



(10) **DE 20 2016 008 044 U1** 2017.03.09

(12) **Gebrauchsmusterschrift**

(21) Aktenzeichen: **20 2016 008 044.0**

(51) Int Cl.: **G06F 9/44 (2006.01)**

(22) Anmeldetag: **27.06.2016**

(67) aus Patentanmeldung: **PCT/US2016/039561**

(47) Eintragungstag: **27.01.2017**

(45) Bekanntmachungstag im Patentblatt: **09.03.2017**

(30) Unionspriorität:

14/796,695 10.07.2015 US

(74) Name und Wohnsitz des Vertreters:

**Betten & Resch Patent- und Rechtsanwälte
PartGmbH, 80333 München, DE**

(73) Name und Wohnsitz des Inhabers:

GOOGLE INC., Mountain View, Calif., US

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Automatische Importe und Abhängigkeiten in gross angelegten Quellcode-Repositoryn**

(57) Hauptanspruch: System zum Suchen von Kandidaten qualifizierter Namen für mindestens einen einfachen Namen in einer Quellcodedatei, wobei das Verfahren Folgendes umfasst:

ein oder mehrere Verarbeitungsvorrichtungen und ein oder mehrere Speichervorrichtungen, die Anweisungen speichern, die bei Ausführung durch ein oder mehrere Verarbeitungsvorrichtungen, die eine oder mehreren Verarbeitungsgeräte zu Folgendem veranlassen:

empfangen einer Quellcodedatei mit mindestens einem einfachen Namen für den ein entsprechender qualifizierter Name erforderlich ist;

analysieren der Quellcodedatei, um die einfachen Namen in der Datei und die qualifizierten Namen in der Datei zu ermitteln;

für den mindestens einen einfachen Namen, für den ein entsprechender qualifizierter Name erforderlich ist, Durchsuchen einer Quellcodegrafik, um potenzielle qualifizierte Namen für den mindestens einfachen Namen zu finden; und nach Empfang einer Liste potenzieller Kandidaten qualifizierter Namen aus der Quellcodegrafik, Ausgeben der Liste.

```
1. import java.util.List;
2. public class classDemo{
3. List theList = new ArrayList();
4. ....
}
```

Beschreibung**HINTERGRUND**

[0001] Entwickler bauen häufig komplexe Softwareanwendungen auf, die externe Codes oder Programmbibliotheken miteinbeziehen. So kann ein Entwickler beispielsweise eine Liste von Menschen in seine Anwendung einbauen wollen. Anstatt sein eigenes Listenobjekt zu definieren, kann der Entwickler eine Listenklasse aus einem externen Paket einfügen. In diesem Beispiel, wie in **Fig. 1** dargestellt, kann der Entwickler eine Liste aus einer Standardbibliothek in seinen Code einfügen, wie „java.util.List.“ Die „java.util.List“ ist eine Klasse strukturierter Sammlungen, die öffentlich über das java.util-Paket verfügbar ist. Sobald die Liste in eine Quellcodedatei eines Entwicklers importiert wurde, kann der Entwickler sich mit dem einfachen Namen „Liste“ auf diese Liste beziehen, ohne dass er den vollständigen Namen „java.util.List“ überall dort, wie eine Instanz der Liste erstellt oder diese eingesetzt wird, verwenden zu müssen. Ein Beispiel findet sich in Zeile 3 von **Fig. 1**, wo der Code sich einfach auf den einfachen Namen „List“, und nicht auf den vollständigen Namen „java.util.List“ bezieht. Unter Schutz gestellt werden und Gegenstand des Gebrauchsmusters sind, entsprechend den Vorschriften des Gebrauchsmustergesetzes, lediglich Vorrichtungen wie in den beigefügten Schutzansprüchen definiert, jedoch keine Verfahren. Soweit nachfolgend in der Beschreibung gegebenenfalls auf Verfahren Bezug genommen wird, dienen diese Bezugnahmen lediglich der beispielhaften Erläuterung der in den beigefügten Schutzansprüchen unter Schutz gestellten Vorrichtung oder Vorrichtungen.

[0002] Zwar kann der Prozess des Importierens von Paketen, Bibliotheken oder eines externen Codes, der die vollständigen Namen in einen Quellcode festlegt, bei kleinen Codebasen, die standardmäßige integrierte Entwicklungsumgebungen (IDEs, integrated development environments) verwenden, recht unkompliziert sein, groß angelegte Codebasen haben jedoch zu viele externe Abhängigkeiten, sodass IDEs die entsprechenden vollständigen Namen, bezeichnet als qualifizierte Namen, oder die angemessenen Build-Systemkonstrukte für die Abhängigkeiten nicht wirksam finden können.

[0003] Konventionelle Systeme versuchen, qualifizierte Namen in Echtzeit zu finden, indem sie die gesamte Codebasis durchsuchen und versuchen, das entsprechende Paket, die Bibliothek oder einen anderen externen Code zu ermitteln. Dieser Prozess ist sehr zeitaufwändig. Mit diesem Ansatz ist es nicht möglich, große Mengen an Quellcode innerhalb eines angemessenen Zeitraums zu verarbeiten.

[0004] Wie der Erfinder erkannt hat, ist es wünschenswert, schnell Kandidaten qualifizierter Namen zu finden, um das entsprechende Paket, die Bibliothek oder einen externen Code für einfache Namensreferenzen, die im Quellcode vorhanden sind, und die Build-Systemabhängigkeiten, welche die qualifizierten Namen liefern, zu importieren.

KURZDARSTELLUNG

[0005] Diese Spezifikation beschreibt Technologien, die sich auf die Suche nach Kandidaten qualifizierter Namen für mindestens einen einfachen Namen in eine Quellcodedatei beziehen, z. B. in einer Quellcodedatei in der Java-Sprache.

[0006] Im Allgemeinen kann ein Aspekt des in dieser Spezifikation beschriebenen Gegenstands in einem System zum Suchen von Kandidaten qualifizierter Namen für mindestens einen einfachen Namen in einer Quellcodedatei enthalten sein. Ein Beispielsystem könnte eines oder mehrere Verarbeitungsgereäte beinhalten, sowie eines oder mehrere Speichergereäte, die Anweisungen speichern, die ein Beispielverfahren umsetzen. Ein Beispielverfahren kann beinhalten: Empfangen einer Quellcodedatei mit mindestens einem einfachen Namen, für den ein zugeordneter qualifizierter Name erforderlich ist; Parsen der Quellcodedatei, um die einfachen Namen in der Datei und der qualifizierten Namen in der Datei zu ermitteln; für mindestens einen einfachen Namen, für den ein zugehöriger qualifizierter Name erforderlich ist, Suchen einer Quellcodegrafik, um potenzielle qualifizierte Namen zu finden, die mit dem mindestens einen einfachen Namen zugeordnet sind; und, nach Empfang einer Liste potenzieller Kandidaten für qualifizierte Namen aus der Quellcodegrafik, Ausgeben der Liste.

[0007] Diese und andere Ausführungsformen können wahlweise eine oder mehrere der folgenden Eigenschaften beinhalten: die Liste potenzieller Kandidaten qualifizierter Namen kann vor Ausgeben der Liste mit zusätzlichen Informationen gefiltert oder angereichert werden; Build-Abhängigkeiten und Build-Ziele für spezifische qualifizierte Namen können ermittelt werden, nachdem ein qualifizierter Name ausgewählt wurde; die Liste potenzieller Kandidaten von qualifizierten Namen kann vor Ausgeben der Liste basierend auf der Anzahl der Male, die ein qualifizierter Name eines bestimmten Typs von anderen Quellcodedateien benutzt wurde, gefiltert werden; die Liste kann eine Rangfolge haben, die auf der Häufigkeit basiert, mit der ein bestimmter qualifizierter Name in anderen Teilen eines Quellcode-Repositoriums verwendet wurde; falls die Liste nur einen potenziellen qualifizierten Namen enthält, kann der Name ohne Interaktion mit dem Nutzer automatisch als der qualifizierte Name gewählt werden; eine Oberfläche kann bereitgestellt werden, damit der Nutzer eine Aus-

wahl qualifizierter Namen aus der Liste mit potenziellen Kandidaten von qualifizierten Namen empfangen kann; das Filtern kann das Ermitteln beinhalten, ob ein qualifiziertes Namenspaket von der Quellcodedatei sichtbar ist, in der auf einen einfachen, möglicherweise mit dem qualifizierten Namen übereinstimmenden, Namen aktuell Bezug genommen wird und als Reaktion auf die Ermittlung, dass der qualifizierte Name von der Quellcodedatei sichtbar ist, die Bereitstellung des Namens des qualifizierten Namenspakets als potenzieller Kandidat für den qualifizierten Namen erfolgt; und das Filtern kann das Ermitteln des Ausmaßes an Komplikationen beinhalten, die ein qualifiziertes Namenspaket zum Build-Prozess hinzufügen kann, einschließlich des Anstiegs von Zeitaufwand oder Ressourcen, um die Quellcodedatei aufzubauen und als Reaktion auf die Ermittlung, dass das Ausmaß der Komplikation, die ein qualifiziertes Namenspaket dem Build-Prozess hinzufügt, ein nachhaltiges Ausmaß ist, das Hinzufügen des qualifizierten Namenspakets zur Liste der potenziellen Kandidaten qualifizierter Namen erfolgt.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0008] Fig. 1 zeigt ein Beispiel eines Quellcodes, der einen qualifizierten Namen und einen einfachen Namen enthält.

[0009] Fig. 2 ist ein Blockdiagramm, das ein Beispielsystem zur Ermittlung qualifizierter Namen für einfache Namen in einer Quellcodedatei darstellt.

[0010] Fig. 3 ist ein Blockdiagramm zur Veranschaulichung eines Beispiel-Computergeräts.

AUSFÜHRLICHE BESCHREIBUNG

[0011] Gemäß einer beispielhaften Ausführungsform, kann es eine großformatige Grafik in einem verteilten System vorhanden sein, die alle Quellcodes für ein bestimmtes Quellcode-Repository oder -Repositories repräsentiert. Der Quellcode kann indiziert sein, um die Grafik zu erstellen. In einigen Ausführungsformen kann diese Grafik auf einem Rechner oder mehreren Rechnern vorhanden sein, auf die Entwickler fernzugreifen können, sodass die Grafik nicht auf dem Rechner eines einzelnen Entwicklers ausgeführt wird und mehrere Entwicklern gleichzeitig darauf zugreifen können. In ihrer einfachsten Form kann die Grafik aus zwei Tabellen bestehen, wobei die eine einfache Namen qualifizierten Namen zuweist (mappt) und eine zweite Tabelle, die qualifizierte Namen zu Build-Systemabhängigkeiten mappt. Durch Erstellen und Zugriff auf die Grafik in einem verteilten System oder einer Cloudumgebung, können Berechnungen unter Entwicklern zwischengespeichert und erneut benutzt werden. In anderen Ausführungsformen kann die Grafik auf dem eigenen Rechner des Entwicklers vorhanden sein.

[0012] Wie in Fig. 2 dargestellt, gibt es in einer exemplarischen Quellcodedatei zwei einfache Namen (201): „Bar“ in Zeile 3 und „Foo“ in Zeile 5. Der qualifizierte Name für „Bar“ wird im Code als „com.sourcecode.Bar“ in Zeile 1 bereitgestellt. Der qualifizierte Name für „Foo“ wird jedoch nicht bereitgestellt.

[0013] In einem Beispielsystem kann eine auf dem Rechner eines Entwicklers ausgeführte integrierte Entwicklungsumgebung (IDE, an integrated development environment) einen Parser (203) haben, der möglicherweise eine Liste einfacher Namen bereitstellt, die in einer bestimmten Quellcodedatei verwendet werden. Der Parser (203) kann außerdem eine Liste qualifizierter Namen in der Quellcodedatei bereitstellen. Bei der Quellcodedatei, dargestellt in Fig. 2, kann die Liste einfacher Namen wie folgt sein: [Foo, Bar]. Die Liste qualifizierter Namen kann wie folgt sein: [com.sourcecode.Bar]. Wie dargestellt, wird ein dem einfachen Namen zugeordneter qualifizierter Name möglicherweise nicht bereitgestellt. Um den qualifizierten Namen zu finden und das Build-Systemkonstrukt, das den qualifizierten Namen für einfache Namen in dem Quellcode angibt, für den der zugeordnete qualifizierte Name nicht bereitgestellt wurde, kann die Quellcodegrafik (205) nach möglichen Erweiterungen der einfachen Namen zu qualifizierten Namen abgefragt werden. In dem dargestellten Beispiel, kann die Quellcodegrafik (205) nach dem qualifizierten Namen für „Foo“ abgefragt werden und „Foo“ zugehörigen Abhängigkeiten aufbauen. Falls der qualifizierte Namen als ein Import bereitgestellt wird, wie „Bar“ in diesem Beispiel, kann die Build-Systemabhängigkeit vorhanden sein oder nicht vorhanden sein. Aus diesem Grund kann die Quellcodegrafik (205) auch nach Build-Systemabhängigkeiten abgefragt werden, selbst wenn der vollständig qualifizierte Namen im Quellcode bereitgestellt wird.

[0014] Bei der Suche nach „Foo“ kann z. B. beim Durchsuchen der Quellcodegrafik (205) nach passenden Klassen und Build-Systemen gesucht werden, die den Namen „Foo“ enthalten. In einigen Ausführungsformen können die Ergebnisse danach gefiltert werden, wie ein einfacher Name im Code verwendet wird. Sollte beispielsweise ein bestimmtes Verfahren von „Foo“ gerufen werden, können vollqualifizierte potenzielle Kandidaten für die Fertigstellung, die das Verfahren nicht enthalten, aus der Liste der potenziellen Kandidaten herausgefiltert werden. Die Abfrage kann anschließend eine Liste mit Importen und Build-Systemabhängigkeiten zurückgeben. Wie jeder mit normalen Fähigkeiten in diesem Bereich sehen kann, gibt es mehrere Möglichkeiten, diese Liste zu ordnen und zu filtern.

[0015] Die Quellcodegrafik (205) kann durchlaufen werden, um Informationen wie Build-Systemabhängigkeiten für einfache, in den Dateien gefundene, Namen zu erhalten. So kann beispielsweise die Quell-

codegrafik in der Lage sein, die Ziele für die Abhängigkeiten auf den qualifizierten Namen für den einfachen Namen „Foo“ bereitzustellen, sodass der Quellcode in Objektdateien zusammengestellt werden kann.

[0016] Die Liste potenzieller qualifizierter Namen für einen vorgegebenen einfachen Namen, die durch das Durchlaufen der Quellcodegrafik (205) erhalten wurde, kann mit anderen Daten angereichert und weiter gefiltert werden (213). Anreicherungsdaten können beinhalten: wie oft der qualifizierte Name im Repository verwendet wurde (um eine Rangordnung zu ermöglichen); ob der qualifizierte Name als abgelehnt gekennzeichnet wurde, und falls dies der Fall ist, ob das qualifizierte Namenpaket vom Quellcode sichtbar ist, in dem aktuell darauf Bezug genommen wird (einige Klassen können vorsätzlich für bestimmte Softwarepakete gesperrt sein); und das Maß an Komplikationen, die dieser Name zum Build-Prozess beiträgt, einschließlich eines Anstiegs von Zeitaufwand und/oder der Ressourcen, die zum Aufbau des Quellcodes benötigt werden. Die Liste kann z. B. auf Grundlage der Häufigkeit der Verwendung eines qualifizierten Namens von einem bestimmten Typ von anderen Quellcodedateien gefiltert werden. Das Filtern kann auch das Entfernen aller nicht-sichtbaren Klassen oder das Entfernen aller Klassen, die nicht in den spezifischen Kontext im Code passen, beinhalten. So kann beispielsweise „Foo“ ein spezifisches Verfahren in einer Quellcodedatei verlangen, sodass alle passenden „Foo“-Bibliotheken, die dieses Verfahren nicht haben, aus der Liste potenzieller Kandidaten herausgefiltert werden. Bei der Rückgabe einer Liste potenzieller Kandidaten für die Fertigstellung und Abhängigkeiten an einen Nutzer (215) zur Begutachtung, kann ein Beispielsystem die zurückgegebenen Ergebnisse auf Grundlage der Häufigkeit, mit der ein bestimmter qualifizierter Name von anderen Entwicklern in anderen Teilen des Quellcode-Repositories verwendet wurde, in einer Rangordnung aufstellen.

[0017] In einigen Ausführungsformen sind intensiv genutzte qualifizierte Namen die Namen, die am wahrscheinlichsten von Entwicklern in ihren Quellcodedateien verwendet wurden. Neben anderen Rangordnungen können Namen auch danach angeordnet werden, ob sie für die Quellcodedatei sichtbar sind, in der sie eingesetzt werden sollen, ob die Klasse, die der Name repräsentiert, ein Verfahren hat, das in der Quellcodedatei verwendet wird, und ob Klassen abgelehnt sind. Klassen, die abgelehnt wurden, können in der Namenrangfolge zurückgestuft werden. Anschließend kann einem Nutzer (215) die angereicherte, gefilterte Liste potenzieller qualifizierter Namenskandidaten für eine spezifische einfache Namensreferenz im Code angezeigt werden und der Nutzer kann das geeignetste Paket, die Bibliothek oder eine andere externe Codequelle für das einfache Namensobjekt wählen. So-

bald der Nutzer (215) den korrekten qualifizierten Namen für den einfachen Namen im Quellcode aus den Kandidaten der qualifizierten Namen ausgewählt hat, können die ausgewählten qualifizierten Namen erneut an die Quellgrafik (205) gesendet werden, um Informationen über das Build-Systemabhängigkeiten zu erhalten, um Bild-Regelabhängigkeiten zu manipulieren und Ziele für die spezifischen qualifizierten Namen aufzubauen. In einigen Ausführungsformen können die Liste potenzieller Kandidaten für die Fertigstellung und Build-Systemabhängigkeiten gefiltert werden und die Abfrage kann nur den Import und Build-Systemabhängigkeiten zurückgeben, welche die wahrscheinlichsten Übereinstimmungen sind.

[0018] Wie in Fig. 2 dargestellt, können die qualifizierten Namen, sobald die eindeutigen qualifizierten Namen (207, 209) in der Quellcodegrafik (205) gefunden oder von der Datei selbst (203) analysiert wurden und mit den in der Quellcodedatei (201) verwendeten Namen abgeglichen wurden, in die Datei (201) importiert bzw. ihr zugefügt werden. Die Quellgrafik kann periodisch neu erstellt werden, um zu ermitteln, welche Klasse/Build-Abhängigkeiten am häufigsten verwendet werden.

[0019] Ein Beispielfahrer, wie in Fig. 3 dargestellt, beginnt mit dem Empfang einer Quellcodedatei mit mindestens einem einfachen Namen für den ein entsprechender qualifizierter Name gefunden werden soll (oder erforderlich ist) (301). Die Quellcodedatei kann dann analysiert werden, um eine Liste einfacher, in der Datei verwendeter Namen zu finden und eine weitere Liste aller qualifizierter Namen in der Datei (303). Für den mindestens einen einfachen Namen, für den ein entsprechender qualifizierter Name gefunden werden soll, kann die Grafik des gesamten Quellcodes durchsucht werden, um potenzielle Kandidaten qualifizierter Namen für den einfachen Namen zu finden (305). In einigen Ausführungsformen kann die Kandidatenliste der Namen mit zusätzlichen Informationen angereichert oder gefiltert werden. Die Liste kann einem Nutzer zur Verfügung gestellt werden, damit der Nutzer den angemessenen qualifizierten Namen für einen vorgegebenen einfachen Namen auswählen kann (309). In einigen Ausführungen, falls nur ein Kandidat für den qualifizierten Namen vorhanden ist, muss ein Nutzer möglicherweise den qualifizierten Namen nicht auswählen und der qualifizierte Name kann automatisch ausgewählt werden. In einigen Ausführungen, falls nur ein Kandidat für den qualifizierten Namen vorhanden ist, muss ein Nutzer möglicherweise nur bestätigen, dass der qualifizierte Name eine passende Übereinstimmung für den einfachen Namen ist. Nachdem der qualifizierte Name gewählt wurde, kann ein Beispielfahrer Build-Abhängigkeiten und Build-Ziele für spezifische qualifizierte Namen ermitteln. Die mit den einfachen Namen übereinstimmenden qualifizierten Namen in der Quellcodedatei können der Quelldatei hinzuge-

fügt werden. Anschließend kann die Quellcodegrafik erneut erstellt werden und Informationen zur Häufigkeit der einfachen Namen/qualifizierten Namen/Build-Abhängigkeit können bei zukünftigen Analysen verwendet werden, um qualifizierte Namen und Build-Abhängigkeiten zu finden.

[0020] Fig. 4 ist ein hochrangiges Blockdiagramm eines Beispiel-Computers (**400**), der für das Hosten und Publizieren von Softwarepaketen organisiert ist. In der Basiskonfiguration (**401**) besitzt das Computergerät (**400**) normalerweise einen oder mehrere Prozessoren (**410**) und einen Systemspeicher (**420**). Ein Speicherbus (**430**) kann zur Kommunikation zwischen dem Prozessor (**410**) und dem Systemspeicher (**420**) verwendet werden.

[0021] Je nach der gewünschten Konfiguration kann es sich bei dem Prozessor (**410**) um einen beliebigen Typ handeln, einschließlich unter anderem um einen Mikroprozessor (μ P), einen Mikrokontroller (μ C), einen digitalen Signalprozessor (DSP) oder eine Kombination dieser Prozessoren. Der Prozessor (**410**) kann eine oder mehrere Cache-Ebenen besitzen wie einen Cache der ersten Ebene (**411**) und einen Cache der zweiten Ebene (**412**), einen Prozessorkern (**413**) und Register (**414**). Der Prozessorkern (**413**) kann aus einer arithmetisch-logischen Einheit (ALU), einer Gleitkommaeinheit (FPU), einem digitalen Signalprozessorkern (DSP-Kern) oder einer beliebigen Kombination dieser Komponenten bestehen. Ein Speichercontroller (**416**) kann zusätzlich zum Prozessor (**410**) vorhanden sein oder der Speichercontroller (**415**) kann bei manchen Implementierungen ein interner Teil des Prozessors (**410**) sein.

[0022] Je nach der gewünschten Konfiguration kann es sich bei dem Systemspeicher (**420**) um einen beliebigen Typ handeln, einschließlich unter anderem um einen flüchtigen Speicher (wie RAM), einen nicht-flüchtigen Speicher (wie ROM, Flashspeicher usw.) oder eine beliebige Kombination dieser Speichermöglichkeiten. Zum Systemspeicher (**420**) gehört normalerweise ein Betriebssystem (**421**), eine oder mehrere Anwendungen (**422**) und Programmdateien (**424**). Die Anwendung (**422**) kann ein Verfahren zum Hosten und Publizieren von Softwarepaketen beinhalten. Die Programmdateien (**424**) enthalten das Speichern von Anweisungen, die, wenn sie von der einen oder mehreren Verarbeitungsvorrichtungen ausgeführt werden, ein Verfahren zum Hosten und Publizieren von Softwarepaketen implementieren. (**423**). Bei einigen Ausführungsformen kann die Anwendung (**422**) so eingerichtet werden, um mit Programmdateien (**424**) auf einem Betriebssystem (**421**) zu arbeiten.

[0023] Das Computergerät (**400**) kann zusätzliche Funktionen oder Funktionalitäten haben und zusätzliche Schnittstellen, um die Kommunikation zwischen

der Grundkonfiguration (**401**) und jeglichen erforderlichen Geräten und Schnittstellen zu ermöglichen.

[0024] Der Systemspeicher (**420**) ist ein Beispiel eines Computerspeichermediums. Zu den Computerspeichermedien zählen unter anderem RAM, ROM, EEPROM, Flashspeicher und andere Speichertechnologie, CD-ROM, Digital Versatile Disks (DVD) sowie andere optische Speicher, Magnetkassetten, Magnetband, Magnetplattenspeicher und andere Magnetspeichergeräte oder jedes sonstige Medium, das zum Speichern der gewünschten Informationen verwendet werden und auf das durch das Computergerät **400** zugegriffen werden kann. Alle Computerspeichermedien dieser Art können ein Teil des Geräts (**400**) sein.

[0025] Das Computergerät (**400**) kann als ein Teil einer mobilen elektronischen Gerät eines kleinen Formfaktors, wie beispielsweise eines Mobiltelefons, eines Smartphones, eines PDA (Minicomputer), eines persönlichen Media-Player, eines Tablet-Computer (Tablet), eines drahtlosen Web-Watch-Geräts, eines persönlichen Headset-Geräts, eines anwendungsspezifischen Geräts oder eines Hybrid-Geräts implementieren werden, das eine der oben genannten Funktionen umfasst. Das Computergerät (**400**) kann auch als Personal Computer implementiert werden mit sowohl Laptop- als auch Nicht-Laptopkonfiguration.

[0026] Die vorangegangene detaillierte Beschreibung legt verschiedene Ausführungsformen der Geräte und/oder Prozesse über die Nutzung von Blockdiagrammen, Flowcharts und/oder Beispiele dar. Insofern, als solche Blockdiagramme, Flussdiagramme und/oder Beispiele eine oder mehrere Funktionen und/oder Tätigkeiten enthalten, ist es für Fachleute offensichtlich, dass jede Funktion und/oder Tätigkeit innerhalb solcher Blockdiagramme, Flussdiagramme oder Beispiele implementiert werden können, einzeln und/oder zusammen, durch eine breite Palette an Hardware, Software, Firmware oder praktisch jeder möglichen Kombination davon. In einer Ausführung können verschiedene Bestandteile des hier beschriebenen Themas über anwendungsspezifische, integrierte Schaltkreise (ASICs), feldprogrammierbare Gate-Arrays (FPGAs), digitale Signalprozessoren (DSPs), sonstige integrierte Formate oder als Web-Service enthalten. Dennoch werden die Fachkundigen feststellen, dass einige Aspekte der hier dargelegten Ausführungen, teilweise oder gänzlich, ebenso in integrierten Kreisläufen implementiert werden können, als ein oder mehrere Computerprogramme, die auf einem oder mehreren Computern ausgeführt werden, als ein oder mehrere Programme, die auf einem oder mehreren Prozessoren ausgeführt werden, als Firmware oder einer beliebigen Kombination davon; und dass ein Entwurf der Kreisläufe und/oder des Schreibens des Codes für die Software und/oder

Firmware unter Berücksichtigung der vorliegenden Offenbarung eine große Leistung wäre. Außerdem werden Fachleute verstehen, dass die Mechanismen dieses hier beschriebenen Gegenstands in der Lage sind, als ein Programmprodukt in einer Vielzahl von Formen verteilt zu werden, und dass eine veranschaulichende Ausführungsform des hier beschriebenen Gegenstands unabhängig von der besonderen Art des nicht-flüchtigen signalführenden Mediums gilt, das für die tatsächliche Verteilung verwendet wird. Beispiele für ein nicht-transitorisches Signal tragenden Mediums umfassen, ohne Einschränkung darauf, Folgendes: Floppy Disk, Festplattenlaufwerk, CD (Compact Disc), DVD (Digital Video Disk), digitales Band, Computerspeicher usw.; sowie ein Medium des Übertragungstyps, wie beispielsweise ein digitales und/oder analoges Kommunikationsmedium. (z. B. ein Glasfaserkabel, ein Wellenleiter, eine kabelgebundene Kommunikationsverbindung, eine kabellose Kommunikationsverbindung usw.)

[0027] Was die Nutzung von im Wesentlichen allen Mehrzahl- und/oder Einzahlbegriffen hierin betrifft, können die Experten je nach Eignung für den Kontext und/oder die Anwendung die Mehrzahlform zur Einzahlform und vice versa bilden. Die verschiedenen Singular-/Plural-Permutationen können hierin ausdrücklich aus Gründen der Klarheit dargelegt.

[0028] Folglich wurden bestimmte Ausführungsformen des Gegenstands beschrieben. Weitere Ausführungsformen gehören zum Umfang der folgenden Ansprüche. In einigen Fällen können die in den Ansprüchen beschriebenen Handlungen in einer anderen Reihenfolge ausgeführt werden und dennoch erwünschte Ergebnisse erzielen. Zusätzlich erfordern die in den beigefügten Figuren dargestellten Prozesse nicht notwendigerweise die bestimmte gezeigte Reihenfolge oder aufeinanderfolgende Reihenfolge, um erwünschte Ergebnisse zu erzielen. Bei bestimmten Implementierungen können Multitasking und eine Parallelbearbeitung vorteilhaft sein.

Schutzansprüche

1. System zum Suchen von Kandidaten qualifizierter Namen für mindestens einen einfachen Namen in einer Quellcodedatei, wobei das Verfahren Folgendes umfasst:
ein oder mehrere Verarbeitungsvorrichtungen und ein oder mehrere Speichervorrichtungen, die Anweisungen speichern, die bei Ausführung durch ein oder mehrere Verarbeitungsvorrichtungen, die eine oder mehreren Verarbeitungsgeräte zu Folgendem veranlassen:
empfangen einer Quellcodedatei mit mindestens einem einfachen Namen für den ein entsprechender qualifizierter Name erforderlich ist;

analysieren der Quellcodedatei, um die einfachen Namen in der Datei und die qualifizierten Namen in der Datei zu ermitteln;
für den mindestens einen einfachen Namen, für den ein entsprechender qualifizierter Name erforderlich ist, Durchsuchen einer Quellcodegrafik, um potenzielle qualifizierte Namen für den mindestens einfachen Namen zu finden; und
nach Empfang einer Liste potenzieller Kandidaten qualifizierter Namen aus der Quellcodegrafik, Ausgeben der Liste.

2. System nach Anspruch 1, worin die Anweisungen ferner bewirken, dass der eine oder die mehreren Prozessoren:
Build-Abhängigkeiten und Build-Ziele für spezifische qualifizierte Namen ermitteln, nachdem ein qualifizierter Name ausgewählt ist.

3. System nach Anspruch 1 oder 2, worin die Anweisungen ferner bewirken, dass der eine oder die mehreren Prozessoren:
vor Ausgeben der Liste die Liste mit zusätzlichen Informationen filtern oder anreichern.

4. System nach Anspruch 3, worin Filtern oder Anreichern der Liste beinhaltet:
filtern der Liste basierend darauf, wie oft ein qualifizierter Namen eines bestimmten Typs von anderen Quellcodedateien verwendet wurde vor Ausgeben der Liste.

5. System nach Anspruch 3 oder 4, worin Filtern oder Anreichern der Liste beinhaltet:
erstellen einer Rangordnung der Liste basierend auf der Häufigkeit, mit der ein bestimmter qualifizierter Name von anderen Entwicklern in anderen Teilen des Quellcode-Repositories verwendet wurde.

6. System nach einem der Ansprüche 3 bis 5, worin Filtern oder Anreichern der Liste beinhaltet:
ermitteln, ob ein qualifiziertes Namenspaket von der Quellcodedatei sichtbar ist, in der auf einen einfachen Namen, der potenziell dem qualifizierten Namen entspricht, aktuell Bezug genommen wird; und
als Reaktion auf die Ermittlung, dass der qualifizierte Name von der Quellcodedatei sichtbar ist, die Bereitstellung des Namens des qualifizierten Namenspakets als potenzieller Kandidat für den qualifizierten Namen.

7. System nach einem der Ansprüche 4 bis 6, worin Filtern oder Anreichern der Liste beinhaltet:
ermitteln des Ausmaßes an Komplikationen, die ein qualifiziertes Namenspaket zum Build-Prozess hinzufügen kann, einschließlich des Anstiegs von Zeitaufwand oder Ressourcen, um die Quellcodedatei aufzubauen; und
als Reaktion auf die Ermittlung, dass das Ausmaß der Komplikation, die ein qualifiziertes Namenspaket

dem Build-Prozess hinzufügt, ein nachhaltiges Ausmaß ist, Hinzufügen des qualifizierten Namenspakets zur Liste der potenziellen Kandidaten qualifizierter Namen.

8. System nach einem der Ansprüche 1 bis 7, worin die Anweisungen ferner bewirken, dass der eine oder die mehreren Prozessoren:
den qualifizierten Namen ohne Interaktion mit dem Nutzer automatisch auswählen, falls die Liste nur einen potenziellen qualifizierten Namen enthält.

9. System nach einem der Ansprüche 1 bis 8, worin die Anweisungen ferner bewirken, dass der eine oder die mehreren Prozessoren:
eine Oberfläche für den Empfang einer Auswahl eines qualifizierten Namens aus der Liste potenzieller Kandidaten qualifizierter Namen bereitstellen.

Es folgen 4 Seiten Zeichnungen

Anhängende Zeichnungen

```
1. import java.util.List;  
2. public class classDemo{  
3. List theList = new ArrayList();  
4. ....  
}
```

FIG. 1

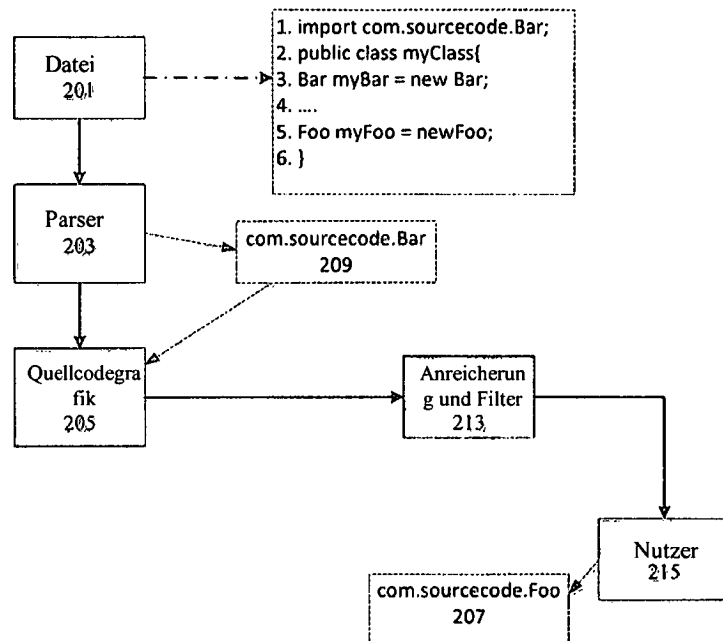


FIG. 2

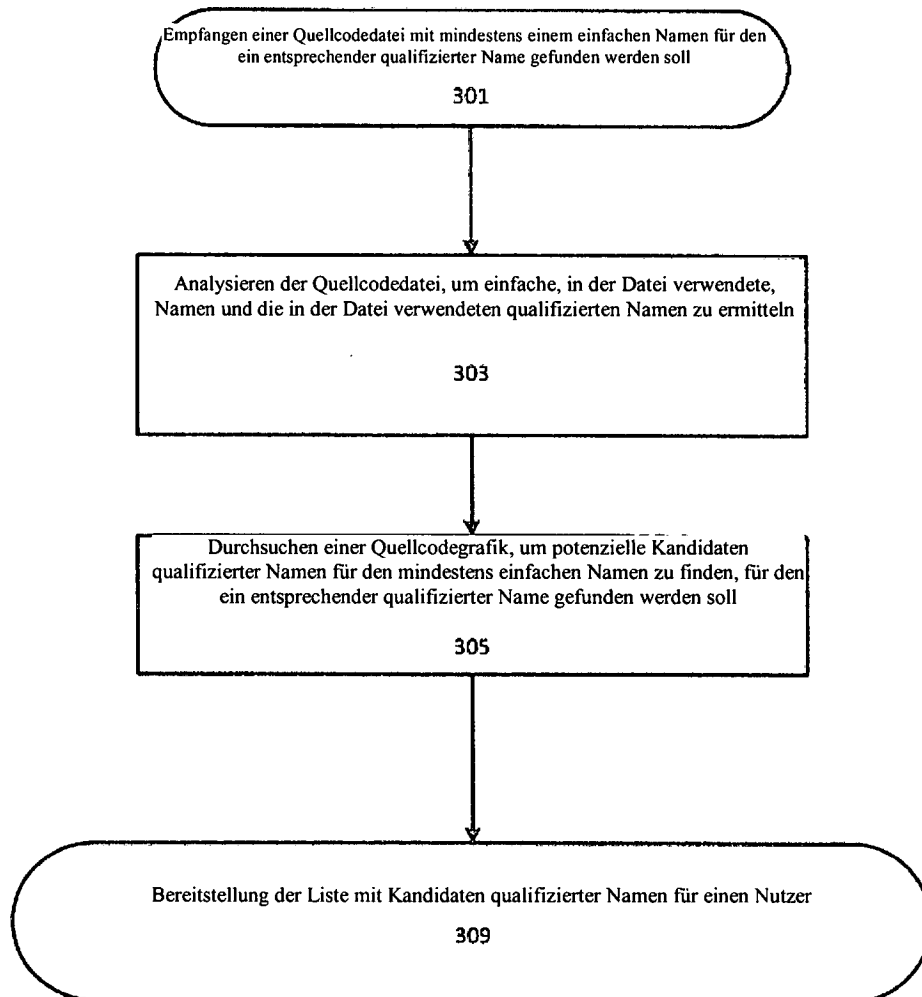


FIG. 3

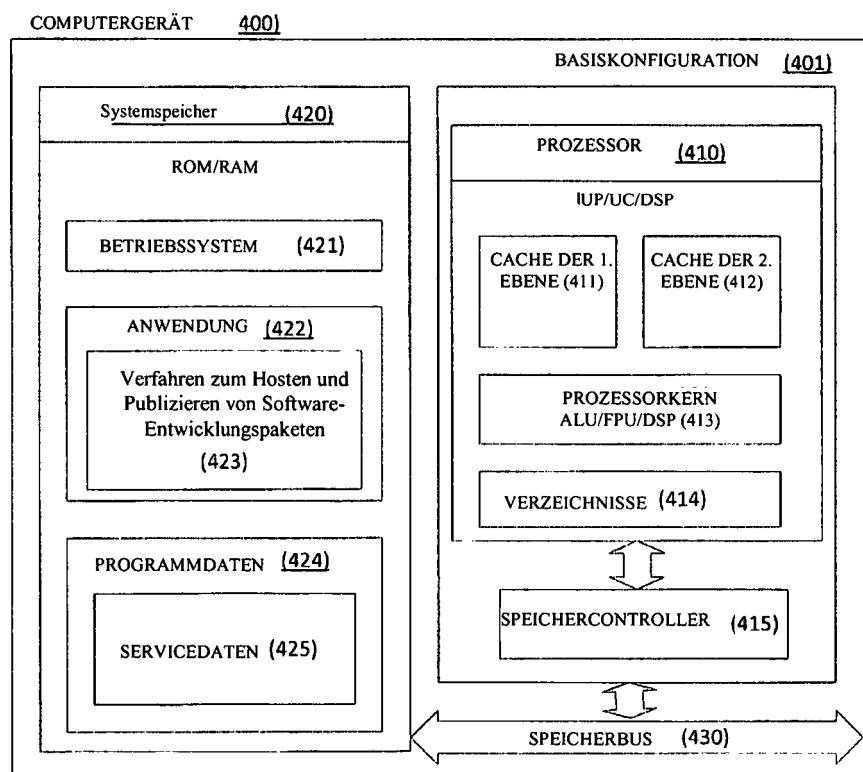


FIG. 4