen. In einigen Ausführungsformen der Erfindung können der Prozessor **102**, der Hauptspeicher **104**, die Sammelkomponente **106**, die Risikobeurteilungskomponente **108**, die Konformitätskomponente **110** und/oder die Konstruktionskomponente **112** kommunikativ und/oder funktionsmäßig miteinander verbunden sein, um eine oder mehrere Funktionen des Systems **100** auszuführen.

[0012] In einer oder mehreren Ausführungsformen des Systems **100** kann prädiktive Analytik verwendet werden, um einen oder mehrere konformitätsbewusste Laufzeit-Container automatisch zu erzeugen. Die automatische Erzeugung kann beispielsweise auf Grundlage von in einer Musterwissensbank aufbewahrten Informationen erfolgen. Bei der Bezeichnung „Wissensbank“ wie vorliegend verwendet kann es sich um eine Datenbank oder einen anderen Speicher- oder Aufbewahrungsort handeln, der eine oder mehrere Arten von Informationen speichern kann. Alle diese Ausführungsformen der Erfindung kommen in Betracht.

[0013] Die Musterwissensbank kann eine oder mehrere Anwendungen betreffende Informationen aufweisen. In einigen Ausführungsformen der Erfindung können die die eine oder die mehreren Anwendungen betreffenden Informationen im Zeitverlauf gesammelt und in der Musterwissensbank aufbewahrt werden. In einigen Ausführungsformen können die gesammelten Informationen für die Zielanwendung verwendete Risiko-, Konformitäts- und/oder Sicherheitsverstöße enthalten. Auf Grundlage der gewonnenen Informationen kann das System **100** beim oder nach dem Starten einer Anwendung die Musterwissensbank (oder mehrere Musterwissensbanken) auswerten und ein oder mehrere Muster erzeugen und/oder kann über die Zielanwendung bekannte Informationen den über andere Anwendungen bekannten Informationen zuordnen. Die prädiktive Analytik des Systems **100** kann feststellen, dass, falls Informationen der Zielanwendung einer oder mehreren anderen Anwendungen ähneln, die konformitätsbewussten Laufzeit-Container der ähnlichen Anwendungen genutzt werden können, um die Zielanwendung automatisch auszuführen.

[0014] Die vorliegend beschriebenen Computer-Verarbeitungssysteme, durch einen Computer umgesetzten Verfahren, Vorrichtungen und/oder Computerprogrammprodukte können Hardware und/oder Software einsetzen, um konformitätsbewusste Laufzeit-Container zu erzeugen, die hochgradig technischer Natur sind, nicht abstrakt sind und nicht als menschliche Gedankengänge ausführbar sind. Beispielsweise können die eine oder die mehreren Ausführungsformen der Erfindung an den verfügbaren Informationen die langwierige Interpretation und Analyse vornehmen, um zu ermitteln, welche Laufzeit-Container unter dem einen oder den mehreren Laufzeit-Containern für eine Zielanwendung genutzt werden sollten. In einem weiteren Beispiel können die eine oder die mehreren Ausführungsformen der Erfindung an einer großen Datenmenge prädiktive Analytik vornehmen, um selbst bei fehlender Detailkenntnis über die Zielanwendung eine konformitätsbewusste Laufzeiterzeugung von Containern mit hoher Genauigkeit automatisch zu ermöglichen. Die Genauigkeit kann durch Vergleichen eines Trainingssatzes mit einem Testsatz beurteilt werden. Nachdem ein Modell unter Verwendung eines Trainingssatzes trainiert wurde, kann die Genauigkeit mittels eines Testsatzes berechnet werden, indem ein Prozentsatz an Übereinstimmung einer Ausgabe, die durch das mit den Trainingssatzelementen laufende Modell erzeugt wird, mit einem vorhergesagten Ziel berechnet wird.

[0015] In verschiedenen Ausführungsformen der Erfindung kann die Sammelkomponente **106** Systeminformationen wie beispielsweise, ohne jedoch auf diese beschränkt zu sein, dynamische Bibliotheken (z.B. abhängige Dateien und Bibliotheken) und Systemaufrufe (z.B. Anfragen eines Prozesses nach Durchführung eines Dienstes) erkennen, die durch eine Zielanwendung im Einsatz verwendet werden. Beispielsweise kann in modernen Unix-basierten Betriebssystemen die Sammelkomponente **106** die Befehle „list dynamic dependencies (ldd)“ und „strace“ ausführen, um Abhängigkeiten (z.B. offene Dateien sowie Systemaufrufe, die bei oder nach dem Starten einer neuen Anwendung erfolgen) nachzuverfol-

gen und Fehler (z.B. Versionskonflikt, fehlende Dateien oder Bibliotheken usw.) zu finden. Beispielsweise kann die Sammelkomponente **106** aus einem Docker-Container mit einem Dienst, der in diesem eingesetzt wurde, Code aufbewahren, der den Einsatz einer Anwendung ermöglicht. Beispielsweise kann die Sammelkomponente **106** ein Docker-Abbild für eine Zielanwendung von einer Docker-Hub herunterladen.

[0016] Zudem kann in modernen Unix-basierten Betriebssystemen die Sammelkomponente **106** außerdem mittels der Befehle „chroot“ und „strace“ die Zielanwendung in einer Umgebung oder einem Ordner einer separaten Partition laufen lassen, um alle (oder in einigen Ausführungsformen einen oder mehrere) aktuell laufenden Codes zu erfassen, um Fehler zu finden. Die Sammelkomponente **106** kann eine Anwendung in einer Umgebung einer separaten Partition laufen lassen, indem sie die zum Laufenlassen der Anwendung eingesetzten Dateien in einem Ordner ablegt und den „chroot“-Befehl ausführt. Der „chroot“-Befehl kann ein Stammdateisystem in ein Verzeichnis einer separaten Partition verlegen. Falls nach Durchführen der Befehle „chroot“ und „strace“ ein Vorgang riskant ist, weil ein Fehler anzeigt, dass eine Datei oder Bibliothek fehlt, kann die Sammelkomponente **106** die fehlende Datei oder Bibliothek aus dem Docker-Abbild kopieren, falls die fehlende Datei lokalisiert werden kann. Das System **100** kann diesen Prozess wiederholt durchlaufen, bis die Zielanwendung in dem Ordner der separaten Partition läuft, indem die Dateien und Bibliotheken, die zum Laufenlassen einer bereits auf einem Betriebssystem installierten Anwendungsinstanz eingesetzt werden, in den Ordner der separaten Partition kopiert und/oder abgelegt werden.

[0017] Eine oder mehrere der Ausführungsformen der Erfindung können die Anzahl laufender Dateien, die typischerweise eingesetzt werden, drastisch verringern, um Konformitätsrichtlinien und/oder -regelungen zu erfüllen. Mit einer verringerten Anzahl an Dateien kann auch die Angreifbarkeit abnehmen. Die verringerte Abbildgröße kann in einer oder mehreren Ausführungsformen zu einer Verringerung des Netzwerkverkehrs führen. Zudem kann der verringerte Speicherplatzbedarf des Abbilds den Einsatz und/oder das Starten der Anwendung beschleunigen.

[0018] Anwendungen werden typischerweise in zwei Kategorien unterteilt: ausführbare Dateien, die direkt einen Satz von Systembibliotheken laden können, und Anwendungen, die eine Sprachenlaufzeit einsetzen. Für Anwendungen mit ausführbaren Dateien, die direkt einen Satz von Systembibliotheken laden können, können die vorstehend beschriebenen durch einen Computer umgesetzten Verfahren einen konformitätsbewussten Laufzeit-Container erzeugen. Zu Beispielen dieser Anwendungen, die ohne eine Laufzeit direkt laufen können, zählen, ohne jedoch auf diese beschränkt zu sein, nginx, MySQL, Redis usw. Für Anwendungen, die eine Sprachenlaufzeit einsetzen (z.B. Ruby, Python, Java, Erlang usw.), können die vorstehenden durch einen Computer umgesetzten Verfahren zudem ein Laufzeitabbild erstellen, und zusätzliche Logik kann sprachenspezifische Pakete (z.B. Ruby Gems, Python Wheels, Java JARs usw.) bestimmen, die auf dem Laufzeitabbild installiert werden können.

[0019] In einigen Ausführungsformen der Erfindung kann das System **100** einen konformitätsbewussten Laufzeit-Container erzeugen, indem als Eingabe eine bereits auf einem Betriebssystem installierte Anwendung verwendet wird. Die resultierende Laufzeitumgebung für den Dienst kann nichts außer den von dem Dienst für den Einsatz verwendeten Codes enthalten. Jede in dem konformitätsbewussten Laufzeit-Container enthaltene Datei und Bibliothek (und/oder in einigen Ausführungsformen mindestens eine oder mehrere Dateien und/oder Bibliotheken) kann auf Angreifbarkeit und/oder Konformität hin bewertet werden. Beim oder nach dem Erstellen von Container-Abbildern kann ein Risiko angemessen beurteilt werden, damit Container keine potentiell angreifbaren Bibliotheken einbeziehen. Beispielsweise kann die Risikobeurteilungskomponente **108** die Dateien und/oder Bibliotheken erfassen, die geöffnet werden und/oder auf die zugegriffen wird, während eine Anwendung in ihrer ursprünglichen Umgebung läuft, und die Dateien und/oder Bibliotheken auf Angreifbarkeiten analysieren sowie die Konfigurationsdateien auf Konformität analysieren. In einigen Ausführungsformen der Erfindung kann die Risikobeurteilungskomponente **108** dynamische und/oder statische Analyse von Anwendungseigenschaften durchführen, um zu ermitteln, ob zusätzliche Informationen (z.B. Dateien, Bibliotheken, Codes usw.) verwendet werden müssen, damit ein Container eine Zielanwendung einsetzen kann. Die Risikobeurteilungskomponente **108** kann die Machbarkeit von Anpassungen analysieren und ermitteln. Beispielsweise kann die Risikobeurteilungskomponente **108** ein oder mehrere Probleme ermitteln, die mit dem Laufenlassen einer Anwendung in einem erzeugten Container mit einem verringerten Satz aus Dateien und Bibliotheken verbunden sind.

[0020] Die Risikobeurteilungskomponente **108** kann ermitteln, ob für die Informationen innerhalb eines oder mehrerer definierter Grenzwerte ein oder mehrere Risikoverstöße vorliegen. Beispielsweise kann eine Risiko- oder Auswirkungsfunktion verwendet werden, um eine Höhe eines mit dem Durchführen einer Aktion oder dem Einsatz eines Dienstes verbundenen Risikos zu messen. Falls der Risikowert gleich einem oder größer als ein bestimmter definierter Grenzwert ist, kann dies darauf hinweisen, dass das System **100** die Zielanwendung möglicher-

weise nicht laufen lassen kann. Durch die Risikobeurteilungskomponente **108** erfasste Informationen können ausgewertet werden, um Risikoanalytik und/oder Musterbildung bereitzustellen, die beim Erzeugen von Containern genutzt werden können. Falls die Risikobeurteilungskomponente **108** anzeigt, dass ein einer Aktion innewohnendes Risiko größer als der oder gleich dem definierten Grenzwert ist, kann die Anwendung repliziert werden und/oder eine oder mehrere Aktionen simuliert werden, bevor ein neuer Container erstellt wird. Falls beispielsweise nach dem Durchführen der Befehle „chroot“ und „strace“ (z.B. über die Sammelkomponente **106**) einer Aktion ein Risiko innewohnt, das größer als ein oder gleich einem definierten Grenzwert ist, weil ein Fehler anzeigt, dass eine Datei oder Bibliothek fehlt, kann das System **100** die fehlende Datei oder Bibliothek aus dem Docker-Abbild kopieren, falls sich die fehlende Datei oder Bibliothek in dem Docker-Abbild befindet. Dieser Prozess kann sich wiederholen, bis die Zielanwendung in dem Ordner der separaten Partition laufen kann.

[0021] In einigen Ausführungsformen der Erfindung kann die Konformitätskomponente **110** ermitteln, ob in den Informationen ein Konformitäts- oder Sicherheitsverstoß vorliegt. Die Ermittlung durch die Konformitätskomponente **110** kann auf Grundlage einer Feststellung eines Nichtvorliegens eines oder mehrerer definierter Risikoverstöße durch die Risikobeurteilungskomponente **108** erfolgen. Wenn beispielsweise die Risikobeurteilungskomponente **108** feststellen kann, dass einer Aktion ein Risiko innewohnt, das nicht größer als ein und nicht gleich einem definierten Grenzwert ist (so dass die Aktion als nicht riskant erachtet wird), kann die Konformitätskomponente **110** unter Verwendung eines Algorithmus für aktives Lernen ermitteln, ob ein Konformitäts- oder Sicherheitsverstoß vorliegt. Falls ein Konformitäts- oder Sicherheitsverstoß gefunden wird, kann diese Information in einer Musterwissensbank gespeichert werden. Das Lernen kann als Resultat des Analysierens zweier Muster für eine oder mehrere Anwendungen und der entdeckten Sicherheitsprobleme erfolgen. Nach Bedarf können für eine bestimmte Anwendung mehr als zwei verschiedene Container erstellt werden. Durch Verwenden aktueller Aktivitäten zum Trainieren von Daten in einem Lernalgorithmus (z.B. Support Vector Machine (SVM)) können durch Analysieren von Unterschieden Muster gelernt und/oder verwendet werden, um Konformitäts- und/oder Sicherheitsauswertungen zu beschleunigen. Zudem kann ein Maschinenlernprozess des Systems **100** verwendet werden, um die Schritte beim Erzeugen eines konformitätsbewussten Laufzeit-Containers zu analysieren und den Prozess zu beschleunigen. Beispielsweise kann ein Maschinenlernprozess Muster erkennen und/oder kann bewerten, ob ein Codeabschnitt redundant, unnötig oder irrelevant ist. Muster und/oder Mustererkennung können verwendet werden, um den Prozess des Sammelns von Systeminformationen durch die Sammelkomponente **106** effizienter zu machen, wodurch Ressourcen und Zeit eingespart werden. Falls kein Konformitäts- oder Sicherheitsverstoß vorliegt, kann ein konformitätsbewusster Laufzeit-Container erzeugt werden.

[0022] Die Erstellungskomponente **112** kann einen neuen Container erzeugen, der in einigen Ausführungsformen eine Ausführung der Zielanwendung ohne ein zugrunde liegendes Betriebssystem ermöglicht. Die Erzeugung eines konformitätsbewussten Laufzeit-Containers kann auf Grundlage einer Feststellung erfolgen, dass in den Informationen (z.B. Dateien, Bibliotheken, Codes usw., die zum Laufenlassen einer Zielanwendung verwendet werden) kein Konformitäts- oder Sicherheitsverstoß vorliegt. Das Fehlen eines Betriebssystems in einem Container kann die Notwendigkeit der Durchführung von Konformitätsbewertungen wie Passwortverwaltung und Betriebssystemressourcen beseitigen. Dies kann auch automatisch verhindern, dass Benutzer zusätzliche Dienste wie beispielsweise Secure Socket Shell (SSH) in einem Container installieren, welche den Container anderen potentiellen Sicherheitslücken aussetzen können.

[0023] Genauer gesagt, die Erstellungskomponente **112** kann Container erzeugen und kann in einigen Ausführungsformen nur die Dateien und Bibliotheken einbeziehen, die von einem Container verwendet werden, um einen bestimmten Dienst oder eine bestimmte Anwendung laufen zu lassen. Nachdem festgestellt wurde, dass kein Konformitäts- oder Sicherheitsverstoß vorliegt (z.B. über die Risikobeurteilungskomponente **108** und die Konformitätskomponente **110**) und die Zielanwendung in einem Ordner einer separaten Partition läuft (z.B. über die Sammelkomponente **106**), kann die Erstellungskomponente **112** die zum Laufenlassen der Anwendung verwendeten Dateien und Bibliotheken in eine Docker-Datei kopieren. Entsprechend kann in einigen Ausführungsformen der Erfindung die Erstellungskomponente **112** die Docker-Datei verwenden, um Container-Abbilder mit garantierter Konformität und minimalem Sicherheitsrisiko zu erzeugen. Dieselben Komponenten und/oder Prozesse, die zum Erzeugen dieser Container-Abbilder mit garantierter Konformität verwendet werden, können für alle laufenden Container verwendet werden, um Risiko-, Konformitäts- und/oder Sicherheitsverstöße zu beurteilen.

[0024] In einigen Ausführungsformen der Erfindung sind diese per konformitätsbewusster Laufzeit erzeugten Container zudem unveränderbar. In einigen Ausführungsformen ist ein Eindringen in und/oder Modifizieren von unveränderbaren Containern nicht möglich, da keine Shells vorhanden sind. Somit kann es sich bei dem ursprünglich eingesetzten Abbild um das Abbild handeln, das laufen wird, und es ist nicht

ohne Weiteres möglich, die Struktur dieses unveränderbaren, konformitätsbewussten Laufzeit-Containers zu verändern. Diese erzeugten Container können es ermöglichen, dass die Anwendung läuft, ohne dass es hierfür eines zugrunde liegenden Betriebssystems bedarf. Ein Container ohne ein zugrunde liegendes Betriebssystem kann auch ein geringeres Angriffsrisiko bzw. verringerte Angriffsfläche bieten, da ohne ein zugrunde liegendes Betriebssystem weniger Dateien und Bibliotheken vorhanden sind, die modifiziert (d.h. angegriffen) werden können. Eine geringere Angriffsfläche kann wiederum konformitätsbezogene Kosten senken und die Sicherheit erhöhen. Das Ergebnis kann eine starke Senkung der Kosten zum Betreiben eines Dienstes sein, indem der Fokus auf Konformität für die Anwendung selbst gerichtet wird und die Kosten zum Konfigurieren und Sichern eines komplexen zugrunde liegenden Betriebssystems vermieden werden.

[0025] Fig. 2 veranschaulicht ein Blockschaubild eines Systems **200**, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglicht, mit einer Zuordnungskomponente **202**. Das System **200** kann auch die Zuordnungskomponente **202** enthalten. Die Zuordnungskomponente **202** kann automatisch ein oder mehrere Merkmale einem oder mehreren Systembetriebsabläufen oder -bibliotheken zuordnen. Die Zuordnung eines oder mehrerer Systembetriebe oder -bibliotheken kann ein Zuordnen eines oder mehrerer konformitäts- und sicherheitsbezogener Merkmale umfassen. Beispielsweise kann die Zuordnungskomponente **202** einer durch die Risikobeurteilungskomponente **108** und die Konformitätskomponente **110** beurteilten gelernten konformitätsbewussten Laufzeit zugehörige Systeminformationen zuordnen.

[0026] Fig. 3 ist ein Blockschaubild eines Systems **300**, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglicht, mit einer Musterkomponente. Die Musterkomponente **302** kann ein oder mehrere Muster erzeugen, die verwendet werden, um ein oder mehrere konformitätsbewusste Laufzeit-Container-Abbilder automatisch zu erstellen. Die Erzeugung des einen oder der mehreren Muster kann auf Grundlage einer Feststellung durch die Risikobeurteilungskomponente **108**, dass für die Informationen ein oder mehrere Risikoverstöße vorliegen, und/oder auf Grundlage einer Feststellung durch die Konformitätskomponente **110** erfolgen, dass in den Informationen Konformitäts- oder Sicherheitsverstöße vorliegen. Falls beispielsweise die Risikobeurteilungskomponente **108** feststellt, dass eine Aktion einen Risikowert aufweist, der gleich einem oder größer als ein definierter Schwellenwert ist, kann die Musterkomponente **302** ein oder mehrere Muster erzeugen, die verwendet werden können, um ein oder mehrere Container-Abbilder zu erstellen. Zudem kann, falls die Konformitätskomponente **110** feststellt, dass in den Informationen Konformitäts- oder Sicherheitsverstöße vorliegen, die Musterkomponente **302** ein oder mehrere Muster erzeugen, die verwendet werden, um ein oder mehrere Container-Abbilder zu erstellen.

[0027] Fig. 4 ist ein Blockschaubild eines Systems **400**, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglicht, mit einer Betriebsablaufkomponente **402**. Die Betriebsablaufkomponente **402** kann einen Prozess des Betriebssystems anhalten. In einigen Ausführungsformen kann die Betriebsablaufkomponente **402** auf Grundlage von durch die Musterkomponente **302** erzeugten Mustern ermitteln, wann ein Prozess des Betriebssystems angehalten werden soll. Durch Referenzieren der durch die Musterkomponente **302** erzeugten Muster, die auf Grundlage einer Feststellung, dass ein oder mehrere Risikoverstöße vorliegen (z.B. über die Risikobeurteilungskomponente **108**), oder auf Grundlage einer Feststellung erzeugt werden können, dass in den Informationen Konformitäts- und Sicherheitsverstöße vorliegen (z.B. über die Konformitätskomponente **110**), kann die Betriebsablaufkomponente **402** das Betriebssystem anhalten, falls ein Risiko-, Konformitäts- oder Sicherheitsverstoß vorliegt. Genauer gesagt, die Betriebsablaufkomponente **402** kann auf durch die Musterkomponente **302** analysierte Informationen über Risiko-, Konformitäts- und/oder Sicherheitsverstöße zugreifen, um Benutzer zu warnen, dass einer Aktion ein Risiko innewohnt, das größer als ein oder gleich einem definierten Grenzwert ist, dass die Aktion Konformitätsverstöße aufweist oder dass sie Sicherheitsverstöße aufweist.

[0028] Zusammen können die Komponenten des Systems **100**, **200**, **300** und/oder **400** miteinander kommunizieren, um einen konformitätsbewussten Laufzeit-Container zu erzeugen und/oder die durch den Prozess gelernten Informationen zur künftigen Verwendung zu sammeln. Die Sammelkomponente **106** kann von einer Zielanwendung verwendete Informationen erkennen, und diese Informationen können von der Risikobeurteilungskomponente **108** und der Konformitätskomponente **110** verwendet werden, um Risiko, Konformität und Sicherheit betreffende Probleme zu beurteilen. Risiko-, Konformitäts- und/oder Sicherheitsbeurteilung können für sowohl dynamische als auch statische Analysen der Anwendungseigenschaften kontinuierlich erfolgen. Die durch die Risikokomponente **108** und die Konformitätskomponente **110** analysierten Informationen können von der Zuordnungskomponente **202** genutzt werden, um Zuordnungen zu erzeugen, und können von der Musterkomponente **302** genutzt werden, um Muster zu erzeugen. Die Betriebsablaufkomponente **402** kann die auf Grundlage von Risiko-, Konformitäts- und Sicherheitsverstößen erzeugten Muster verwenden, um Prozesse des Betriebssystems anzuhalten.

[0029] **Fig. 5** veranschaulicht ein durch einen Computer umgesetztes Verfahren **500**, das konformitätsbewusste Laufzeiterzeugung von Containern ermöglicht. Das Verfahren **500** verwendet nginx als Beispiel einer ausführbaren Datei, die direkt einen Satz Systembibliotheken laden kann, oder einer ausführbaren Datei, die direkt ohne Laufzeit laufen kann. Bei **510** kann das Verfahren **500** Erkennen einer in Container einzubindenden Zielanwendung (z.B. über die Sammelkomponente **106**) umfassen.

[0030] Bei **520** kann das Verfahren **500** Sammeln von Systeminformationen wie beispielsweise dynamischen Bibliotheken und Systemaufrufen umfassen, die für die Zielanwendung verwendet werden (z.B. über die Sammelkomponente **106**). Systeminformationen können durch Ausführen des Idd-Befehls zum Erkennen unmittelbarer Abhängigkeiten für die ausführbare Datei ermittelt werden. Die Erkennung kann auch Herunterladen eines Docker-Abbilds für die Zielanwendung von der Docker Hub und Ausführen der „chroot"- und „strace"-Befehle beinhalten, um alle aktuell laufenden Codes zu erfassen, um Fehler zu finden. Dieser Prozess kann durch Erstellen einer Umgebung oder eines Ordners einer separaten Partition abgeschlossen werden, in der bzw. dem durch Ausführen des „chroot"-Befehls eine ausführbare Datei laufen kann. Die in dem Ordner der separaten Partition laufende ausführbare Datei reagiert so, als sei sie das Stammverzeichnis des Dateisystems. Falls nach Durchführen der Befehle „chroot" und „strace" ein Fehler anzeigt, dass eine Datei oder Bibliothek fehlt, wird die fehlende Datei oder Bibliothek aus dem Docker-Abbild kopiert, falls sich die fehlende Datei oder Bibliothek dort befindet. Bei diesem Prozess kann es sich um einen sich wiederholenden Prozess handeln, der wiederholt wird, bis die ausführbare Datei in dem Ordner der separaten Partition laufen kann. Das Resultat ist ein Ordner einer separaten Partition mit nichts außer einem zum Laufenlassen des Codes verwendeten Satz aus Dateien und Bibliotheken. An einem Punkt, an dem die verwendeten Dateien und Bibliotheken in dem Ordner der separaten Partition laufen, kann das Verfahren **500** Kopieren der zum Laufenlassen der Zielanwendung verwendeten Dateien und Bibliotheken in eine neue Docker-Datei umfassen. Die neue Docker-Datei kann verwendet werden, um ein neues Docker-Abbild mit einer konformitätsbewussten Laufzeit zu erzeugen. Bei **530** kann das Verfahren **500** Durchführen einer Risikobeurteilung an den Bibliotheken und Systemaufrufen (z.B. über die Risikobeurteilungskomponente **108**) umfassen. Eine Risiko- oder Auswirkungsfunktion kann verwendet werden, um die potenziellen Risiken eines Ausführens der Anwendung mit einem verringerten Satz aus Dateien und Bibliotheken zu beurteilen.

[0031] Ein nicht-einschränkendes Beispiel einer beispielhaften Risiko- oder Auswirkungsfunktion ist $R(\theta, \delta) = E_\theta L(\theta,\delta(X)) = \int_X L(\theta,\delta(X))dP_\theta(X)$, wobei es sich bei der Risikofunktion R um einen Wert handelt, der von 0 bis 1 variiert. θ ist ein einen festen Wert aufweisender, möglicherweise unbekannter Naturzustand. X ist ein Vektor aus Beobachtungen, die stochastisch aus einer Population gezogen werden, z.B. vorherigen Anwendungs- oder Serverprofilen, einer Liste zugehöriger Upgrade-Vorgänge, Versionen, verfügbarer Dienstverwaltungsfunktion usw. E_θ ist die Erwartung über alle Populationswerte von X. dP_θ ist ein Wahrscheinlichkeitsmaß über den Ereignisraum von X, parametrisiert durch θ, und das Integral wird über die gesamte Unterstützung von X beurteilt. Falls der Risikowert einen definierten Grenzwert überschreitet, kann für eine Sicherheitsprüfung oder eine Risikoanalyse eine Anwendung repliziert und/oder Aktionen simuliert werden.

[0032] Bei **540** kann das Verfahren **500** Ermitteln umfassen, ob ein Risikoverstoß gefunden wird (z.B. über die Risikobeurteilungskomponente **108**). Eine Risikoanalyse kann durch Vergleichen des Risikowerts mit einem definierten Grenzwert durchgeführt werden. Falls ein Risikoverstoß gefunden wird, kann das Verfahren **500** bei **542** Eingeben von Risikoverstoßinformationen in eine Musterwissensbank umfassen (z.B. über die Risikobeurteilungskomponente). Falls kein Risikoverstoß gefunden wird, kann das Verfahren **500** bei **550** Durchführen einer Konformitäts- und Sicherheitsbeurteilung an den Bibliotheken und Systemaufrufen umfassen (z.B. über die Konformitätskomponente **110**). Konformität und Sicherheit können durch Verwenden eines Algorithmus für aktives Lernen ermittelt werden, wie nachfolgend in **Fig. 6** näher ausgeführt. Anschließend kann das Verfahren **500** bei **560** mit einem vorab bestimmten Grenzwert erfolgendes Ermitteln umfassen, ob Konformitäts- und Sicherheitsverstöße gefunden werden (z.B. über die Konformitätskomponente **110**). Falls ein Konformitäts- und Sicherheitsverstoß gefunden wird, kann das durch einen Computer umgesetzte Verfahren **500** bei **542** Eingeben der Konformitäts- und Sicherheitsverstoßinformationen in eine Musterwissensbank umfassen (z.B. über die Konformitätskomponente **110**). Falls kein Konformitäts- und Sicherheitsverstoß gefunden wird, kann das Verfahren **500** bei **570** Erzeugen eines konformitätsbewussten Containers umfassen (z.B. über die Erstellungskomponente **112**). Anschließend kann das Verfahren **500** bei **580** das Durchführen eines Einsatztests umfassen.

[0033] Die verschiedenen Aspekte (z.B. in Verbindung mit dem automatischen Beurteilen von Konformitäts- und Sicherheitsverstößen) können verschiedene auf künstlicher Intelligenz basierende Verfahrensweisen einsetzen, um verschiedene Aspekte davon auszuführen. Beispielsweise kann ein Prozess zum Bewerten eines oder mehrerer Parameter einer Zielanwendung genutzt werden, um ohne Interaktion seitens der Zielanwendung eine oder mehrere Reak-

tionen auf die Beurteilung vorherzusagen, was durch einen Algorithmus für aktives Lernen ermöglicht werden kann. Eine Support Vector Machine (SVM) ist ein Beispiel eines Klassifikators, der verwendet werden kann. Die SVM arbeitet mit dem Finden einer Hyperfläche in dem Raum möglicher Eingaben, wobei die Hyperfläche versucht, die auslösenden Kriterien von den nicht-auslösenden Ereignissen zu trennen. Die Klassifikation ist hierdurch offenkundig korrekt für Testdaten, die Trainingsdaten zwar ähneln können, aber nicht notwendigerweise mit diesen identisch sind. Es können auch andere gerichtete und ungerichtete Klassifikationsmodellansätze (z.B. naiv-Bayes, Bayessche Netze, Entscheidungsbäume, neuronale Netze, Unschärfelogikmodelle und probabilistische Klassifizierungsmodelle) verwendet werden, die andere Unabhängigkeitsmuster bereitstellen. Klassifizierung im vorliegenden Sinne kann statistische Regression einschließen, die genutzt wird, um Prioritätsmodelle zu entwickeln.

[0034] Fig. 6 veranschaulicht ein durch einen Computer umgesetztes Verfahren **600**, das Konformitäts- und Sicherheitsklassifizierung ermöglicht. Bei dem Verfahren **600** kann es sich um einen Mehrfachmarkierungs-SVM-basierten Algorithmus für aktives Lernen handeln, der Konformitäts- und Sicherheitsklassifikation ermöglicht. In einigen Ausführungsformen der Erfindung kann das Verfahren **600** als Eingabe einen markierten Satz D_l, einen unmarkierten Satz D_u, eine Anzahl an Schritten T und eine Anzahl an Beispielen pro Wiederholung verwenden. Das Verfahren **600** kann einen Mehrfachmarkierungs-SVM-Klassifikator f auf Grundlage von Trainingsdaten D_l trainieren. Beispielsweise kann für jede Instanz x in dem unmarkierten Satz D_u das Verfahren **600** deren Markierungsvektor y mittels des auf Verlustminderung (LR, Loss Reduction) basierenden Vorhersageverfahrens mit einer Gleichung für maximale Verlustminderung mit maximaler Konfidenz vorhersagen. Das Verfahren **600** kann die erwartete Verlustminderung mit dem Markierungsvektor y mit der höchsten Konfidenz berechnen, den Wert x in absteigender Ordnung für alle x in D_u sortieren und einen Satz von Beispielen mit den höchsten Werten auswählen. Die Auswahl kann auf Grundlage einer Experten- (SME-, Subject Matter Expert) Eingabe erfolgen. Bei der Funktion $f_i(x)$ handelt es sich um einen der Klasse i zugehörigen SVM-Klassifikator. Die Datenpunkte $x_1,...x_n$ stellen einen Merkmalsvektor für jedes x (Bibliotheken, Systemaufrufe, Betriebssysteminformationen, Konfigurationsinformationen, Anwendungsinformationen usw.) und Konformitäts- und Sicherheitskompatibilität dar.

[0035] Fig. 7 veranschaulicht ein durch einen Computer umgesetztes Verfahren **700**. Bei **702** kann das Verfahren **700** durch ein funktionsmäßig mit einem Prozessor verbundenes System erfolgendes Erkennen von Informationen umfassen, die von einer in Container einzubindenden Zielanwendung verwendet werden (z.B. über die Sammelkomponente. Der Idd-Befehl kann verwendet werden, um unmittelbare oder Ausgangsabhängigkeiten festzustellen. Die „chroot"- und „strace"-Befehle können ausgeführt werden, um aktuell laufende Codes zu erfassen und zu analysieren, um Fehler zu finden. Bei **704** kann das Verfahren **700** durch das System erfolgendes Ermitteln umfassen, ob für die Informationen innerhalb eines oder mehrerer definierter Grenzwerte ein oder mehrere Risikoverstöße vorliegen (z.B. über die Risikobeurteilungskomponente **108**). Die Risikoanalyse kann durch Verwenden einer Risiko- oder Auswirkungsfunktion erfolgen. Falls der Risikowert größer als der Grenzwert ist, kann die Dienstinstanz ersetzt werden. Bei **706** kann das Verfahren **700** durch das System erfolgendes Ermitteln umfassen, ob in den Informationen ein Konformitäts- oder ein Sicherheitsverstoß vorliegt, wobei das Ermitteln, ob der Konformitäts- oder Sicherheitsverstoß vorliegt, auf Grundlage einer Feststellung eines Nichtvorliegens eines oder mehrerer Risikoverstöße durch die Risikobeurteilungskomponente erfolgt (z.B. über die Konformitätskomponente **110**). Ein Mehrfachmarkierungs-SVM-basierter Lernalgorithmus kann angewendet werden, um Konformität und Sicherheit betreffende Probleme zu erfassen. Bei **708** kann das Verfahren **700** durch das System erfolgendes Erzeugen eines neuen Containers definierten Komponenten der Zielanwendung entsprechender Komponenten umfassen, die es der Zielanwendung ermöglichen, ohne ein zugrunde liegendes Betriebssystem ausgeführt zu werden, wobei das Erzeugen auf Grundlage einer Feststellung erfolgt, dass in den Informationen kein Konformitäts- oder Sicherheitsverstoß vorliegt (z.B. über die Erstellungskomponente **112**). Der per konformitätsbewusster Laufzeit erzeugte Container kann es ermöglichen, dass die Zielanwendung ohne ein zugrunde liegendes Betriebssystem ausgeführt wird. In einigen Ausführungsformen der Erfindung weist der Container keine Shell auf, was ihn außerdem zu einem unveränderbaren Container macht.

[0036] Fig. 8 veranschaulicht ein durch einen Computer umgesetztes Verfahren **800**. Das Verfahren **800** ist für eine beispielhafte Anwendung bereitgestellt, die als ausführbare Datei klassifiziert ist, die einen Satz Systembibliotheken direkt laden kann. Das Beispiel verwendet hier nginx. Bei **802** kann das Verfahren Erstellen eines Verzeichnisses einer separaten Partition und weiterer Verzeichnisse umfassen, die verwendet werden, um eine Anwendung laufen zu lassen (z.B. über die Sammelkomponente **106**). Bei **804** kann das Verfahren **800** Erstellen von Verzeichnissen umfassen, die benötigt werden, um den „chroot"-Befehl auszuführen (z.B. über die Sammelkomponente **106**). Bei **806** kann das Verfahren **800** Herunterladen des aktuellsten Abbilds aus dem Docker-Archiv und lokales Laufenlassen des Abbilds umfassen (z.B. über die Sammelkomponen-

te **106**). Bei **808** kann das Verfahren **800** Auffinden eines zum Laufenlassen eines Containers verwendeten Befehls in der Docker-Datei umfassen (z.B. über die Sammelkomponente **106**). Bei **810** kann das Verfahren **800** Ermitteln von Ausgangsabhängigkeiten einer ausführbaren Datei mittels des ldd-Befehls umfassen (z.B. über die Sammelkomponente **106**). Bei **812** kann das Verfahren **800** Ermitteln umfassen, ob während eines Startens der Anwendung Fehler aufgrund fehlender Dateien auftreten (z.B. über die Sammelkomponente **106**). Falls ja, kann das Verfahren **800** die folgenden Schritte **814**, **816**, **818** und **820** mit Wiederholung des Schritts **812** umfassen. Falls nein, kann das Verfahren **800** die folgenden Schritte **822**, **824** und **826** umfassen, um den Prozess auszuführen und zu beenden. Falls ein Fehler aufgrund einer fehlenden Datei auftritt, kann bei 814 das Verfahren **800** für jede gefundene Abhängigkeit (oder in einigen Ausführungsformen für eine oder mehrere gefundene Abhängigkeiten) Erstellen von deren Verzeichnis in der separaten Partition und Kopieren der Bibliothek aus dem Container in der separaten Partition umfassen (z.B. über die Sammelkomponente **106**). Bei **816** kann das Verfahren **800** Durchführen der gleichen Arbeitsschritte wie vorstehend für Dateien umfassen, bei denen es sich nicht um Bibliotheken handelt, die aber dennoch genutzt werden, damit die Anwendung funktioniert, wie beispielsweise Konfigurationsdateien, Zertifikatdateien, Secure-Socket-Shell- (SSH-) Schlüssel usw. (z.B. über die Sammelkomponente **106**). Bei **818** kann das Verfahren **800** Starten der Anwendung in dem Verzeichnis der separaten Partition mittels der „chroot“- und „strace“-Befehle und Parsen der „strace“-Ausgabe umfassen, um fehlende Dateien oder Bibliotheken zu finden (z.B. über die Sammelkomponente **106**). Bei **820** kann das Verfahren **800** Lokalisieren der fehlenden Dateien und Bibliotheken in dem Docker-Container umfassen (z.B. über die Sammelkomponente **106**). Falls die Dateien in dem Container gefunden werden, kann das Verfahren **800** Hinzufügen der Dateien zu einer Liste von Abhängigkeiten zum Kopieren in das Verzeichnis der separaten Partition umfassen. Anschließend kann das Verfahren **800** Wiederholen der Schritte **812**, **814**, **816**, **818** und **820** umfassen, bis beim Starten einer Anwendung keine Fehler aufgrund fehlender Dateien mehr auftreten. Falls beim Starten einer Anwendung keine Fehler aufgrund fehlender Dateien auftreten, kann bei **822** das Verfahren **800** Erzeugen einer neuen Docker-Datei von Grund auf und Kopieren der Inhalte des Verzeichnisses der separaten Partition im Abbild-Stammverzeichnis umfassen (z.B. über die Sammelkomponente **106**). Bei **824** kann das Verfahren **800** Parsen der ursprünglichen Docker-Datei nach anderen Befehlen wie beispielsweise Offenlegen von Ports, Definieren von Volumina usw. und Hinzufügen der Befehle zu der neuen Docker-Datei umfassen (z.B. über die Sammelkomponente **106**). Bei **826** kann das Verfahren **800** vor einem stabilen Laufenlassen des Containers Ausführen etwaiger weiterer zur Initialisierung und Konfiguration genutzter Befehle umfassen (z.B. Initialisieren, Migrieren und/oder Platzieren einer Datenbank einer Anwendung).

[0037] Fig. 9 und **Fig. 10** veranschaulichen beispielhafte Verfahren **900** und **1000**. Verfahren **900** und **1000** enthalten beispielhafte Anwendungen, die eine Sprachenlaufzeit verwenden. Für diese Arten von Anwendungen wird ein etwas anderer Prozess verwendet, um sicherzustellen bzw. die Wahrscheinlichkeit zu erhöhen, dass die Interpreter in dem Container laufen können und die Anwendungen alle (oder in einigen Ausführungsformen eine oder mehrere der) für den Einsatz genutzten Dateien und/oder Bibliotheken aufweisen. Nachdem die Sprachenlaufzeit zu laufen begonnen hat, kann die Anwendung eingesetzt werden. Python-Anwendungen weisen üblicherweise eine Anforderungsdatei auf, die Python-Wheels und -Pakete aufweist, die installiert werden können. Python weist üblicherweise einen eigenen Installer wie etwa pip auf. Diese Pakete, die eigens zu einer bestimmten Anwendung gehören, können in einem konformitätsbewussten Laufzeit-Container installiert werden. Andere Anwendungen, die eine Sprachenlaufzeit verwenden, können ebenfalls ähnliche Prozesse aufweisen. Bei Ruby können gems anstelle von Paketen verwendet werden, und Ruby kann zudem die Erstellung einer Ruby-Laufzeit verwenden. Eine vollständige Sammlung aus in einer gemfile aufgelisteten gems, die von einer Anwendung verwendet werden, kann durch Ausführen eines „gem list“-Befehls in einer Anwendungslaufzeitumgebung erhalten werden. Bei Java können jars anstelle von gems oder Paketen verwendet werden.

[0038] Fig. 9 veranschaulicht ein Verfahren **900**. Bei dem durch einen Computer umgesetzten Verfahren **900** handelt es sich um eine beispielhafte Python-Laufzeitanwendung. Bei **910** kann das Verfahren **900** Erstellen einer konformitätsbewussten Laufzeit für Python zum Laufenlassen einer Python-Anwendung umfassen (z.B. über die Sammelkomponente **106**). Bei **912** kann das Verfahren **900** Ermitteln einer Liste aller (oder in einigen Ausführungsformen der Erfindung eines oder mehrerer) auf einem System verfügbaren Python-Pakete umfassen (z.B. über die Sammelkomponente **106**). Bei **914** kann das Verfahren **900** Ermitteln einer Liste aller (oder in einigen Ausführungsformen der Erfindung eines oder mehrerer) Python-Pakete umfassen, auf die von einer Zielanwendung zugegriffen wird (z.B. über die Sammelkomponente **106**). Bei **916** kann das Verfahren **900** Ermitteln umfassen, ob eine requirements.txt-Datei spezifiziert ist (z.B. über die Sammelkomponente **106**). Falls ja (eine requirements.txt-Datei ist spezifiziert), kann bei 918 das Verfahren **900** Parsen der Anforderungsdatei umfassen, um eine Liste von Python-Paketabhängigkeiten zu ermitteln (z.B. über die Sammelkomponente **106**). Bei **920** kann das Verfahren **900** Kopie-

ren von Anwendungsskripten in ein Verzeichnis einer separaten Partition umfassen (z.B. über die Sammelkomponente **106**). Ferner kann bei 922 das Verfahren **900** Kopieren definierter (oder in einigen Ausführungsformen der Erfindung verwendeter oder erforderlicher) Python-Pakete in das Verzeichnis der separaten Partition umfassen (z.B. über die Sammelkomponente **106**). Falls keine requirements.txt-Datei spezifiziert ist, kann bei 924 das durch einen Computer umgesetzte Verfahren **900** Beziehen einer Liste von .py-Dateien für diese Anwendung umfassen (z.B. über die Sammelkomponente **106**). Anschließend kann bei 926 das Verfahren **900** Parsen der Importangaben aus den .py-Dateien umfassen (z.B. über die Sammelkomponente **106**). Bei **928** kann das Verfahren **900** Verwenden von Pyflakes und Pylint-Tools umfassen, um ungenutzte Bibliotheken zu entfernen (z.B. über die Sammelkomponente **106**). Bei **930** kann das Verfahren **900** Sammeln aller von einer Anwendung verwendeten eindeutigen Python-Bibliotheken umfassen (z.B. über die Sammelkomponente **106**). Beispielsweise kann eine vollständige Sammlung durch eine Anwendung verwendeter Python-Pakete durch Ausführen eines „pip freeze“-Befehls in der Anwendungslaufzeitumgebung erhalten werden.

[0039] Fig. 10 veranschaulicht ein Verfahren **1000**. **Fig. 10** veranschaulicht einen beispielhaften Prozess **1000** für eine Java-Laufzeitanwendung. Bei **1002** kann das Verfahren **1000** Erstellen einer Mindestlaufzeit für eine Java Virtual Machine (JVM) wie beispielsweise eine Java Runtime Environment (JRE) umfassen, um eine Java-Anwendung laufen zu lassen (z.B. über die Sammelkomponente **106**). Bei **1004** kann das Verfahren **1000** Bestimmen einer Liste aller auf einem System verfügbaren Java-Bibliotheken umfassen (z.B. über die Sammelkomponente **106**). Bei **1006** kann das Verfahren **1000** Lesen der Projekterstellungsdatei umfassen, um alle Bibliotheken zu finden, die an eine Anwendung gebunden sind (z.B. über die Sammelkomponente **106**). Bei **1008** kann das Verfahren **1000** Kopieren von .jar-Dateien in ein Verzeichnis einer separaten Partition umfassen (z.B. über die Sammelkomponente **106**).

[0040] Um einen Kontext für die verschiedenen Aspekte des offenbarten Gegenstands bereitzustellen, sollen **Fig. 11** sowie die nachfolgende Erörterung eine allgemeine Beschreibung einer geeigneten Umgebung bereitstellen, in der die verschiedenen Aspekte des offenbarten Gegenstands umgesetzt werden können. **Fig. 11** veranschaulicht ein Blockschaubild einer beispielhaften, nicht-einschränkenden Betriebsumgebung, in der eine oder mehrere Ausführungsformen der Erfindung ermöglicht werden können.

[0041] Gemäß **Fig. 11** kann eine geeignete Betriebsumgebung **1100** zum Umsetzen verschiedener Aspekte der Erfindung auch einen Computer **1112** beinhalten. Der Computer **1112** kann auch eine Verarbeitungseinheit **1114**, einen Systemspeicher **1116** und einen Systembus **1118** enthalten. Der Systembus **1118** verbindet Systemkomponenten, darunter, ohne jedoch hierauf beschränkt zu sein, den Systemspeicher **1116**, mit der Verarbeitungseinheit **1114**. Bei der Verarbeitungseinheit **1114** kann es sich um einen beliebigen aus verschiedenen erhältlichen Prozessoren handeln. Duale Mikroprozessoren und andere Multiprozessor-Architekturen können ebenfalls als Verarbeitungseinheit **1114** verwendet werden. Bei dem Systembus **1118** kann es sich um einen beliebigen aus verschiedenen Arten von Busstruktur (en) handeln, die den Speicherbus oder die Speichersteuerung, einen Peripheriebus oder externen Bus und/oder einen lokalen Bus enthält, der eine Vielfalt verfügbarer Busarchitekturen verwendet, darunter, ohne jedoch auf diese beschränkt zu sein, Industrial Standard Architecture (ISA), Micro-Channel Architecture (MSA), Extended ISA (EISA), Intelligent Drive Electronics (IDE), VESA Local Bus (VLB), Peripheral Component Interconnect (PCI), Kartenbus, Universal Serial Bus (USB), Advanced Graphics Port (AGP), Firewire (IEEE **1394**) und Small Computer Systems Interface (SCSI).

[0042] Der Systemspeicher **1116** kann zudem flüchtigen Speicher **1120** und nichtflüchtigen Speicher **1122** enthalten. Das Basic Input/Output System (BIOS), das die grundlegenden Routinen zum Übermitteln von Informationen zwischen Elementen innerhalb des Computers **1112** wie beispielsweise während des Startens enthält, ist im nichtflüchtigen Speicher **1122** gespeichert. Zur Veranschaulichung und nicht einschränkend können zu nichtflüchtigem Speicher **1122** Nur-Lese-Speicher (ROM), programmierbarer ROM (PROM), elektrisch programmierbarer ROM (EPROM), elektrisch löschbarer programmierbarer ROM (EEPROM), Flashspeicher oder nichtflüchtiger Direktzugriffsspeicher (RAM) (z.B. ferroelektrischer RAM (FeRAM) zählen. Der flüchtige Speicher **1120** kann ebenfalls einen Direktzugriffsspeicher (RAM) enthalten, der als externer Cachespeicher fungiert. Zur Veranschaulichung und nicht einschränkend ist RAM in vielen Formen erhältlich, z.B. als statischer RAM (SRAM), dynamischer RAM (DRAM), synchroner DRAM (SDRAM), SDRAM mit doppelter Datenrate (DDR SDRAM), weiterentwickelter SDRAM (ESDRAM), Synchlink-DRAM (SLDRAM), Direct Rambus RAM (DRRAM), Direct Rambus Dynamic RAM (DRDRAM) und Rambus Dynamic RAM.

[0043] Der Computer **1112** kann zudem weitere entfernbare/nicht entfernbare, flüchtige/nichtflüchtige Computerspeichermedien enthalten. **Fig. 11** veranschaulicht beispielsweise einen Plattenspeicher **1124**. Zu dem Plattenspeicher **1124** können auch, ohne jedoch hierauf beschränkt zu sein, Einheiten wie ein Magnetplattenlaufwerk, Diskettenlaufwerk, Band-

laufwerk, Jaz-Laufwerk, Zip-Laufwerk, LS-100-Laufwerk, eine Flashspeicherkarte oder ein Speicherstick zählen. Der Plattenspeicher **1124** kann auch Speichermedien separat oder in Kombination mit anderen Speichermedien enthalten, darunter, ohne jedoch auf diese beschränkt zu sein, ein optisches Plattenlaufwerk wie beispielsweise eine Compact-Disk-ROM-Einheit (CD-ROM), beschreibbares CD-Laufwerk (CD-R-Laufwerk), wiederbeschreibbares CD-Laufwerk (CD-RW-Laufwerk) oder ein Digital-Versatile-Disk-ROM-Laufwerk (DVD-ROM). Um die Verbindung des Plattenspeichers **1124** mit dem Systembus **1118** zu ermöglichen, wird typischerweise eine lösbare oder nicht lösbare Schnittstelle wie beispielsweise die Schnittstelle **1126** verwendet. **Fig. 11** zeigt zudem Software, die in der geeigneten Betriebsumgebung **1100** als Vermittler zwischen Benutzern und den beschriebenen grundlegenden Computerressourcen fungiert. Eine solche Software kann beispielsweise auch ein Betriebssystem **1128** enthalten. Das Betriebssystem **1128**, welches auf dem Plattenspeicher **1124** gespeichert sein kann, steuert Ressourcen des Computers **1112** und teilt diese zu.

[0044] Systemanwendungen **1130** nutzen die Verwaltung von Ressourcen durch das Betriebssystem **1128** durch Programmmodule **1132** und Programmdaten **1134**, die z.B. entweder im Systemspeicher **1116** oder auf dem Plattenspeicher **1124** gespeichert werden. Es sei angemerkt, dass diese Offenbarung mit verschiedenen Betriebssystemen oder Kombinationen aus Betriebssystemen umgesetzt werden kann. Ein Benutzer gibt Befehle oder Informationen durch die Eingabeeinheit(en) **1136** in den Computer **1112** ein. Zu den Eingabeeinheiten **1136** zählen, ohne jedoch auf diese beschränkt zu sein, eine Zeigeeinheit wie beispielsweise eine Maus, ein Trackball, ein Stylus, ein Touchpad, eine Tastatur, ein Mikrofon, ein Joystick, ein Gamepad, eine Satellitenantenne, ein Scanner, eine TV-Tuner-Karte, eine Digitalkamera, eine digitale Videokamera, eine Webkamera und dergleichen. Diese und andere Eingabeeinheiten sind durch den Systembus **1118** über einen oder mehrere Schnittstellenanschlüsse **1138** mit der Verarbeitungseinheit **1114** verbunden. Zu dem einen oder den mehreren Schnittstellenanschlüssen **1138** zählen beispielsweise ein serieller Anschluss, ein paralleler Anschluss, ein Gameport und ein Universal Serial Bus (USB). Ausgabeeinheit(en) **1140** nutzen einige der gleichen Arten von Anschlüssen wie die Eingabeeinheit(en) **1136**. Somit kann beispielsweise ein USB-Anschluss verwendet werden, um dem Computer **1112** Eingaben bereitzustellen, und um Informationen aus dem Computer **1112** an eine Ausgabeeinheit **1140** auszugeben. Der Ausgabeadapter **1142** ist bereitgestellt, um zu veranschaulichen, dass es einige Ausgabeeinheiten **1140** wie Monitore, Lautsprecher und Drucker, unter anderen Ausgabeeinheiten **1140**, gibt, die spezielle Adapter erfordern. Zu den Ausgabeadaptern **1142** gehören, veranschaulichend und nicht einschränkend, Video- und Soundkarten, die ein Mittel zur Verbindung zwischen der Ausgabeeinheit **1140** und dem Systembus **1118** bereitstellen. Es wird angemerkt, dass andere Einheiten und/oder Systeme aus Einheiten sowohl Eingabe- als auch Ausgabefunktionen bereitstellen, wie beispielsweise ein oder mehrere entfernt angeordnete Computer **1144**.

[0045] Der Computer **1112** kann in einer vernetzten Umgebung mittels logischer Verbindungen mit einem oder mehreren entfernt angeordneten Computern wie beispielsweise dem einen oder den mehreren entfernt angeordneten Computern **1144** arbeiten. Der oder die entfernt angeordneten Computer **1144** können ein Computer, ein Server, ein Leitwegrechner, ein Netzwerk-PC, eine Workstation, ein mikroprozessorgestütztes Gerät, eine Partnereinheit oder ein anderer gemeinsamer Netzwerkknoten und dergleichen sein und zudem viele oder alle der in Bezug auf den Computer **1112** beschriebenen Elemente enthalten. Der Kürze halber ist bei dem oder den entfernt angeordneten Computern **1144** lediglich eine Arbeitsspeicher- bzw. Speichereinheit **1146** veranschaulicht. Der bzw. die entfernt angeordneten Computer **1144** sind durch eine Netzwerkschnittstelle **1148** logisch und dann über eine Kommunikationsverbindung **1150** physisch mit dem Computer **1112** verbunden. Die Netzwerkschnittstelle **1148** weist drahtgebundene und/oder drahtlose Kommunikationsnetzwerke wie beispielsweise lokale Netzwerke (LAN), Weitverkehrsnetze (WAN), Zellennetze usw. auf. Zu LAN-Technologien zählen Fiber Distributed Data Interface (FDDI), Copper Distributed Data Interface (CDDI), Ethernet, Token Ring und dergleichen. Zu WAN-Technologien zählen, ohne jedoch hierauf beschränkt zu sein, Punkt-zu-Punkt-Verbindungen, leitungsvermittelte Netzwerke wie Integrated Services Digital Networks (ISDN) und Variationen aus diesem, paketvermittelte Netzwerke und Digital Subscriber Lines (DSL). Kommunikationsverbindung(en) **1150** beziehen sich auf die zum Verbinden der Netzwerkschnittstelle **1148** mit dem Systembus **1118** verwendete Hardware/Software. Zwar ist die Kommunikationsverbindung **1150** der Anschaulichkeit halber innerhalb des Computers **1112** gezeigt, jedoch kann sich diese auch außerhalb des Computers **1112** befinden. Die Hardware/Software zur Verbindung mit der Netzwerkschnittstelle **1148** kann zudem, rein beispielhaft, interne und externe Technologien beinhalten, beispielsweise Modems wie reguläre Telefonmodems, Kabelmodems und DSL-Modems, ISDN-Adapter und Ethernet-Karten.

[0046] Es sei klargestellt, dass das Umsetzen der hierin angeführten Lehren nicht auf eine Cloud-Computing-Umgebung beschränkt ist, auch wenn diese Offenbarung eine ausführliche Beschreibung von Cloud-Computing enthält. Stattdessen können Ausführungsformen der vorliegenden Erfindung gemeinsam mit jeder beliebigen Art von jetzt bekann-

ter oder später erfundener Datenverarbeitungsumgebung umgesetzt werden.

[0047] Cloud-Computing ist ein Dienstbereitstellungsmodell zum Ermöglichen eines problemlosen bedarfsgesteuerten Netzwerkzugriffs auf einen gemeinsam genutzten Pool von konfigurierbaren Datenverarbeitungsressourcen (z.B. Netzwerke, Netzwerkbandbreite, Server, Verarbeitung, Hauptspeicher, Speicher, Anwendungen, virtuelle Maschinen und Dienste), die mit minimalem Verwaltungsaufwand bzw. minimaler Interaktion mit einem Anbieter des Dienstes schnell bereitgestellt und freigegeben werden können. Dieses Cloud-Modell kann mindestens fünf Eigenschaften enthalten, mindestens drei Dienstmodelle und mindestens vier Implementierungsmodelle.

[0048] Bei den Eigenschaften handelt es sich um die Folgenden:

On-Demand Self-Service: Ein Cloud-Nutzer kann einseitig automatisch nach Bedarf für Datenverarbeitungsfunktionen wie Serverzeit und Netzwerkspeicher sorgen, ohne dass eine menschliche Interaktion mit dem Anbieter des Dienstes erforderlich ist.

[0049] Broad Network Access: Es sind Funktionen über ein Netzwerk verfügbar, auf die durch Standardmechanismen zugegriffen wird, welche die Verwendung durch heterogene Thin- oder Thick-Client-Plattformen (z.B. Mobiltelefone, Laptops und PDAs) unterstützen.

[0050] Resource-Pooling: Die Datenverarbeitungsressourcen des Anbieters werden zusammengeschlossen, um mehreren Nutzern unter Verwendung eines Multi-Tenant-Modells zu dienen, wobei verschiedene physische und virtuelle Ressourcen dynamisch nach Bedarf zugewiesen und neu zugewiesen werden. Es gibt eine gefühlte Standortunabhängigkeit, da der Nutzer allgemein keine Kontrolle bzw. Kenntnis über den genauen Standort der bereitgestellten Ressourcen hat, aber in der Lage sein kann, einen Standort auf einer höheren Abstraktionsebene festzulegen (z.B. Land, Staat oder Rechenzentrum).

[0051] Rapid Elasticity: Funktionen können für eine schnelle horizontale Skalierung (scale out) schnell und elastisch bereitgestellt werden, in einigen Fällen auch automatisch, und für ein schnelles Scale-in schnell freigegeben werden. Für den Nutzer erscheinen die für das Bereitstellen verfügbaren Funktionen häufig unbegrenzt und sie können jederzeit in jeder beliebigen Menge gekauft werden.

[0052] Measured Service: Cloud-Systeme steuern und optimieren die Verwendung von Ressourcen automatisch, indem sie eine Messfunktion auf einer gewissen Abstraktionsebene nutzen, die für die Art von Dienst geeignet ist (z.B. Speicher, Verarbeitung, Bandbreite sowie aktive Benutzerkonten). Die Inanspruchnahme von Ressourcen kann überwacht, gesteuert und gemeldet werden, wodurch sowohl für den Anbieter als auch für den Nutzer des verwendeten Dienstes Transparenz geschaffen wird.

[0053] Bei den Dienstmodellen handelt es sich um die Folgenden:

Software as a Service (SaaS): Die dem Nutzer bereitgestellte Funktion besteht darin, die in einer Cloud-Infrastruktur laufenden Anwendungen des Anbieters zu verwenden. Die Anwendungen sind über eine Thin-Client-Schnittstelle wie einen Web-Browser (z.B. auf dem Web beruhende E-Mail) von verschiedenen Client-Einheiten her zugänglich. Der Nutzer verwaltet bzw. steuert die zugrunde liegende Cloud-Infrastruktur nicht, darunter das Netzwerk, Server, Betriebssysteme, Speicher bzw. sogar einzelne Anwendungsfunktionen, mit der möglichen Ausnahme von eingeschränkten benutzerspezifischen Anwendungskonfigurationseinstellungen.

[0054] Platform as a Service (PaaS): Die dem Nutzer bereitgestellte Funktion besteht darin, durch einen Nutzer erstellte bzw. erhaltene Anwendungen, die unter Verwendung von durch den Anbieter unterstützten Programmiersprachen und Tools erstellt wurden, in der Cloud-Infrastruktur einzusetzen. Der Nutzer verwaltet bzw. steuert die zugrunde liegende Cloud-Infrastruktur nicht, darunter Netzwerke, Server, Betriebssysteme bzw. Speicher, hat aber die Kontrolle über die eingesetzten Anwendungen und möglicherweise über Konfigurationen des Application Hosting Environment.

[0055] Infrastructure as a Service (IaaS): Die dem Nutzer bereitgestellte Funktion besteht darin, das Verarbeiten, Speicher, Netzwerke und andere grundlegende Datenverarbeitungsressourcen bereitzustellen, wobei der Nutzer in der Lage ist, beliebige Software einzusetzen und auszuführen, zu der Betriebssysteme und Anwendungen gehören können. Der Nutzer verwaltet bzw. steuert die zugrunde liegende Cloud-Infrastruktur nicht, hat aber die Kontrolle über Betriebssysteme, Speicher, eingesetzte Anwendungen und möglicherweise eine eingeschränkte Kontrolle über ausgewählte Netzwerkkomponenten (z.B. Host-Firewalls).

[0056] Bei den Einsatzmodellen handelt es sich um die Folgenden:

Private Cloud: Die Cloud-Infrastruktur wird einzig und allein für eine Organisation betrieben.

Sie kann durch die Organisation oder einen Dritten verwaltet werden und kann sich in den eigenen Räumen oder in fremden Räumen befinden.

[0057] Community Cloud: Die Cloud-Infrastruktur wird von mehreren Organisationen gemeinsam genutzt und unterstützt eine spezielle Benutzergemeinschaft, die gemeinsame Angelegenheiten hat (z.B. Mission, Sicherheitsanforderungen, Richtlinien sowie Überlegungen bezüglich der Einhaltung von Vorschriften). Sie kann durch die Organisationen oder einen Dritten verwaltet werden und kann in den eigenen Räumen oder fremden Räumen stehen.

[0058] Public Cloud: Die Cloud-Infrastruktur wird der allgemeinen Öffentlichkeit oder einer großen Industriegruppe zur Verfügung gestellt und sie gehört einer Cloud-Dienste verkaufenden Organisation.

[0059] Hybrid Cloud: Die Cloud-Infrastruktur ist eine Zusammensetzung aus zwei oder mehreren Clouds (privat, Benutzergemeinschaft oder öffentlich), die zwar einzelne Einheiten bleiben, aber durch eine standardisierte oder proprietäre Technologie miteinander verbunden sind, die Daten- und Anwendungsportierbarkeit ermöglicht (z.B. Cloud-Zielgruppenverteilung für den Lastenausgleich zwischen Clouds).

[0060] Eine Cloud-Computing-Umgebung ist dienstorientiert mit Fokus auf Statusunabhängigkeit, geringer Kopplung, Modularität und semantischer Interoperabilität. Im Herzen von Cloud-Computing liegt eine Infrastruktur, die ein Netzwerk aus zusammengeschalteten Knoten aufweist.

[0061] Unter Bezugnahme auf **Fig. 12** ist eine veranschaulichende Cloud-Computing-Umgebung **1250** abgebildet. Wie gezeigt ist, weist die Cloud-Computing-Umgebung **1250** einen oder mehrere Cloud-Computing-Knoten **1210** auf, mit denen von Cloud-Nutzern verwendete lokale Datenverarbeitungseinheiten wie der elektronische Assistent (PDA, personal digital assistant) oder das Mobiltelefon **1254A**, der Desktop-Computer **1254B**, der Laptop-Computer **1254C** und/oder das Automobil-Computer-System **1254N** Daten austauschen können. Die Knoten **1210** können miteinander Daten austauschen. Sie können physisch oder virtuell in ein oder mehrere Netzwerke wie private, Benutzergemeinschafts-, öffentliche oder hybride Clouds gruppiert werden (nicht gezeigt), wie vorstehend beschrieben wurde, oder in eine Kombination daraus. Dies ermöglicht es der Cloud-Computing-Umgebung **1250**, Infrastruktur, Plattformen und/oder Software als Dienste anzubieten, für die ein Cloud-Nutzer keine Ressourcen auf einer lokalen Datenverarbeitungseinheit vorhalten muss. Es sei darauf hingewiesen, dass die Arten von in **Fig. 12** gezeigten Datenverarbeitungseinheiten **1254A** bis N lediglich veranschaulichend sein sollen und dass die Datenverarbeitungsknoten **1210** und die Cloud-Computing-Umgebung **1250** über eine beliebige Art Netzwerk und/oder über eine beliebige Art von über ein Netzwerk aufrufbarer Verbindung (z.B. unter Verwendung eines Web-Browsers) mit einer beliebigen Art von computergestützter Einheit Daten austauschen können.

[0062] Unter Bezugnahme auf **Fig. 13** wird ein Satz von funktionalen Abstraktionsschichten gezeigt, die durch die Cloud-Computing-Umgebung **1250** (**Fig. 12**) bereitgestellt werden. Es sollte von vornherein klar sein, dass die in **Fig. 13** gezeigten Komponenten, Schichten und Funktionen lediglich veranschaulichend sein sollen und Ausführungsformen der Erfindung nicht darauf beschränkt sind. Wie abgebildet ist, werden die folgenden Schichten und entsprechenden Funktionen bereitgestellt:

Eine Hardware- und Software-Schicht **1360** enthält Hardware- und Software-Komponenten.Zu Beispielen für Hardware-Komponenten gehören: Mainframe-Computer **1361**; auf der RISC- (Reduced Instruction Set Computer) Architektur beruhende Server **1362**; Server **1363**; Blade-Server **1364**; Speichereinheiten **1365**; und Netzwerke sowie Netzwerkkomponenten **1366**. In einigen Ausführungsformen beinhalten Software-Komponenten eine Netzwerk-Anwendungsserver-Software **1367** und eine Datenbank-Software **1368**.

[0063] Die Virtualisierungsschicht **1370** stellt eine Abstraktionsschicht bereit, aus der die folgenden Beispiele für virtuelle Einheiten bereitgestellt werden können: virtuelle Server **1371**, virtueller Speicher **1372**, virtuelle Netzwerke **1373**, darunter virtuelle private Netzwerke, virtuelle Anwendungen und Betriebssysteme **1374**; und virtuelle Clients **1375**.

[0064] In einem Beispiel kann die Verwaltungsschicht **1380** die nachstehend beschriebenen Funktionen bereitstellen. Eine Ressourcen-Bereitstellung **1381** stellt die dynamische Beschaffung von Datenverarbeitungsressourcen sowie anderen Ressourcen bereit, die zum Durchführen von Aufgaben innerhalb der Cloud-Computing-Umgebung verwendet werden. Ein Messen und eine Preisfindung **1382** stellen die Kostenverfolgung beim Verwenden von Ressourcen innerhalb der Cloud-Computing-Umgebung sowie die Abrechnung oder Rechnungsstellung für den Verbrauch dieser Ressourcen bereit. In einem Beispiel können diese Ressourcen Anwendungs-Software-Lizenzen aufweisen. Die Sicherheit stellt die Identitätsüberprüfung für Cloud-Nutzer und Aufgaben sowie Schutz für Daten und andere Ressourcen bereit. Ein Benutzerportal **1383** stellt Nutzern und Systemadministratoren den Zugang zu der Cloud-Computing-Umgebung bereit. Eine Verwaltung des Dienstumfangs **1384** stellt die Zuordnung und Verwaltung von Cloud-Computing-Ressourcen bereit, so dass die benötigten Dienstziele erreicht werden. Ein Planen und Erfüllen von Vereinbarungen zum Dienstumfang (SLA, Service Level Agreement) 1385 stellt die Anordnung vorab und die Beschaffung von Cloud-

Computing-Ressourcen, für die eine zukünftige Anforderung vorausgesehen wird, gemäß einem SLA bereit.

[0065] Eine Arbeitslastschicht **1390** stellt Beispiele für die Funktionalität bereit, für welche die Cloud-Computing-Umgebung verwendet werden kann. Zu nicht-einschränkenden Beispielen für Arbeitslasten und Funktionen, die von dieser Schicht bereitgestellt werden können, gehören: Abbildung und Navigation **1391**; Software-Entwicklung und Lebenszyklusverwaltung **1392**; Bereitstellung von Ausbildung in virtuellen Klassenzimmern **1393**; Datenanalytikverarbeitung **1394**; Transaktionsverarbeitung **1395**; und Transaktionsmodell-Software **1396**.

[0066] Bei der vorliegenden Erfindung kann es sich um ein System, ein Verfahren, eine Vorrichtung und/oder ein Computerprogrammprodukt jedes möglichen technisch detaillierten Integrationsgrads handeln. Das Computerprogrammprodukt kann (ein) durch einen Computer lesbare(s) Speichermedium (oder -medien) beinhalten, auf dem/denen durch einen Computer lesbare Programmanweisungen gespeichert ist/sind, um einen Prozessor dazu zu veranlassen, Aspekte der vorliegenden Erfindung auszuführen. Bei dem durch einen Computer lesbaren Speichermedium kann es sich um eine physische Einheit handeln, die Anweisungen zur Verwendung durch eine Einheit zur Ausführung von Anweisungen behalten und speichern kann. Bei dem durch einen Computer lesbaren Speichermedium kann es sich zum Beispiel um eine elektronische Speichereinheit, eine magnetische Speichereinheit, eine optische Speichereinheit, eine elektromagnetische Speichereinheit, eine Halbleiterspeichereinheit oder jede geeignete Kombination daraus handeln, ohne auf diese beschränkt zu sein. Zu einer nicht erschöpfenden Liste spezifischerer Beispiele des durch einen Computer lesbaren Speichermediums können auch die Folgenden gehören: eine tragbare Computerdiskette, eine Festplatte, ein Direktzugriffsspeicher (RAM), ein Nur-Lese-Speicher (ROM), ein löschbarer programmierbarer Nur-Lese-Speicher (EPROM bzw. FlashSpeicher), ein statischer Direktzugriffsspeicher (SRAM), ein tragbarer Kompaktspeicherplatte-Nur-Lese-Speicher (CD-ROM), eine DVD (digital versatile disc), ein Speicher-Stick, eine Diskette, eine mechanisch kodierte Einheit wie zum Beispiel Lochkarten oder gehobene Strukturen in einer Rille, auf denen Anweisungen gespeichert sind, und jede geeignete Kombination daraus. Ein durch einen Computer lesbares Speichermedium soll in der Verwendung hierin nicht als flüchtige Signale an sich aufgefasst werden, wie zum Beispiel Funkwellen oder andere sich frei ausbreitende elektromagnetische Wellen, elektromagnetische Wellen, die sich durch einen Wellenleiter oder ein anderes Übertragungsmedium ausbreiten (z.B. durch ein Glasfaserkabel geleitete Lichtimpulse) oder durch einen Draht übertragene elektrische Signale.

[0067] Hierin beschriebene, durch einen Computer lesbare Programmanweisungen können von einem durch einen Computer lesbaren Speichermedium auf jeweilige Datenverarbeitungs-/Verarbeitungseinheiten oder über ein Netzwerk wie zum Beispiel das Internet, ein lokales Netzwerk, ein Weitverkehrsnetz und/oder ein drahtloses Netzwerk auf einen externen Computer oder eine externe Speichereinheit heruntergeladen werden. Das Netzwerk kann Kupferübertragungskabel, Lichtwellenübertragungsleiter, drahtlose Übertragung, Leitwegrechner, Firewalls, Vermittlungseinheiten, Gateway-Computer und/oder Edge-Server aufweisen. Eine Netzwerkadapterkarte oder Netzwerkschnittstelle in jeder Datenverarbeitungs-/Verarbeitungseinheit empfängt durch einen Computer lesbare Programmanweisungen aus dem Netzwerk und leitet die durch einen Computer lesbaren Programmanweisungen zur Speicherung in einem durch einen Computer lesbaren Speichermedium innerhalb der entsprechenden Datenverarbeitungs-/Verarbeitungseinheit weiter. Bei durch einen Computer lesbaren Programmanweisungen zum Ausführen von Arbeitsschritten der vorliegenden Erfindung kann es sich um Assembler-Anweisungen, ISA-Anweisungen (Instruction-Set-Architecture), Maschinenanweisungen, maschinenabhängige Anweisungen, Mikrocode, Firmware-Anweisungen, zustandssetzende Daten, Konfigurationsdaten für integrierte Schaltungen oder entweder Quellcode oder Objektcode handeln, die in einer beliebigen Kombination aus einer oder mehreren Programmiersprachen geschrieben werden, darunter eine objektorientierte Programmiersprache wie Smalltalk, C++ o.ä. sowie herkömmliche prozedurale Programmiersprachen wie die Programmiersprache „C“ oder ähnliche Programmiersprachen. Die durch einen Computer lesbaren Programmanweisungen können vollständig auf dem Computer des Benutzers, teilweise auf dem Computer des Benutzers, als eigenständiges Software-Paket, teilweise auf dem Computer des Benutzers und teilweise auf einem fernen Computer oder vollständig auf dem fernen Computer oder Server ausgeführt werden. In letzterem Fall kann der entfernt angeordnete Computer mit dem Computer des Benutzers durch eine beliebige Art Netzwerk verbunden sein, darunter ein lokales Netzwerk (LAN) oder ein Weitverkehrsnetz (WAN), oder die Verbindung kann mit einem externen Computer hergestellt werden (zum Beispiel über das Internet unter Verwendung eines Internet-Dienstanbieters). In einigen Ausführungsformen können elektronische Schaltungen, darunter zum Beispiel programmierbare Logikschaltungen, im Feld programmierbare Gatter-Anordnungen (FPGA, field programmable gate arrays) oder programmierbare Logikanordnungen (PLA, programmable logic arrays) die durch einen Computer lesbaren Programmanweisungen ausführen, indem sie Zustandsinformationen der durch einen Computer lesbaren Programmanweisungen nutzen, um die

elektronischen Schaltungen zu personalisieren, um Aspekte der vorliegenden Erfindung durchzuführen.

[0068] Aspekte der vorliegenden Erfindung sind hierin unter Bezugnahme auf Ablaufpläne und/oder Blockschaltbilder bzw. Schaubilder von Verfahren, Vorrichtungen (Systemen) und Computerprogrammprodukten gemäß Ausführungsformen der Erfindung beschrieben. Es wird darauf hingewiesen, dass jeder Block der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder sowie Kombinationen von Blöcken in den Ablaufplänen und/oder den Blockschaltbildern bzw. Schaubildern mittels durch einen Computer lesbarer Programmanweisungen ausgeführt werden können. Diese durch einen Computer lesbaren Programmanweisungen können einem Prozessor eines Universalcomputers, eines Spezialcomputers oder einer anderen programmierbaren Datenverarbeitungsvorrichtung bereitgestellt werden, um eine Maschine zu erzeugen, so dass die über den Prozessor des Computers bzw. der anderen programmierbaren Datenverarbeitungsvorrichtung ausgeführten Anweisungen ein Mittel zur Umsetzung der in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder festgelegten Funktionen/Schritte erzeugen. Diese durch einen Computer lesbaren Programmanweisungen können auch auf einem durch einen Computer lesbaren Speichermedium gespeichert sein, das einen Computer, eine programmierbare Datenverarbeitungsvorrichtung und/oder andere Einheiten so steuern kann, dass sie auf eine bestimmte Art funktionieren, so dass das durch einen Computer lesbare Speichermedium, auf dem Anweisungen gespeichert sind, ein Herstellungsprodukt aufweist, darunter Anweisungen, welche Aspekte der/des in dem Block bzw. den Blöcken des Ablaufplans und/oder der Blockschaltbilder bzw. Schaubilder angegebenen Funktion/Schritts umsetzen. Die durch einen Computer lesbaren Programmanweisungen können auch auf einen Computer, eine andere programmierbare Datenverarbeitungsvorrichtung oder eine andere Einheit geladen werden, um das Ausführen einer Reihe von Prozessschritten auf dem Computer bzw. der anderen programmierbaren Vorrichtung oder anderen Einheit zu verursachen, um einen auf einem Computer ausgeführten Prozess zu erzeugen, so dass die auf dem Computer, einer anderen programmierbaren Vorrichtung oder einer anderen Einheit ausgeführten Anweisungen die in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder festgelegten Funktionen/Schritte umsetzen.

[0069] Die Ablaufpläne und die Blockschaltbilder bzw. Schaubilder in den Figuren veranschaulichen die Architektur, die Funktionalität und den Betrieb möglicher Ausführungen von Systemen, Verfahren und Computerprogrammprodukten gemäß verschiedenen Ausführungsformen der vorliegenden Erfindung. In diesem Zusammenhang kann jeder Block in den Ablaufplänen oder Blockschaltbildern bzw. Schaubildern ein Modul, ein Segment oder einen Teil von Anweisungen darstellen, die eine oder mehrere ausführbare Anweisungen zur Ausführung der bestimmten logischen Funktion(en) aufweisen. In einigen alternativen Ausführungen können die in den Blöcken angegebenen Funktionen in einer anderen Reihenfolge als in den Figuren gezeigt stattfinden. Zwei nacheinander gezeigte Blöcke können zum Beispiel in Wirklichkeit im Wesentlichen gleichzeitig ausgeführt werden, oder die Blöcke können manchmal je nach entsprechender Funktionalität in umgekehrter Reihenfolge ausgeführt werden. Es ist ferner anzumerken, dass jeder Block der Blockschaltbilder bzw. Schaubilder und/oder der Ablaufpläne sowie Kombinationen aus Blöcken in den Blockschaltbildern bzw. Schaubildern und/oder den Ablaufplänen durch spezielle auf Hardware beruhende Systeme umgesetzt werden können, welche die festgelegten Funktionen oder Schritte durchführen, oder Kombinationen aus Spezial-Hardware und Computeranweisungen ausführen.

[0070] Zwar wurde die Erfindung vorstehend im allgemeinen Kontext durch einen Computer ausführbarer Anweisungen eines Computerprogrammprodukts beschrieben, das auf einem Computer und/oder Computern läuft, jedoch erkennt der Fachmann, dass diese Offenbarung auch in Kombination mit anderen Programmmodulen umgesetzt werden kann. Allgemein beinhalten Programmmodule Routinen, Programme, Komponenten, Datenstrukturen usw., die bestimmte Aufgaben erfüllen und/oder bestimmte abstrakte Datentypen umsetzen. Darüber hinaus erkennt ein Fachmann, dass die erfindungsgemäßen durch einen Computer implementierten Verfahren auch mit anderen Computersystemkonfigurationen umgesetzt werden können, darunter Einzelprozessor- oder Mehrprozessor-Computersysteme, Mini-Datenverarbeitungseinheiten, Mainframe-Rechner sowie Computer, Hand-Datenverarbeitungseinheiten (z.B. PDA, Telefon), mikroprozessorgestützte oder programmierbare Nutzer- oder Industrieelektronik und dergleichen. Die veranschaulichten Aspekte können auch in verteilten Rechenumgebungen umgesetzt werden, in denen Aufgaben durch entfernt angeordnete Verarbeitungseinheiten durchgeführt werden, die durch ein Kommunikationsnetz verbunden sind. Einige, wenn nicht alle, Aspekte dieser Offenbarung können jedoch auch auf eigenständigen Computern umgesetzt werden. In einer verteilten Rechenumgebung können sich Programmmodule sowohl in lokalen als auch in entfernt angeordneten Arbeitsspeicher- bzw. Speichereinrichtungen befinden.

[0071] Die Bezeichnungen „Komponente“, „System“, Plattform“, „Schnittstelle“ und dergleichen wie in dieser Anmeldung verwendet können sich auf eine

computerbezogene Einheit oder eine eine betriebsfähige Maschine mit einer oder mehreren spezifischen Funktionalitäten betreffende Einheit beziehen oder eine solche beinhalten. Bei den vorliegend offenbarten Einheiten kann es sich entweder um Hardware, eine Kombination aus Hardware und Software, Software oder Software in der Ausführung handeln. Beispielsweise kann es sich bei einer Komponente, ohne jedoch hierauf beschränkt zu sein, um einen auf einem Prozessor laufenden Prozess, einen Prozessor, ein Objekt, eine ausführbare Datei, einen Ausführungs-Thread, ein Programm und/oder einen Computer handeln. Zur Verdeutlichung kann es sich sowohl bei einer auf einem Server laufenden Anwendung als auch bei dem Server um eine Komponente handeln. Eine oder mehrere Komponenten können Teil eines Prozesses und/oder eines Ausführungs-Threads sein, und eine Komponente kann sich auf einem Computer und/oder auf zwei oder mehr Computer verteilt befinden. In einem weiteren Beispiel können jeweilige Komponenten von verschiedenen durch einen Computer lesbaren Medien aus ausgeführt werden, auf denen verschiedene Datenstrukturen gespeichert sind. Die Komponenten können über lokale und/oder entfernte Prozesse kommunizieren, beispielsweise gemäß einem Signal mit einem oder mehreren Datenpaketen (z.B. Daten von einer Komponente, die über das Signal mit einer anderen Komponente in einem lokalen System, einem verteilten System und/oder über ein Netzwerk hinweg, beispielsweise das Internet, mit anderen Systemen, kommuniziert). Als weiteres Beispiel kann es sich bei einer Komponente um eine Vorrichtung mit bestimmter Funktionalität handeln, die durch mechanische Teile bereitgestellt wird, welche durch elektrische oder elektronische Schaltungen betrieben werden, und die durch eine von einem Prozessor ausgeführte Softwareanwendung oder Firmware-Anwendung betrieben wird. In einem solchen Fall kann der Prozessor innerhalb oder außerhalb der Vorrichtung liegen und zumindest einen Teil der Software- oder Firmware-Anwendung ausführen. Als weiteres Beispiel kann es sich bei einer Komponente um eine Vorrichtung handeln, die bestimmte Funktionalität durch elektronische Komponenten ohne mechanische Teile bereitstellt, wobei die elektronischen Komponenten einen Prozessor oder ein anderes Mittel beinhalten können, um Software oder Firmware auszuführen, welche zumindest teilweise die Funktionalität der elektronischen Komponenten bereitstellt. In einem Aspekt kann eine Komponente eine elektronische Komponente über eine virtuelle Maschine emulieren, z.B. in einem Cloud-Computing-System.

[0072] Zudem soll der Ausdruck „oder“ als ein inklusives „oder“ und nicht ein exklusives „oder“ verstanden werden. Das heißt, soweit nicht anders angegeben bzw. ohne Kontext ist „X wendet A oder B an“ zu verstehen als jedwede der natürlichen inklusiven Permutationen. Das heißt, wenn X A anwendet, X B anwendet oder X sowohl A als auch B anwendet, dann ist „X wendet A oder B an“ in jedem der genannten Fälle erfüllt. Zudem sind die Artikel „ein/e“ wie in dieser Spezifikation und den beiliegenden Zeichnungen verwendet allgemein auszulegen als „ein/e oder mehrere“, sofern nicht anders angegeben bzw. ohne Kontext, der nach einer Singularform verlangt. Die Bezeichnungen „Beispiel“ und/oder „beispielhaft“ wie vorliegend verwendet werden in der Bedeutung „als Beispiel, Beispielfall oder Veranschaulichung dienend“ verwendet. Zur Klarstellung wird angemerkt, dass der vorliegend offenbarte Gegenstand nicht durch solche Beispiele eingeschränkt wird. Zudem sind keine der vorliegend als „Beispiel“ und/oder „beispielhaft“ beschriebenen Aspekte oder Ausgestaltungen notwendigerweise so auszulegen, dass diese gegenüber anderen Aspekten oder Ausgestaltungen bevorzugt oder vorteilhaft seien, noch sollen hierdurch gleichwertige beispielhafte Strukturen und Methoden, die dem Fachmann bekannt sind, ausgeschlossen werden.

[0073] Die Bezeichnung „Prozessor“ wie in dieser Spezifikation verwendet kann sich auf im Wesentlichen jedwede Datenverarbeitungs- bzw. Verarbeitungseinheit oder -einrichtung beziehen, darunter, ohne jedoch hierauf beschränkt zu sein, Einzelkernprozessoren, Einzelprozessoren mit Fähigkeit zur Multithread-Ausführung von Software, Mehrkernprozessoren, Mehrkernprozessoren mit Fähigkeit zur Multithread-Ausführung von Software, Mehrkernprozessoren mit Hardware-Multithread-Technologie, parallele Plattformen und parallele Plattformen mit verteiltem gemeinsamem Speicher. Zudem kann sich ein Prozessor auf eine integrierte Schaltung, eine anwendungsspezifische integrierte Schaltung (ASIC), einen digitalen Signalprozessor (DSP), eine im Feld programmierbare Gatteranordnung (FPGA), eine programmierbare Logiksteuerung (PLC), eine komplexe programmierbare Logikeinheit (CPLD), Logik aus diskreten Gates oder Transistoren, diskrete Hardware-Komponenten oder eine beliebige Kombination aus diesen beziehen, die dafür konzipiert ist, die vorliegend beschriebenen Funktionen zu erfüllen. Ferner können Prozessoren Nano-Architekturen nutzen, beispielsweise, ohne jedoch hierauf beschränkt zu sein, molekulare und Quantenpunkt-basierte Transistoren, Vermittlungseinheiten und Gates, um die Raumausnutzung zu optimieren oder die Leistung von Nutzergeräten zu steigern. Ein Prozessor kann auch als Kombination aus Datenverarbeitungs- bzw. Verarbeitungseinheiten umgesetzt sein. In dieser Offenbarung werden Bezeichnungen wie „Speicher“, „Speicherung“, „Datenspeicher“, „Datenspeicherung“, „Datenbank“ und im Wesentlichen jede andere für den Betrieb und die Funktionalität einer Komponente relevante Informationsspeicherkomponente so verwendet, dass sie sich auf „Hauptspeicherkomponenten“, in einem „Hauptspeicher“ enthaltene Einheiten oder einen Hauptspei-

cher aufweisende Komponenten beziehen. Es sei angemerkt, dass es sich bei vorliegend beschriebenen Hauptspeichern und/oder Hauptspeicherkomponenten um entweder flüchtigen Hauptspeicher oder nichtflüchtigen Hauptspeicher handeln kann oder diese sowohl flüchtigen als auch nichtflüchtigen Speicher enthalten können. Zur Veranschaulichung und nicht einschränkend können zu nichtflüchtigem Speicher Nur-Lese-Speicher (ROM), programmierbarer ROM (PROM), elektrisch programmierbarer ROM (EPROM), elektrisch löschbarer ROM (EEPROM), Flashspeicher oder nichtflüchtiger Direktzugriffsspeicher (RAM) (z.B. ferroelektrischer RAM (FeRAM) zählen. Flüchtiger Speicher kann RAM beinhalten, der beispielsweise als externer Cachespeicher fungieren kann. Zur Veranschaulichung und nicht einschränkend ist RAM in vielen Formen erhältlich, z.B. als synchroner RAM (SRAM), dynamischer RAM (DRAM), synchroner DRAM (SDRAM), SDRAM mit doppelter Datenrate (DDR SDRAM), weiterentwickelter SDRAM (ESDRAM), Synchlink-DRAM (SLDRAM), Direct Rambus RAM (DRRAM), Direct Rambus Dynamic RAM (DRDRAM) und Rambus Dynamic RAM (RDRAM). Zudem sollen die offenbarten Speicherkomponenten vorliegender Systeme oder durch einen Computer umgesetzter Verfahren, ohne hierauf beschränkt zu sein, diese und andere geeignete Arten von Speicher beinhalten.

[0074] Das vorstehend Beschriebene beinhaltet lediglich Beispiele für Systeme und durch einen Computer umgesetzte Verfahren. Natürlich ist es nicht möglich, jede denkbare Kombination aus Komponenten und durch einen Computer umgesetzten Verfahren für die Zwecke der Beschreibung der Erfindung zu beschreiben, jedoch erkennt ein Fachmann, dass viele weitere Kombinationen und Permutationen dieser Offenbarung möglich sind. Des Weiteren sind die Bezeichnungen „beinhaltet/enthält“, „weist auf“, „besitzt“ und dergleichen, soweit sie in der ausführlichen Beschreibung, den Ansprüchen, den Anhängen und Zeichnungen verwendet werden, als in ähnlicher Weise inklusiv zu verstehen, in der die Bezeichnung „umfassend/aufweisend“ interpretiert wird, wenn sie in einem Anspruch als Bindewort verwendet wird.

[0075] Die Beschreibungen der verschiedenen Ausführungsformen der Erfindung wurden für Zwecke der Veranschaulichung dargelegt, sind jedoch nicht als abschließend oder auf die Erfindung beschränkt zu verstehen. Für den Fachmann sind viele Abwandlungen und Variationen ersichtlich, ohne den Umfang der Erfindung zu verlassen. Die hierin verwendete Terminologie wurde gewählt, um bestmöglich die Grundgedanken der Ausführungsformen der Erfindung, der praktischen Anwendung oder technischen Verbesserung gegenüber den auf dem Markt befindlichen Technologien zu erklären oder um dem Fachmann das Verständnis der Erfindung zu ermöglichen.

Patentansprüche

1. System, das Folgendes aufweist:
einen Speicher, der durch einen Computer ausführbare Komponenten speichert,
einen Prozessor, der funktionsmäßig mit dem Speicher verbunden ist und der in dem Speicher gespeicherte, durch einen Computer ausführbare Komponenten ausführt, wobei die durch einen Computer ausführbaren Komponenten Folgendes aufweisen:
eine Sammelkomponente, die Informationen erkennt, die von einer in Container einzubindenden Zielanwendung verwendet werden,
eine Risikobeurteilungskomponente, die ermittelt, ob für die Informationen innerhalb eines oder mehrerer definierter Grenzwerte ein oder mehrere Risikoverstöße vorliegen,
eine Konformitätskomponente, die ermittelt, ob in den Informationen ein Konformitäts- oder ein Sicherheitsverstoß vorliegt, wobei die Ermittlung durch die Konformitätskomponente auf Grundlage einer Feststellung eines Nichtvorliegens eines oder mehrerer Risikoverstöße durch die Risikobeurteilungskomponente erfolgt, und
eine Erstellungskomponente, die auf Grundlage einer Feststellung, dass in den Informationen kein Konformitäts- oder Sicherheitsverstoß vorliegt, einen neuen Container erzeugt, der definierten Komponenten der Zielanwendung entspricht, die es der Zielanwendung ermöglichen, ohne ein zugrunde liegendes Betriebssystem ausgeführt zu werden.

2. System nach Anspruch 1, welches ferner eine Zuordnungskomponente aufweist, die automatisch ein oder mehrere Merkmale einem oder mehreren Systembetriebsabläufen oder einer oder mehreren Bibliotheken zuordnet.

3. System nach Anspruch 2, wobei ein Zuordnen eines oder mehrerer Merkmale zu einem oder mehreren Systembetriebsabläufen oder einer oder mehreren Bibliotheken ein Zuordnen eines oder mehrerer konformitätsbezogener Merkmale umfasst.

4. System nach Anspruch 2, wobei ein Zuordnen eines oder mehrerer Merkmale zu einem oder mehreren Systembetriebsabläufen oder einer oder mehreren Bibliotheken ein Zuordnen eines oder mehrerer sicherheitsbezogener Merkmale umfasst.

5. System nach Anspruch 1, welches ferner eine Musterkomponente aufweist, die ein oder mehrere zum automatischen Erstellen eines oder mehrerer Container-Abbilder verwendete Muster erzeugt.

6. System nach Anspruch 5, wobei die Erzeugung des einen oder der mehreren Muster auf Grundlage einer Feststellung durch die Risikobeurteilungskomponente, dass für die Informationen ein oder mehrere Risikoverstöße vorliegen, oder auf Grundlage ei-

ner Feststellung durch die Konformitätskomponente erfolgt, dass in den Informationen ein Konformitäts- oder Sicherheitsverstoß vorliegt.

7. System nach Anspruch 1, wobei das System für dynamische Erzeugung einer oder mehrerer einen Zielcode enthaltender Laufzeiten sorgt, welche die Ausführung eines Dienstes ermöglichen.

8. System nach Anspruch 1, welches ferner eine Betriebsablaufkomponente aufweist, die einen Prozess des Betriebssystems anhält.

9. Durch einen Computer umgesetztes Verfahren, umfassend:
durch ein funktionsmäßig mit einem Prozessor verbundenes System erfolgendes Erkennen von Informationen, die von einer in Container einzubindenden Zielanwendung verwendet werden,
durch das System erfolgendes Ermitteln, ob für die Informationen innerhalb eines oder mehrerer definierter Grenzwerte ein oder mehrere Risikoverstöße vorliegen,
durch das System erfolgendes Ermitteln, ob in den Informationen ein Konformitäts- oder ein Sicherheitsverstoß vorliegt, wobei das Ermitteln, ob der Konformitäts- oder Sicherheitsverstoß vorliegt, auf Grundlage einer Feststellung eines Nichtvorliegens eines oder mehrerer Risikoverstöße durch die Risikobeurteilungskomponente erfolgt, und
durch das System erfolgendes Erzeugen eines neuen Containers definierten Komponenten der Zielanwendung entsprechender Komponenten, die es der Zielanwendung ermöglichen, ohne ein zugrunde liegendes Betriebssystem ausgeführt zu werden, wobei das Erzeugen auf Grundlage einer Feststellung erfolgt, dass in den Informationen kein Konformitäts- oder Sicherheitsverstoß vorliegt.

10. Durch einen Computer umgesetztes Verfahren nach Anspruch 9, ferner umfassend durch das System erfolgendes automatisches Zuordnen eines oder mehrere Merkmale zu einem oder mehreren Systembetriebsabläufen oder einer oder mehreren Bibliotheken.

11. Durch einen Computer umgesetztes Verfahren nach Anspruch 10, wobei das automatische Zuordnen eines oder mehrerer Merkmale zu einem oder mehreren Systembetriebsabläufen oder einer oder mehreren Bibliotheken ein automatisches Zuordnen eines oder mehrerer konformitätsbezogener Merkmale umfasst.

12. Durch einen Computer umgesetztes Verfahren nach Anspruch 10, wobei das automatische Zuordnen eines oder mehrerer Merkmale zu einem oder mehreren Systembetriebsabläufen oder einer oder mehreren Bibliotheken ein automatisches Zuordnen eines oder mehrerer sicherheitsbezogener Merkmale umfasst.

13. Durch einen Computer umgesetztes Verfahren nach Anspruch 12, ferner umfassend Erzeugen eines oder mehrerer Muster, wobei das Erzeugen des einen oder der mehreren Muster auf Grundlage eines Feststellens durch die Risikobeurteilungskomponente, dass für die Informationen ein oder mehrere Risikoverstöße vorliegen, oder auf Grundlage eines Feststellens durch die Konformitätskomponente erfolgt, dass in den Informationen ein Konformitäts- oder Sicherheitsverstoß vorliegt.

14. Computerprogrammprodukt zum Ermöglichen konformitätsbewusster Laufzeiterzeugung von Containern, wobei das Computerprogrammprodukt ein durch einen Computer lesbares Speichermedium aufweist, das darauf gespeicherte Programmanweisungen enthält, wobei die Programmanweisungen durch einen Prozessor ausführbar sind, um den Prozessor zu Folgendem zu veranlassen:
Informationen zu erkennen, die von einer in Container einzubindenden Zielanwendung verwendet werden,
zu ermitteln, ob für die Informationen innerhalb eines oder mehrerer definierter Grenzwerte ein oder mehrere Risikoverstöße vorliegen,
zu ermitteln, ob in den Informationen ein Konformitäts- oder ein Sicherheitsverstoß vorliegt, wobei die Ermittlung hinsichtlich des Konformitäts- oder Sicherheitsverstoßes auf Grundlage einer Feststellung eines Nichtvorliegens eines oder mehrerer Risikoverstöße durch die Risikobeurteilungskomponente erfolgt, und einen neuen Container definierten Komponenten der Zielanwendung entsprechender Komponenten zu erzeugen, die es der Zielanwendung ermöglichen, ohne ein zugrunde liegendes Betriebssystem ausgeführt zu werden.

15. Computerprogrammprodukt nach Anspruch 14, wobei eine Erzeugung des neuen Containers auf Grundlage einer Feststellung erfolgt, dass in den Informationen kein Konformitäts- oder Sicherheitsverstoß vorliegt.

16. Computerprogrammprodukt nach Anspruch 14, wobei die Programmanweisungen ferner ausführbar sind, um den Prozessor zu Folgendem zu veranlassen:
automatisch ein oder mehrere Merkmale einem oder mehreren Systembetriebsabläufen oder einer oder mehreren Bibliotheken zuzuordnen.

17. Computerprogrammprodukt nach Anspruch 16, wobei das automatische Zuordnen eines oder mehrerer Merkmale zu einem oder mehreren Systembetriebsabläufen oder einer oder mehreren Bibliotheken ein automatisches Zuordnen eines oder mehrerer konformitätsbezogener Merkmale umfasst.

18. Computerprogrammprodukt nach Anspruch 17, wobei das automatische Zuordnen eines oder mehrerer Merkmale zu einem oder mehreren Systembetriebsabläufen oder einer oder mehreren Bibliotheken ein automatisches Zuordnen eines oder mehrerer sicherheitsbezogener Merkmale umfasst.

19. Computerprogrammprodukt nach Anspruch 14, wobei die Programmanweisungen ferner ausführbar sind, um den Prozessor zu Folgendem zu veranlassen:
ein oder mehrere Muster zu erzeugen, die verwendet werden, um automatisch ein oder mehrere Container-Abbilder zu erstellen.

20. Computerprogrammprodukt nach Anspruch 19, wobei die Erzeugung des einen oder der mehreren Muster auf Grundlage einer ersten Feststellung, dass für die Informationen ein oder mehrere Risikoverstöße vorliegen, oder auf Grundlage einer zweiten Feststellung erfolgt, dass in den Informationen ein Konformitäts- oder Sicherheitsverstoß vorliegt.

Es folgen 13 Seiten Zeichnungen

Anhängende Zeichnungen

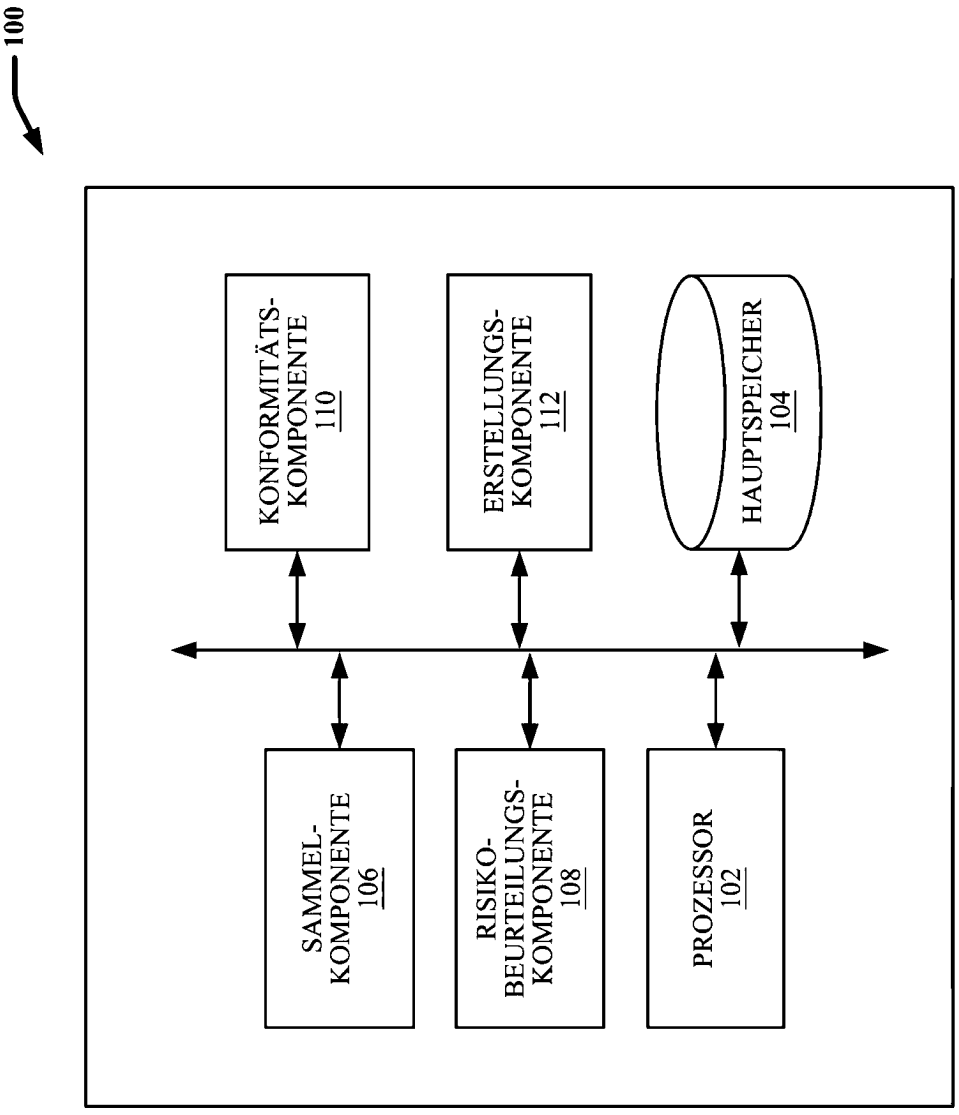


FIG. 1

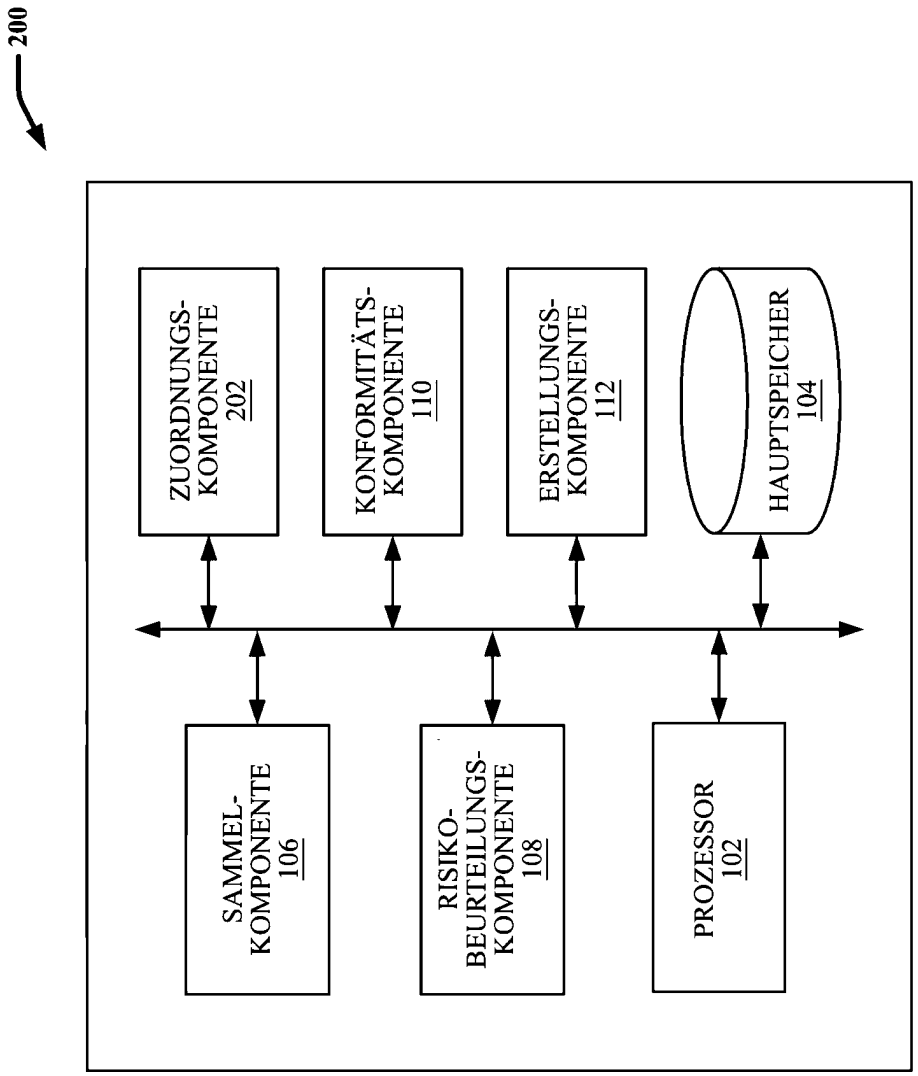

200
ZUORDNUNGS-KOMPONENTE 202
KONFORMITÄTS-KOMPONENTE 110
ERSTELLUNGS-KOMPONENTE 112
HAUPTSPEICHER 104
SAMMEL-KOMPONENTE 106
RISIKO-BEURTEILUNGS-KOMPONENTE 108
PROZESSOR 102

FIG. 2

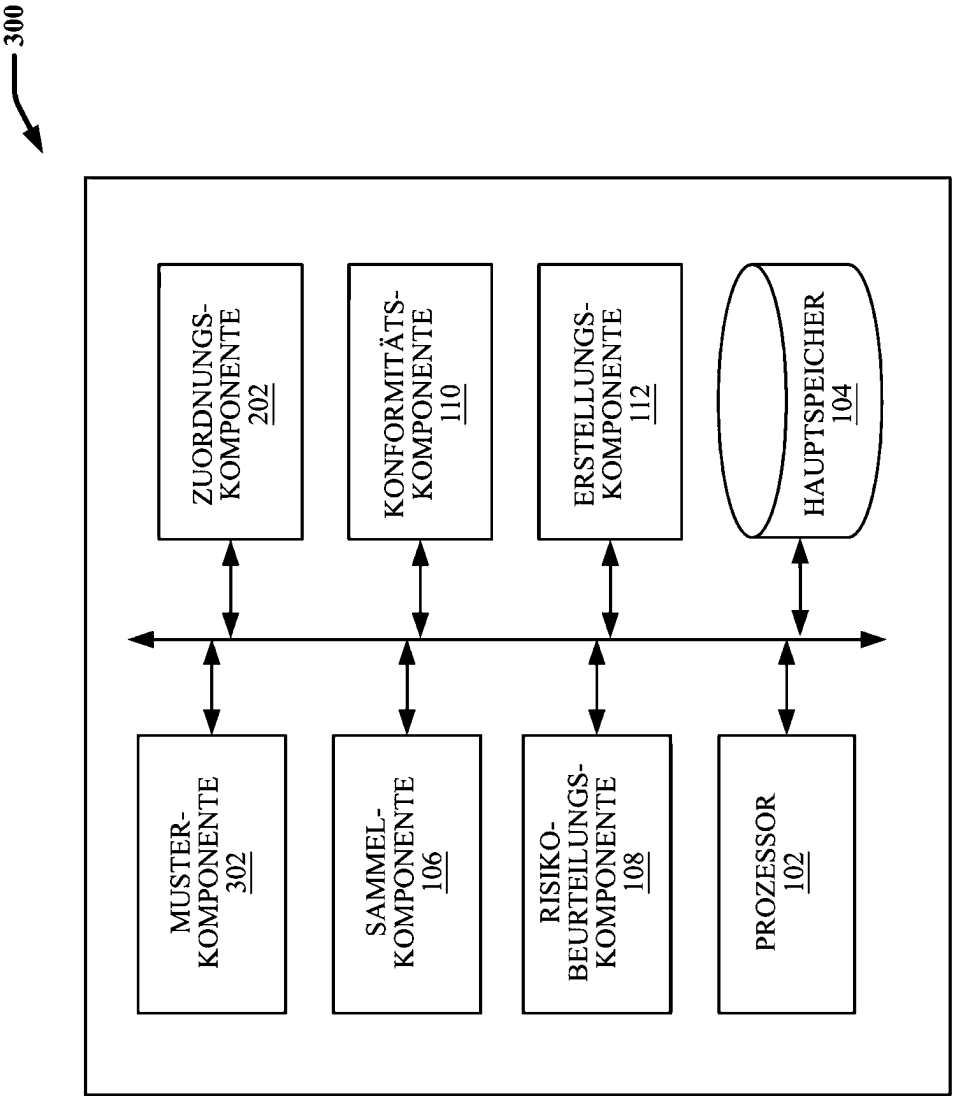

300
ZUORDNUNGS-KOMPONENTE 202
KONFORMITÄTS-KOMPONENTE 110
ERSTELLUNGS-KOMPONENTE 112
HAUPTSPEICHER 104
MUSTER-KOMPONENTE 302
SAMMEL-KOMPONENTE 106
RISIKO-BEURTEILUNGS-KOMPONENTE 108
PROZESSOR 102

FIG. 3

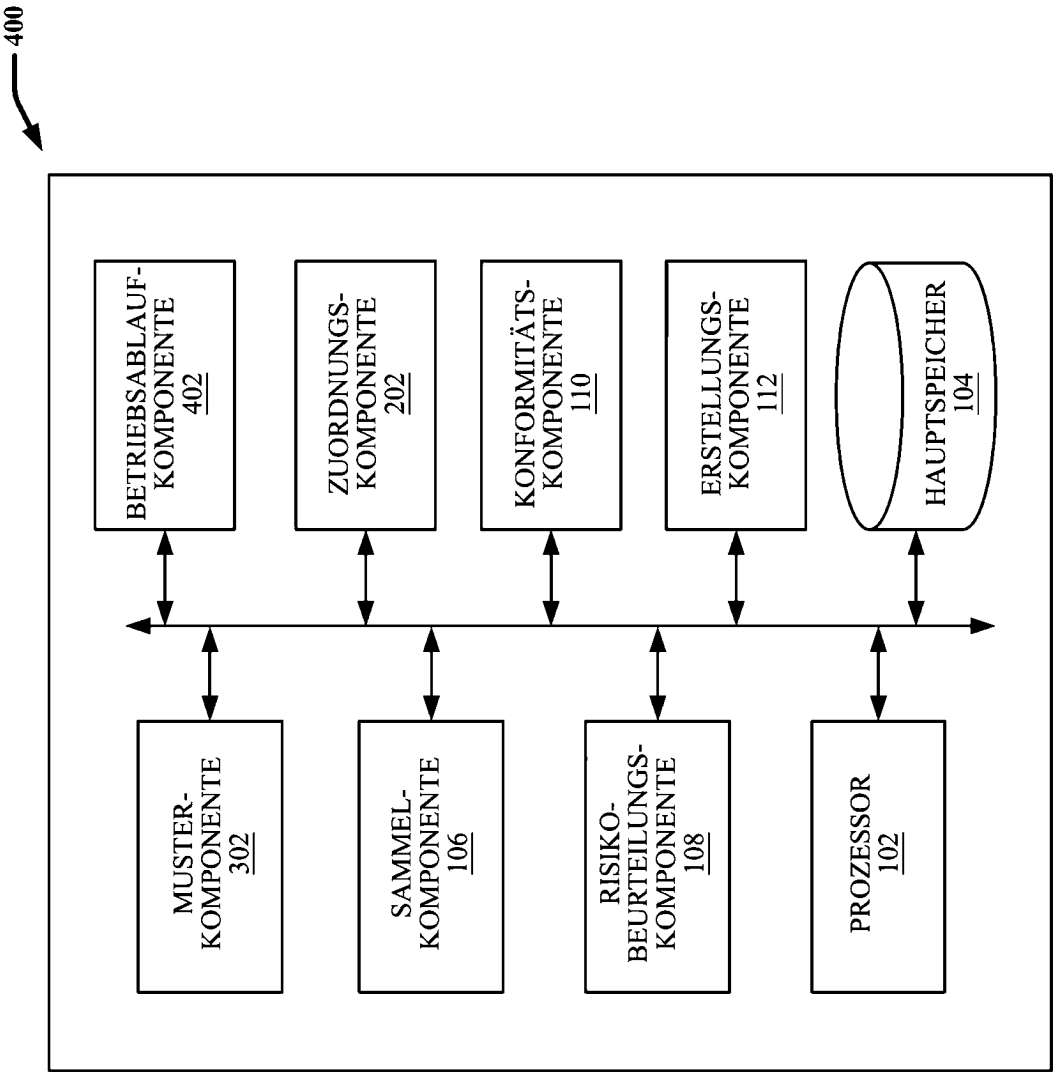

400
BETRIEBSABLAUF-KOMPONENTE 402
ZUORDNUNGS-KOMPONENTE 202
KONFORMITÄTS-KOMPONENTE 110
ERSTELLUNGS-KOMPONENTE 112
HAUPTSPEICHER 104
MUSTER-KOMPONENTE 302
SAMMEL-KOMPONENTE 106
RISIKO-BEURTEILUNGS-KOMPONENTE 108
PROZESSOR 102

FIG. 4

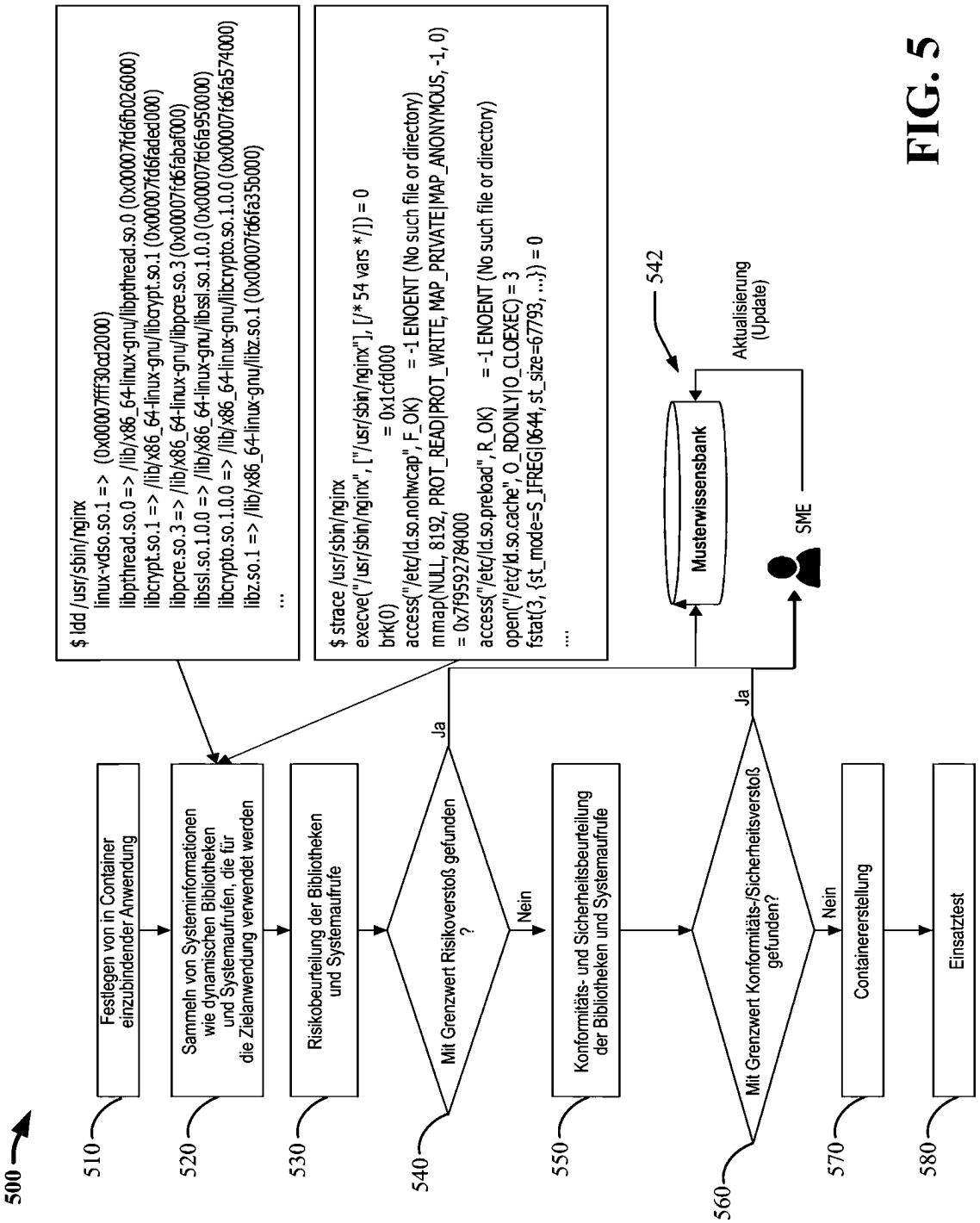

500
510
Festlegen von in Container einzubindender Anwendung
520
Sammeln von Systeminformationen wie dynamischen Bibliotheken und Systemaufrufen, die für die Zielanwendung verwendet werden
530
Risikobeurteilung der Bibliotheken und Systemaufrufe
540
Mit Grenzwert Risikoverstoß gefunden ?
Ja
Nein
550
Konformitäts- und Sicherheitsbeurteilung der Bibliotheken und Systemaufrufe
560
Mit Grenzwert Konformitäts-/Sicherheitsverstoß gefunden?
Ja
Nein
570
Containererstellung
580
Einsatztest
$ ldd /usr/sbin/nginx
linux-vdso.so.1 => (0x00007fff30cd2000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007fd6fb026000)
libcrypt.so.1 => /lib/x86_64-linux-gnu/libcrypt.so.1 (0x00007fd6faded000)
libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007fd6fabaf000)
libssl.so.1.0.0 => /lib/x86_64-linux-gnu/libssl.so.1.0.0 (0x00007fd6fa950000)
libcrypto.so.1.0.0 => /lib/x86_64-linux-gnu/libcrypto.so.1.0.0 (0x00007fd6fa574000)
libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007fd6fa35b000)
...
$ strace /usr/sbin/nginx
execve("/usr/sbin/nginx", ["/usr/sbin/nginx"], [/* 54 vars */]) = 0
brk(0) = 0x1cfd000
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x7f9592784000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=67793, ...}) = 0
...
542
Musterwissensbank
Aktualisierung (Update)
SME

FIG. 5

Eingaben: Markierter Satz D_l, unmarkierter Satz D_u, Anzahl an Schritten T, Anzahl an Beispielen pro Wiederholung S
t = 1
solange t <= T, führe aus
Trainiere einen Mehrfachmarkierungs-SVM-Klassifikator f auf Grundlage von Trainingsdaten D_l
für jede Instanz x in D_u führe aus
Triff Vorhersage über deren Markierungsvektor $\underline{y}$ mittels des LR- (Loss Reduction) basierten Vorhersageverfahrens
$D^*_s = \text{argmax}_{DS} (\sum_{X \in Ds} \sum_{i=1} ((1 - \underline{y}^i f_i(x)) / 2))$
beschränkt auf $\underline{y}^i \in \{-1, 1\}$
(Gleichung für maximale Verlustminderung mit maximaler Konfidenz)
Berechne erwartete Verlustminderung mit höchste Konfidenz aufweisendem Markierungsvektor $\underline{y}$,
$\text{Wert}(x) = \sum^k_{i=1} ((1 - \underline{y}^i f_i(x)) / 2)$
Sortiere Wert (x) in absteigender Ordnung für alle x in D_u
Wähle einen Satz aus S Beispielen D^*_s mit den höchsten Werten (oder Eingabe durch erfahrenen SME),
und aktualisiere den Trainingssatz D_l <- $D_l + D^*_s$
beende für
Trainiere den Mehrfachmarkierungslerner I mit D_l
t = t + 1;
beende solange

FIG. 6

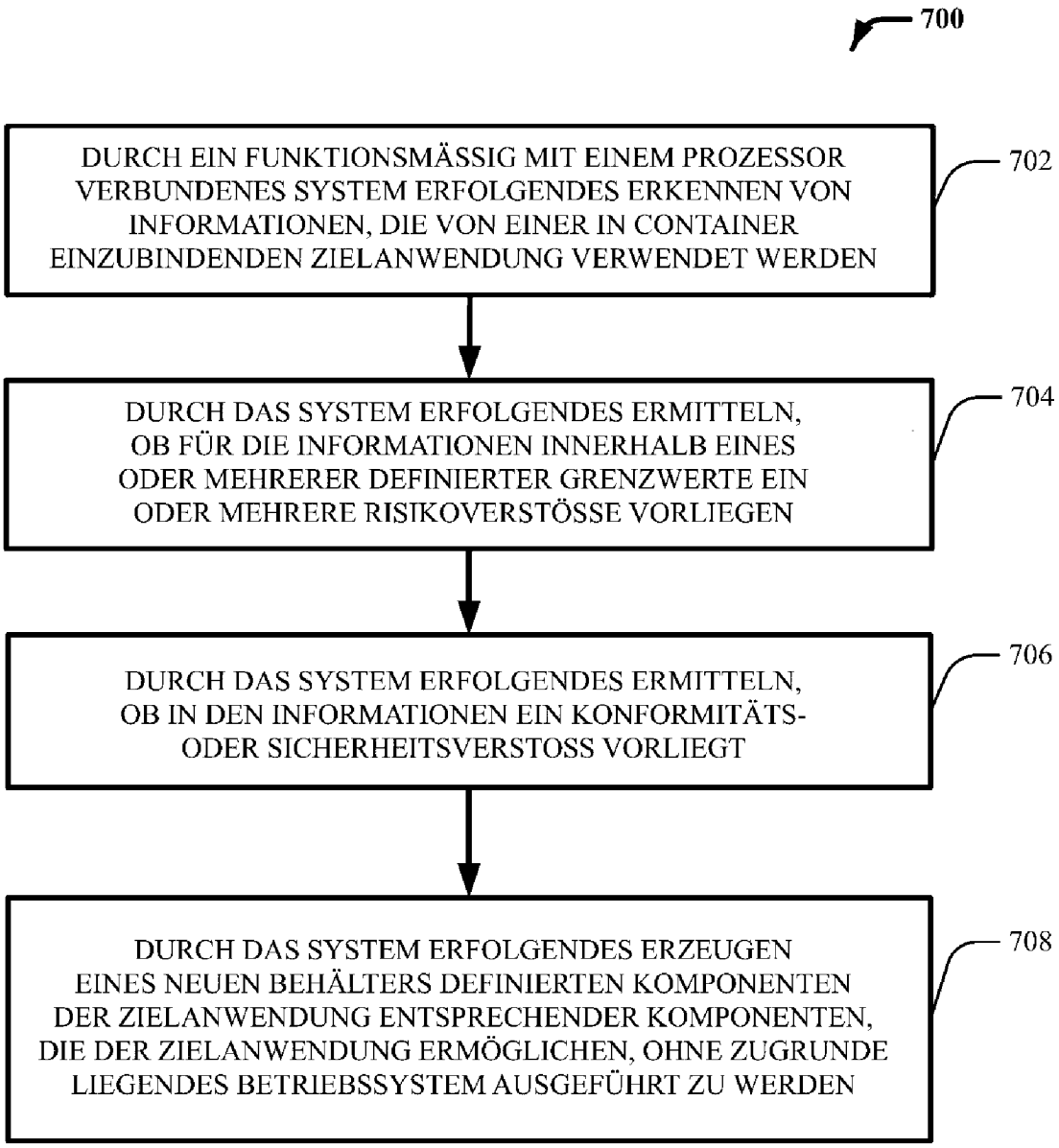

700
DURCH EIN FUNKTIONSMÄSSIG MIT EINEM PROZESSOR VERBUNDENES SYSTEM ERFOLGENDES ERKENNEN VON INFORMATIONEN, DIE VON EINER IN CONTAINER EINZUBINDENDEN ZIELANWENDUNG VERWENDET WERDEN
702
DURCH DAS SYSTEM ERFOLGENDES ERMITTELN, OB FÜR DIE INFORMATIONEN INNERHALB EINES ODER MEHRERER DEFINIERTER GRENZWERTE EIN ODER MEHRERE RISIKOVERSTÖSSE VORLIEGEN
704
DURCH DAS SYSTEM ERFOLGENDES ERMITTELN, OB IN DEN INFORMATIONEN EIN KONFORMITÄTS- ODER SICHERHEITSVERSTOSS VORLIEGT
706
DURCH DAS SYSTEM ERFOLGENDES ERZEUGEN EINES NEUEN BEHÄLTERS DEFINIERTEN KOMPONENTEN DER ZIELANWENDUNG ENTSPRECHENDER KOMPONENTEN, DIE DER ZIELANWENDUNG ERMÖGLICHEN, OHNE ZUGRUNDE LIEGENDES BETRIEBSSYSTEM AUSGEFÜHRT ZU WERDEN
708

FIG. 7

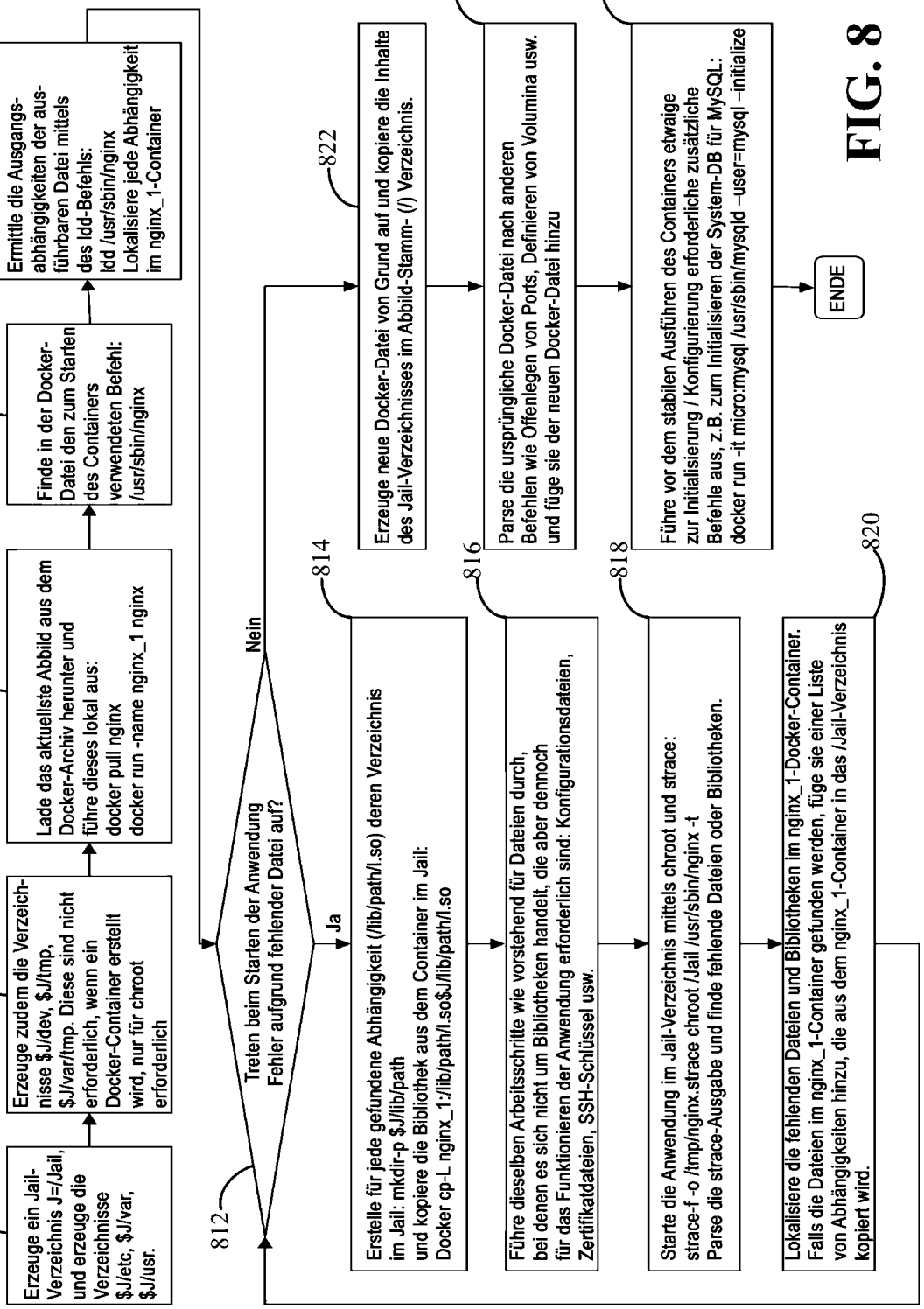

802
Erzeuge ein Jail-Verzeichnis J=/Jail, und erzeuge die Verzeichnisse $J/etc, $J/var, $J/usr.
804
Erzeuge zudem die Verzeichnisse $J/dev, $J/tmp, $J/var/tmp. Diese sind nicht erforderlich, wenn ein Docker-Container erstellt wird, nur für chroot erforderlich
806
Lade das aktuellste Abbild aus dem Docker-Archiv herunter und führe dieses lokal aus:
docker pull nginx
docker run -name nginx_1 nginx
808
Finde in der Docker-Datei den zum Starten des Containers verwendeten Befehl:
/usr/sbin/nginx
810
Ermittle die Ausgangs-abhängigkeiten der ausführbaren Datei mittels des ldd-Befehls:
ldd /usr/sbin/nginx
Lokalisiere jede Abhängigkeit im nginx_1-Container
812
Treten beim Starten der Anwendung Fehler aufgrund fehlender Datei auf?
Nein
Ja
814
Erstelle für jede gefundene Abhängigkeit (/lib/path/l.so) deren Verzeichnis im Jail: mkdir-p $J/lib/path
und kopiere die Bibliothek aus dem Container im Jail:
Docker cp-L nginx_1:/lib/path/l.so$J/lib/path/l.so
816
Führe dieselben Arbeitsschritte wie vorstehend für Dateien durch, bei denen es sich nicht um Bibliotheken handelt, die aber dennoch für das Funktionieren der Anwendung erforderlich sind: Konfigurationsdateien, Zertifikatdateien, SSH-Schlüssel usw.
818
Starte die Anwendung im Jail-Verzeichnis mittels chroot und strace:
strace-f -o /tmp/nginx.strace chroot /Jail /usr/sbin/nginx -t
Parse die strace-Ausgabe und finde fehlende Dateien oder Bibliotheken.
820
Lokalisiere die fehlenden Dateien und Bibliotheken im nginx_1-Docker-Container. Falls die Dateien im nginx_1-Container gefunden werden, füge sie einer Liste von Abhängigkeiten hinzu, die aus dem nginx_1-Container in das /Jail-Verzeichnis kopiert wird.
822
Erzeuge neue Docker-Datei von Grund auf und kopiere die Inhalte des Jail-Verzeichnisses im Abbild-Stamm- (/) Verzeichnis.
824
Parse die ursprüngliche Docker-Datei nach anderen Befehlen wie Öffenlegen von Ports, Definieren von Volumina usw. und füge sie der neuen Docker-Datei hinzu
826
Führe vor dem stabilen Ausführen des Containers etwaige zur Initialisierung / Konfigurierung erforderliche zusätzliche Befehle aus, z.B. zum Initialisieren der System-DB für MySQL:
docker run -it micro:mysql /usr/sbin/mysqld –user=mysql –initialize
ENDE

FIG. 8

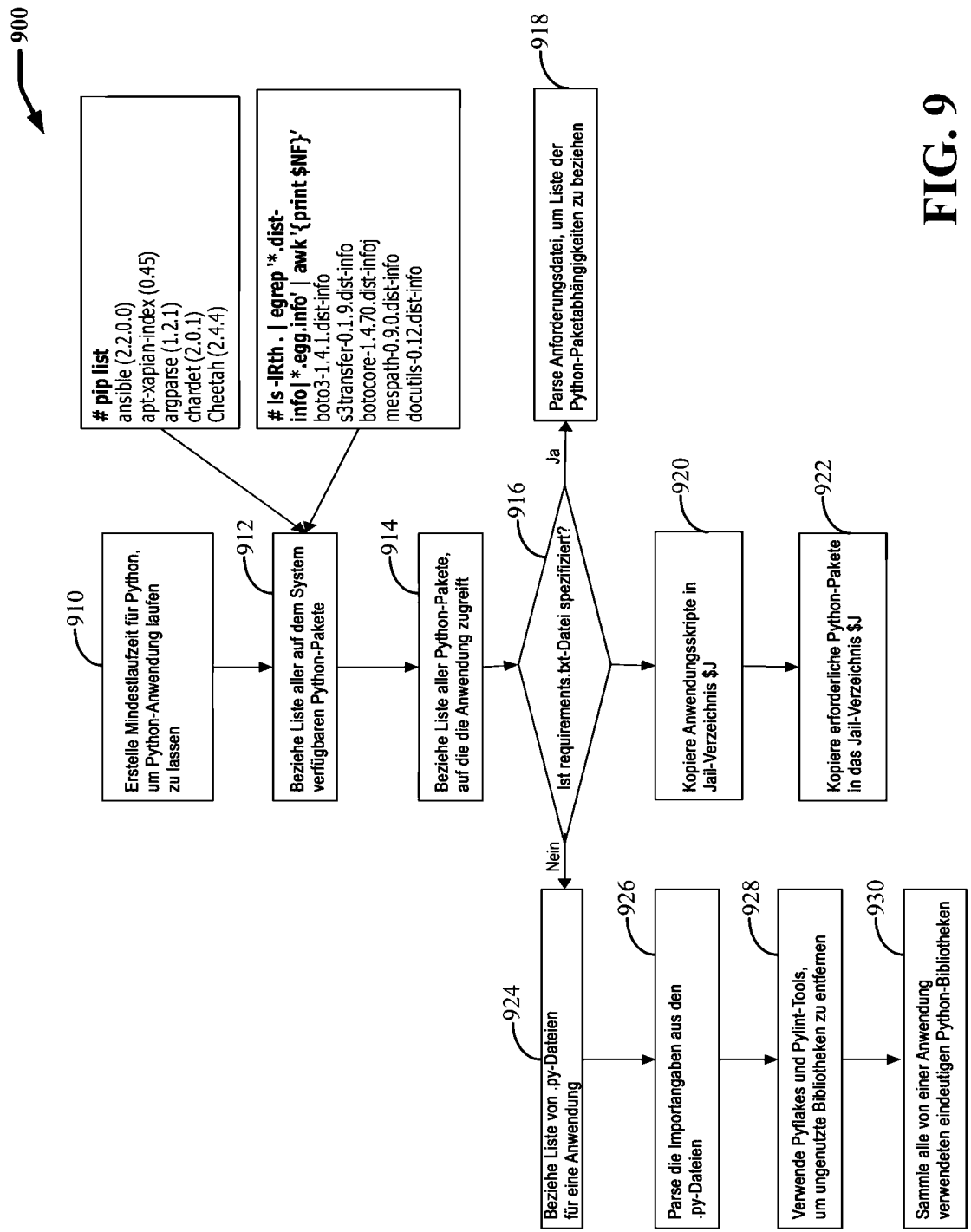

900
910
Erstelle Mindestlaufzeit für Python, um Python-Anwendung laufen zu lassen
912
Beziehe Liste aller auf dem System verfügbaren Python-Pakete
pip list
ansible (2.2.0.0)
apt-xapian-index (0.45)
argparse (1.2.1)
chardet (2.0.1)
Cheetah (2.4.4)
ls -lRth . | egrep '*.dist-info|*.egg.info' | awk '{print $NF}'
boto3-1.4.1.dist-info
s3transfer-0.1.9.dist-info
botocore-1.4.70.dist-infoj
mespath-0.9.0.dist-info
docutils-0.12.dist-info
914
Beziehe Liste aller Python-Pakete, auf die die Anwendung zugreift
916
Ist requirements.txt-Datei spezifiziert?
Ja
Nein
918
Parse Anforderungsdatei, um Liste der Python-Paketabhängigkeiten zu beziehen
920
Kopiere Anwendungsskripte in Jail-Verzeichnis $J
922
Kopiere erforderliche Python-Pakete in das Jail-Verzeichnis $J
924
Beziehe Liste von .py-Dateien für eine Anwendung
926
Parse die Importangaben aus den .py-Dateien
928
Verwende Pyflakes und Pylint-Tools, um ungenutzte Bibliotheken zu entfernen
930
Sammle alle von einer Anwendung verwendeten eindeutigen Python-Bibliotheken

FIG. 9

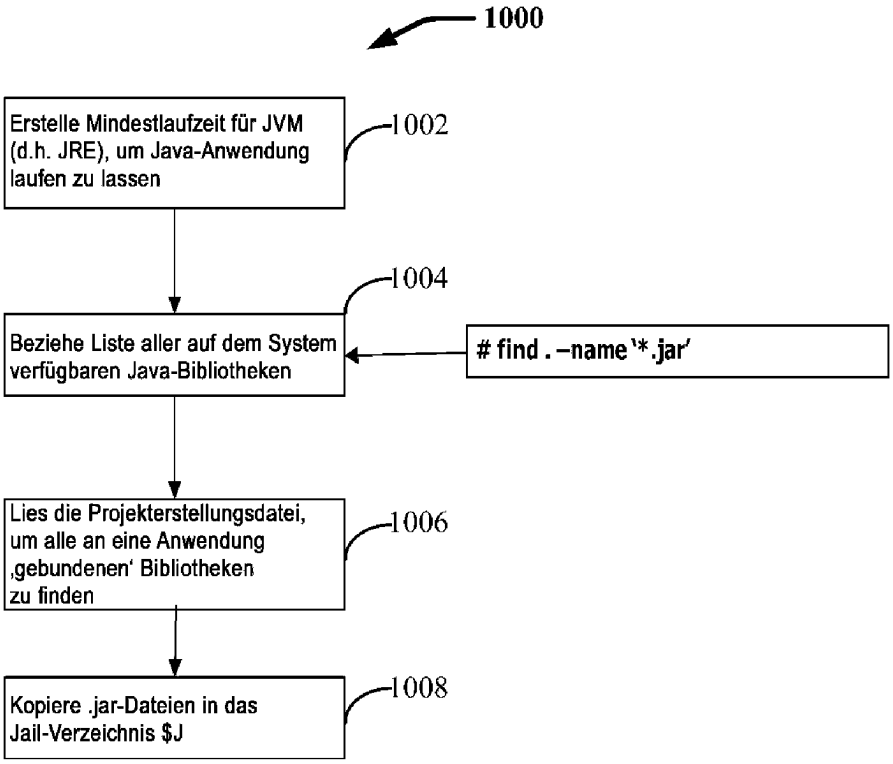

1000
Erstelle Mindestlaufzeit für JVM (d.h. JRE), um Java-Anwendung laufen zu lassen
1002
Beziehe Liste aller auf dem System verfügbaren Java-Bibliotheken
1004
find . –name '*.jar'
Lies die Projekterstellungsdatei, um alle an eine Anwendung ‚gebundenen' Bibliotheken zu finden
1006
Kopiere .jar-Dateien in das Jail-Verzeichnis $J
1008

FIG. 10

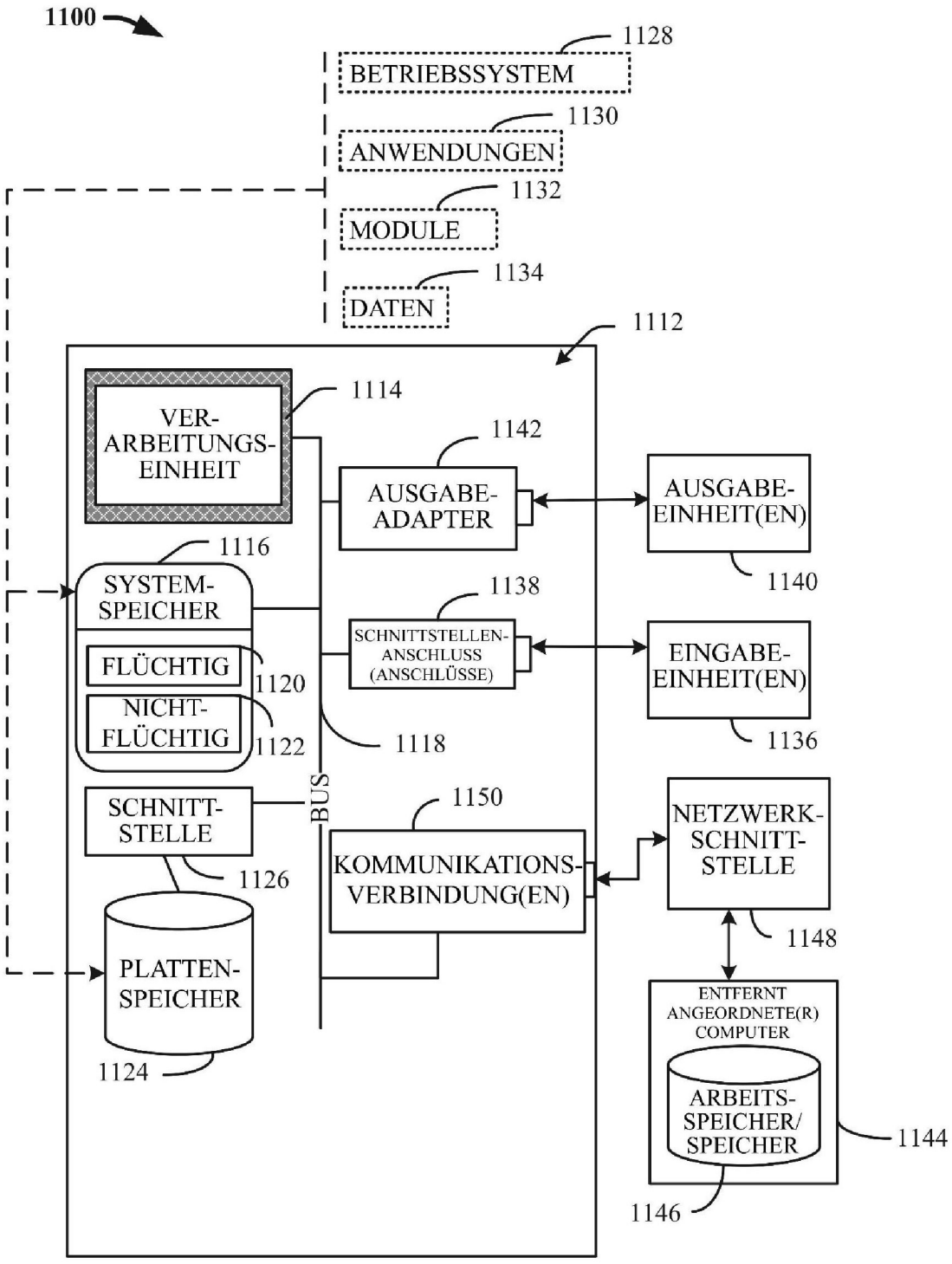

1100
1128
BETRIEBSSYSTEM
1130
ANWENDUNGEN
1132
MODULE
1134
DATEN
1112
1114
VER-
ARBEITUNGS-
EINHEIT
1142
AUSGABE-
ADAPTER
AUSGABE-
EINHEIT(EN)
1140
1116
SYSTEM-
SPEICHER
FLÜCHTIG
1120
NICHT-
FLÜCHTIG
1122
1138
SCHNITTSTELLEN-
ANSCHLUSS
(ANSCHLÜSSE)
EINGABE-
EINHEIT(EN)
1136
1118
BUS
SCHNITT-
STELLE
1126
1150
KOMMUNIKATIONS-
VERBINDUNG(EN)
NETZWERK-
SCHNITT-
STELLE
1148
PLATTEN-
SPEICHER
1124
ENTFERNT
ANGEORDNETE(R)
COMPUTER
ARBEITS-
SPEICHER/
SPEICHER
1144
1146

FIG. 11

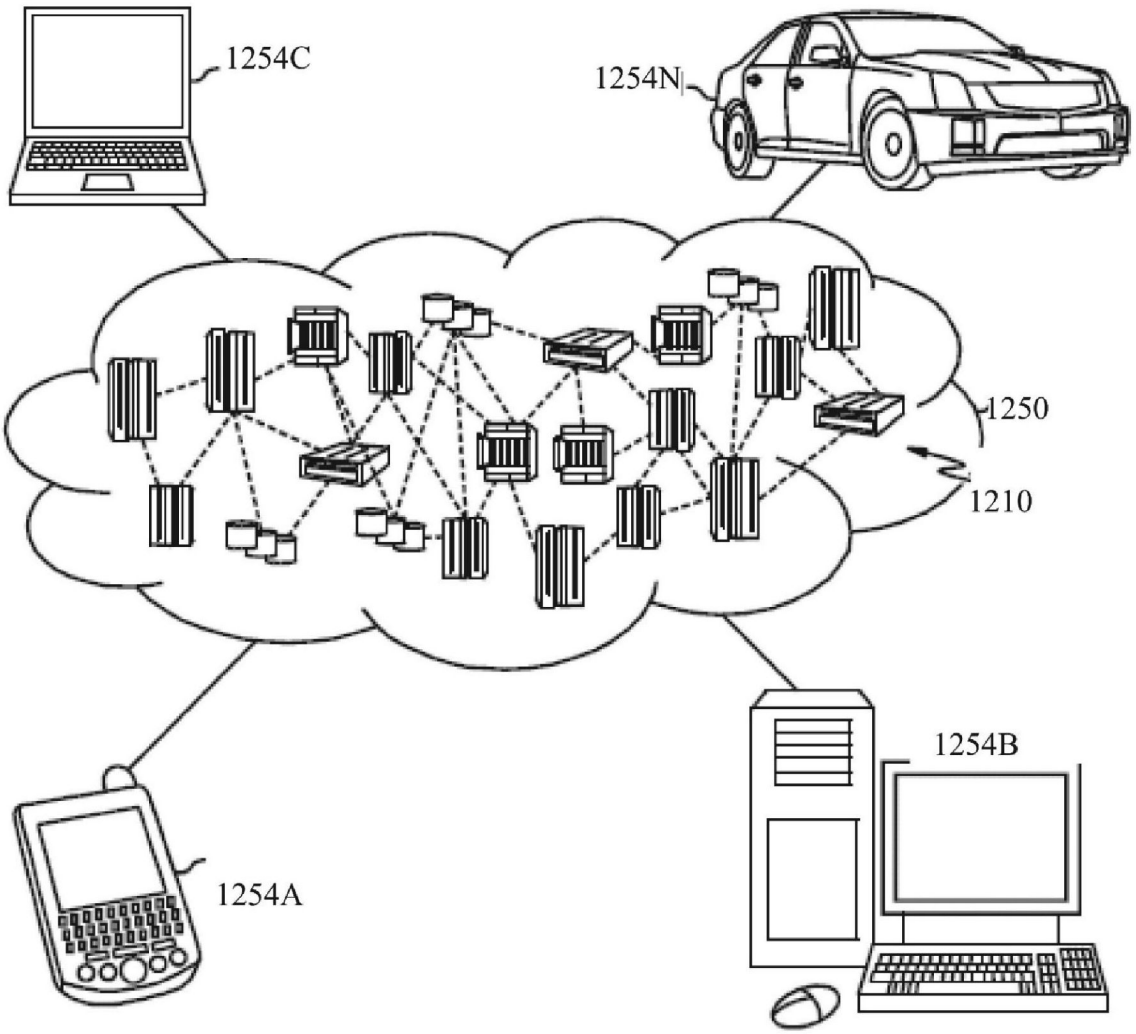


FIG. 12

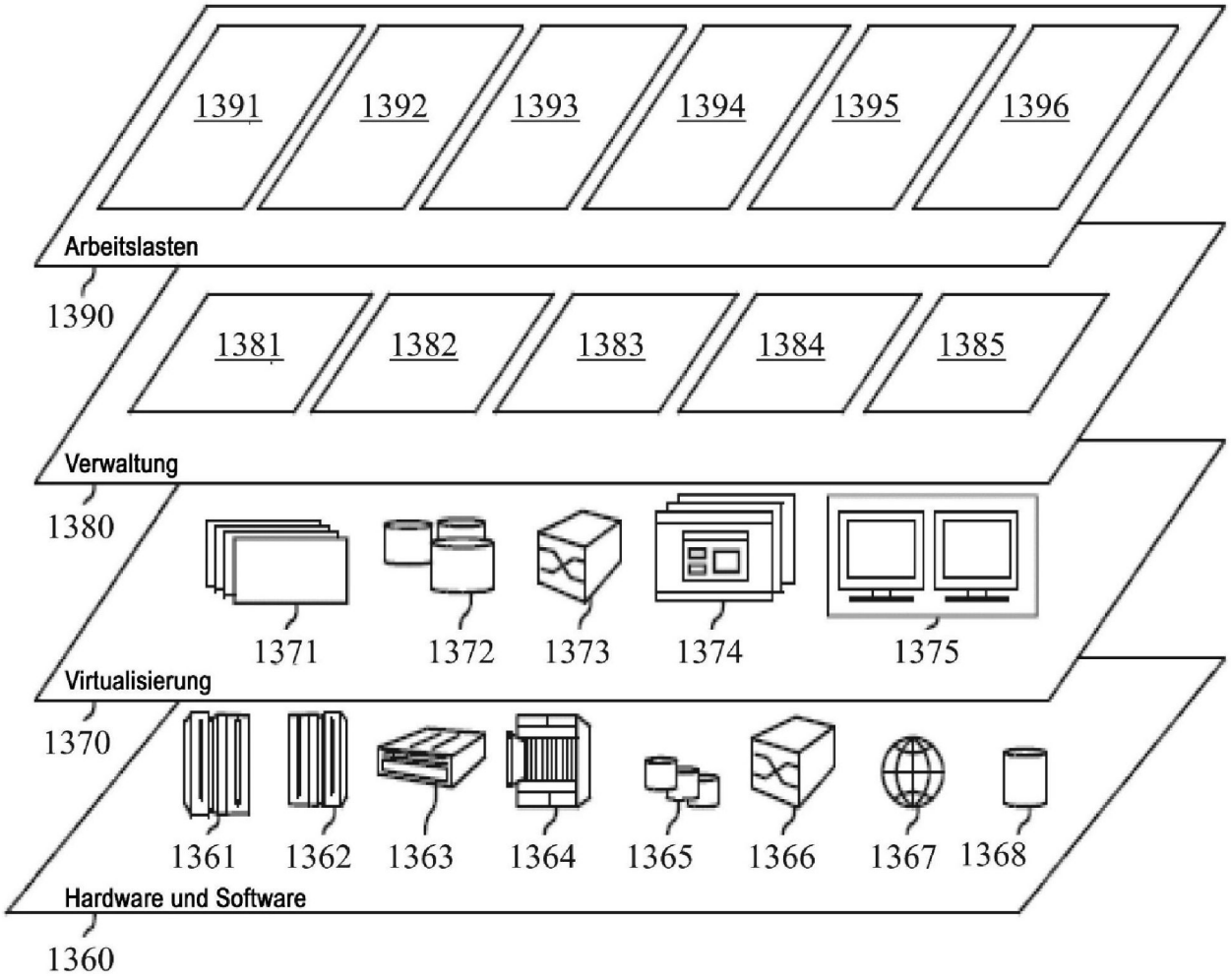

1391
1392
1393
1394
1395
1396
Arbeitslasten
1390
1381
1382
1383
1384
1385
Verwaltung
1380
1371
1372
1373
1374
1375
Virtualisierung
1370
1361
1362
1363
1364
1365
1366
1367
1368
Hardware und Software
1360

FIG. 13