



(51) International Patent Classification:

G06F 21/10 (2013.01) G06F 21/64 (2013.01)

G06F 21/12 (2013.01) H04L 29/06 (2006.01)

G06F 21/62 (2013.01)

(21) International Application Number:

PCT/MY2019/050128

(22) International Filing Date:

24 December 2019 (24.12.2019)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

PI 2018002919 26 December 2018 (26.12.2018) MY

(71) Applicant: MIMOS BERHAD [MY/MY]; Technology

Park Malaysia, Kuala Lumpur, 57000 (MY).

(72) Inventors: ABU BAKAR, Rudhuwan Bin; Mimos

Berhad, Technology Park Malaysia, Kuala Lumpur, 57000

(MY). MD SARKAN, Hasliza Binti; Mimos Berhad,

Technology Park Malaysia, Kuala Lumpur, 57000 (MY).

WAN RAMLI, Wan Mohd Firdaus Bin; Mimos Berhad,

Technology Park Malaysia, Kuala Lumpur, 57000 (MY).

(74) Agent: MIRANDAH ASIA (MALAYSIA) SDN BHD;

Suite 3B-19-3, Plaza Sentral, Jalan Stesen Sentral 5, Kuala

Lumpur, 50470 (MY).

(81) Designated States (unless otherwise indicated, for every

kind of national protection available): AE, AG, AL, AM,

AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,

CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO,

DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,

HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP,

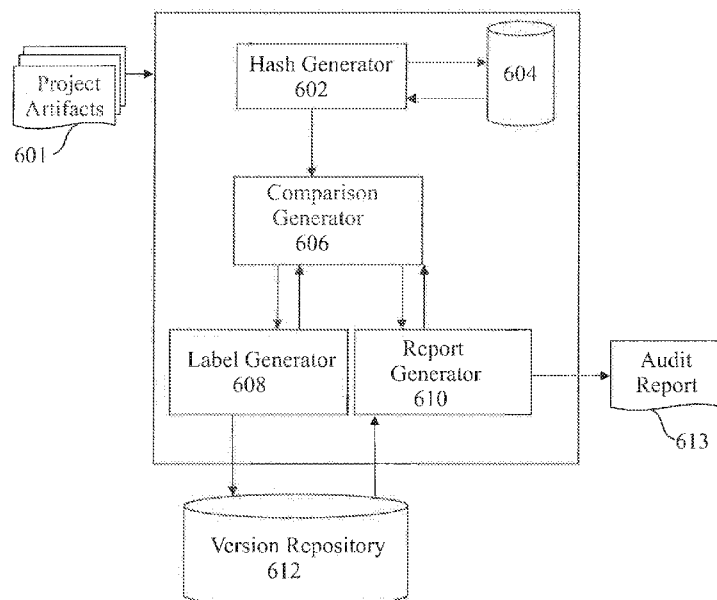
KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME,

MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ,

(54) Title: SYSTEM AND METHOD TO GENERATE HASH VALUES FOR PROJECT ARTIFACT AND SOFTWARE PACKAGE

600

FIG. 6



(57) Abstract: A system and method to generate hash values for project artifacts (601) and software package before being stored in a version repository. The system comprises a hash generator (602), version repository (612), comparison generator (606), label generator (608), and report generator (610). The hash generator (602) generates hash values corresponding to each project artifacts. The version repository (612) stores the project artifacts (601) with the corresponding hash values. The comparison generator (606) matches the project artifacts (601) with the hash values and compares the matched project artifacts (601) against a delivery checklist to determine an existence of the project artifacts (601). The label generator (608) performs the labeling operation on the project artifacts (601) matched with the hash values to obtain a release label. The label generator (608) uploads the release label in version repository (612). The report generator (610) receives the release label to generate an audit report (613).

OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
- *of inventorship (Rule 4.17(iv))*

Published:

- *with international search report (Art. 21(3))*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*
- *in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE*

SYSTEM AND METHOD TO GENERATE HASH VALUES FOR PROJECT ARTIFACT AND SOFTWARE PACKAGE

FIELD OF INVENTION

5

[0001] The present invention relates to integrity and authenticity of the project artifacts and software package, in particular to system and method to generate hash values for a project artifact and a software package before being stored in a version repository.

10

BACKGROUND

[0002] Typically, project artifact is a tangible by-product produced during software development. The developers document the project artifacts and store in a version repository so it can be retrieved upon demand. Examples of project artifacts including but not limited to use cases, class diagrams, UML models, requirements and design documents, etc. Generally, project artifacts describe functions, architecture, and design of the software.

15

[0003] Further, project artifacts needed to be baselined (labeled) and then an audit report needs to be generated to prepare for a release. Currently, releases rely on control procedures to determine and confirm the released artifacts. Thus the verification is performed based on the audit report. However, there is no verification method and system exist to check the authenticity and integrity of the released project artifacts during the labeling and audit report generation processes. Even if the baseline package is tampered, it will be deemed as "good for release" as there is no counter check mechanism implemented within the process to halt the release.

20

[0004] US patent number 7,685,425 B1 issued to Wright, et al. discloses a method for a secure serving of digitally signed computer files. The method receives a request from other computer for access to one or more computer files stored on a server computer. Each said computer file is having an associated unique digital signature created using a signing key from at least one of a plurality of third parties. The signature is stored on the server computer. In response to the request, the method retrieves at least one requested computer file. Further, in response to the request, the method retrieves the digital

25

30

signatures directly at the server computer. Further, the method validates the unique digital signatures associated with the requested computer file. Then the method provides the other computer with access to the requested computer file only if the digital signature associated with the requested computer file is valid. In case the digital signature is invalid, no further digital signature is applied by the server computer, and maliciously altered documents are not sent to the other computer. However, the method for a secure serving of digitally signed computer files is limited to the security of a Webpage, and the computer files or documents reside on a web server. Further, the Webpages are served only when the hash values and digital signature are valid.

[0005] US patent number 5,724,425 issued to Chang, et al. discloses an apparatus for enhancing software security and for distributing software by utilizing public key encryption techniques. The apparatus includes a first computer which is provided with source code to be protected. Further, a software application writer's private key and an application writer's license provided to the first computer. The application writer's license includes identifying information such as the application writer's name as well as the application writer's public key. The first computer executes a compiler program to compile the source code into binary code and computes a message digest for the binary code. The first computer then encrypts the message digest using the application writer's private key, such that the encrypted message digest is defined as a digital "signature" of the application writer. Further, a software passport is generated which includes the application writer's digital signature, the application writer's license, and the binary code. Then software passport is distributed to a user using any number of software distribution models. However, the apparatus for enhancing software security and for distributing software is applied for product authentication, and the binary packages can only be executed if the generated key matched with the pre-generated key.

[0006] It is, therefore, a need for a reliable, and efficient system and method to generate hash values for a project artifact and a software package before being stored in a version repository to ensure the integrity and authenticity of the project artifacts and software package.

SUMMARY

[0007] There is a need for a system and method to ensure project artifacts and software package integrity and authenticity before release.

[0008] The present invention mainly deals the technical problems existing in the prior art.
5 in response to these problems, the present invention provides a system and method for generating hash value or key identifier for project artifacts and software package before being stored into a versioning repository.

[0009] An aspect of the present invention relates to a system to generate hash values for a project artifact and a software package before being stored in a version repository. The
10 system comprises a hash generator, a temporary storage, a version repository, a comparison generator, a label generator, and a report generator. The hash generator generates one or more hash values corresponding to each project artifacts. The temporary storage receives the generated hash values and the project artifacts from the hash generator. The version repository stores the project artifacts with the corresponding hash
15 values. The comparison generator retrieves the project artifacts and the hash values from the version repository and matches the project artifacts with the hash values. The comparison generator further compares the matched project artifacts against a delivery checklist to determine an existence of the project artifacts in the version repository during a labeling operation. The label generator performs the labeling operation on the project
20 artifacts matched with the hash values to obtain a release label. The label generator uploads the release label in the version repository. The report generator receives the release label from the version repository and generates an audit report.

[0010] In another aspect of the present invention, the comparison generator examines a date of the project artifacts in the version repository against a baseline date during the
25 labeling operation.

[0011] In a further aspect of the present invention, the report generator generates an exception message on determining a mismatch of the hash values during of the comparison of the hash values and the project artifacts.

[0012] In yet a further aspect of the present invention, the version repository performs a
30 backend verification of the hash values for each project artifacts.

[0013] Yet, an aspect of the present invention relates to a method for generating hash values for a project artifact and a software package before being stored in a version repository. The method comprises a step of generating one or more hash values corresponding to each project artifacts through a hash generator. The method then includes a step of uploading the generated hash values and the project artifacts into a temporary storage. Further, the method includes the step of storing the project artifacts with the corresponding hash values in a version repository. The method then includes the step of retrieving the project artifacts and the hash values from the version repository and matches the project artifacts with the hash values through a comparison generator.

[0014] Furthermore, the method includes the step of comparing the matched project artifacts against a delivery checklist through the comparison generator to determine the existence of the project artifacts in the version repository during a labeling operation. The method then includes a step of performing the labeling operation on the project artifacts matched with the hash values to obtain a release label through a label generator. The label generator uploads the release label in the version repository. Further, the method includes the step of receiving the release label from the version repository through a report generator to generate an audit report. In yet another aspect, the audit report is stored in the version repository. The method then includes the step of releasing the project artifacts.

[0015] Accordingly, one advantage of the present invention is that it provides a robust auditing process by generating the audit report only after verifying the project artifacts as “good for release”.

[0016] Accordingly, one advantage of the present invention is that it integrates mechanisms into the current release and auditing processes to validate the authenticity and integrity of the project artifacts and software package before release.

[0017] Accordingly, one advantage of the present invention is that it generates a hash value and key to uniquely identify the project artifacts at baseline time.

[0018] Other features of embodiments of the present invention will be apparent from accompanying drawings and from the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] In the figures, similar components and/or features may have the same reference label. Further, various components of the same type may be distinguished by following the reference label with a second label that distinguishes among the similar components. If only the first reference label is used in the specification, the description applies to any one of the similar components having the same first reference label irrespective of the second reference label.

[0020] FIG. 1 illustrates a flowchart of a method for generating hash values for a project artifact and a software package before being stored in a version repository, in accordance with an embodiment of the present invention.

[0021] FIG. 2 illustrates an exemplary block diagram of uploading the project artifacts with the corresponding hash values in a version repository, in accordance with an embodiment of the present invention.

[0022] FIG. 3 exemplifies a process to upload the project artifacts into the version repository, in accordance with an embodiment of the present invention.

[0023] FIG. 4 illustrates an exemplary block diagram of baseline project artifacts for release, in accordance with an embodiment of the present invention.

[0024] FIG. 5 illustrates an exemplary block diagram of generating an audit report, in accordance with an embodiment of the present invention.

[0025] FIG. 6 illustrates a system for generating hash values for a project artifact before being stored in a version repository, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

[0026] Systems and methods are disclosed for generating hash values for a project artifact and a software package before being stored in a version repository. Embodiments of the present invention include various steps, which will be described below. The steps may be performed by hardware components or may be embodied in machine-executable

instructions, which may be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, steps may be performed by a combination of hardware, software, firmware, and/or by human operators.

[0027] Various methods described herein may be practised by combining one or more machine-readable storage media containing the code according to the present invention with appropriate standard computer hardware to execute the code contained therein. An apparatus for practicing various embodiments of the present invention may involve one or more computers (or one or more processors within a single computer) and storage systems containing or having network access to computer program(s) coded in accordance with various methods described herein, and the method steps of the invention could be accomplished by modules, routines, subroutines, or subparts of a computer program product.

[0028] The present invention discloses a system and method whereby the hash value is paired and stored with the project artifacts and release package in the version repository. Further, the hash value is compared to ensure the integrity and authenticity of the project artifacts and release package during the labeling operation. The method further checks existence of the project artifacts in a version repository against delivery checklist during the labeling operation. Then the method checks date of the project artifacts in the version repository against the planned baseline date during the labeling operation. Further, the method generates and matches the hash values when uploading a release label into a report form for report generation. The method then generates an exception message if the hash values are mismatched during the process of comparison.

[0029] Although the present invention has been described with the purpose of generating hash values for a project artifact and a software package before being stored in a version repository, it should be appreciated that the same has been done merely to illustrate the invention in an exemplary manner and any other purpose or function for which explained structures or configurations could be used, is covered within the scope of the present invention.

[0030] Exemplary embodiments will now be described more fully hereinafter with reference to the accompanying drawings, in which exemplary embodiments are shown. This invention may, however, be embodied in many different forms and should not be

construed as limited to the embodiments set forth herein. These embodiments are provided so that this invention will be thorough and complete and will fully convey the scope of the invention to those of ordinary skill in the art. Moreover, all statements herein reciting embodiments of the invention, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future (i.e., any elements developed that perform the same function, regardless of structure).

[0031] Thus, for example, it will be appreciated by those of ordinary skill in the art that the diagrams, schematics, illustrations, and the like represent conceptual views or processes illustrating systems and methods embodying this invention. The functions of the various elements shown in the figures may be provided through the use of dedicated hardware as well as hardware capable of executing associated software. Similarly, any switches shown in the figures are conceptual only. Their function may be carried out through the operation of program logic, through dedicated logic, through the interaction of program control and dedicated logic, or even manually, the particular technique being selectable by the entity implementing this invention. Those of ordinary skill in the art further understand that the exemplary hardware, software, processes, methods, and/or operating systems described herein are for illustrative purposes and, thus, are not intended to be limited to any particular name.

[0032] Specific details are given in the following description to provide a thorough understanding of the embodiments. However, it will be understood by one of ordinary skill in the art that the embodiments may be practised without these specific details. For example, circuits, systems, networks, processes, and other components may be shown as components in block diagram form in order not to obscure the embodiments in unnecessary detail. In other instances, well-known circuits, processes, algorithms, structures, and techniques may be shown without unnecessary detail to avoid obscuring the embodiments.

[0033] FIG. 1 illustrates a flowchart **100** of a method for generating hash values for a project artifact and a software package before being stored in a version repository, in accordance with an embodiment of the present invention. The method initiates with a step **104** of generating one or more hash values corresponding to each project artifacts through

a hash generator. The method then includes a step **106** of uploading the generated hash values and the project artifacts into a temporary storage. Further, the method includes the step **108** of storing the project artifacts with the corresponding hash values in a version repository. The method then includes the step **110** of retrieving the project artifacts and the hash values from the version repository and matches the project artifacts with the hash values through a comparison generator.

[0034] Furthermore, the method includes the step **112** of comparing the matched project artifacts against a delivery checklist through the comparison generator to determine an existence of the project artifacts in the version repository during a labeling operation. In one exemplified illustrations, the delivery checklist may include document title, document type, document category, security level, document number and file name. Further, the document categories may include plan, requirement, design, procedure/SOP/process flow/work instructions/etc., report, manual, notice/release note/memo etc. Each of these categories may be pre-assigned with a code. The code may be one alphanumeric character. Further, the security level may include internal use only, general and public use, confidential proprietary and registered secret document. Similarly, each of these security levels may be assigned with a level code. The level code may also be at least one alphanumeric code. These codes may be adopted for forming the document number for indexing purposes. The document number may further include numeric code representing the year and versions of the document. Further, the document number and the document title may be adopted for forming the file name. In an embodiment, the comparison generator examines a date of the project artifacts in the version repository against a baseline date during the labeling operation.

[0035] The method then includes a step **114** of performing the labeling operation on the project artifacts matched with the hash values to obtain a release label through a label generator. Once the project artifacts verified against the delivery checklist and the hash values are verified, the label generator will generate a release label to the project artifacts in the version repository. Further, the method includes the step **116** of receiving the release label from the version repository through a report generator to generate an audit report. The audit report is stored in the version repository.

[0036] In an embodiment, the report generator generates an exception message on determining a mismatch of the hash values during comparison of the hash values. The

method then includes the step **118** of releasing the project artifacts. For audit reporting, the report generator pulls specific release label. Only release labels with the matching hashes are uploaded for report generation. In an embodiment, the version repository performs a backend verification of the hash values for each project artifacts.

5 **[0037]** FIG. 2 illustrates an exemplary block diagram **108** of uploading the project artifacts with the corresponding hash values in a version repository, in accordance with an embodiment of the present invention. At block **202**, the project artifacts are uploaded into the temporary storage (folder or repository). The project artifacts are secured with SHA 1 (Secure Hash Algorithm 1) which is a cryptographic hash function. At block **204**, the
10 hash value is generated and assigned for each project artifact. The hash values include public keys. At block **206**, the project artifacts are stored with the corresponding hash value in the version repository. At block **208**, the project artifacts along with the corresponding hash values are uploaded to the version repository.

[0038] FIG. 3 exemplifies a process **300** to upload the project artifacts into the version
15 repository, in accordance with an embodiment of the present invention. At step **302**, the hash generator generates hash values corresponding to each project artifacts. In one embodiment, Secure Hash Algorithm (SHA) can be adapted for generating hash values. In another embodiment, Message-Digest Algorithm (MD5) can be adapted for generating the hash value. At step **304**, hash values are assigned to the project artifacts and stored in
20 the version repository. At step **306**, a requirement document is prepared and stored together with the hash value in the version repository. Returning to the step **304**, the hash values assigned to the project artifacts are compared with the stored hash values at step **308**. At step **312**, if the hash values assigned to project artifacts and the stored hash values are different, the process performs a step **310** of re-assigning a new hash value to the
25 project artifacts. The re-assigned and stored hash value is return to the step **308** for comparison again. Similarly, if the newly assigned artifact's hash is different at the step **312**, the process loops to the step **310** to re-assign another hash value, again until the hash value is the same as the stored hash value. The process performs a step **314** of assigning and storing the hash value on determining the same hash value assigned to the
30 corresponding project artifacts.

[0039] At step **316**, the process examines a date of the project artifacts in the version repository against a baseline date during the labeling operation. If the baseline date is the

same, the process performs a step **318** of assigning a hash value for the baseline folder. If the baseline date is different, the process performs the step **310** of re-assigning a new hash value to the project artifacts. At step **320**, the hash values assigned to the project artifacts are matched with the stored hash values to upload the release label into report form to perform a step **324** of generating the audit report. At step **326**, the control loop ends.

[0040] FIG. 4 illustrates an exemplary block diagram **400** of baseline project artifacts for release, in accordance with an embodiment of the present invention. The project artifacts **406** with SHA-1 **408** are retrieved from the version repository **612**. Hash value **402** along with the public key **404** are retrieved from the version repository **612** to verify the hash value at **410**. At step **412**, the hash values are matched with each other. If the hash values do not match, the project artifacts are rejected and return to the step **108** of FIG. 1. At step **414**, in case the hash value is the same, the project artifacts are compared to the delivery checklist. If the project artifacts are not the same as compared to the delivery checklist, the project artifacts are rejected and return to the step **108** of FIG. 1.

[0041] If the project artifacts exist in the delivery list, at step **416**, the process performs a step to determine whether the project artifacts are meeting the baseline date as planned in the delivery checklist. If the project artifacts are not in the delivery list, the process returns to the step **108** of FIG. 1. If the project artifacts meet the baseline date as planned in the checklist, the process applies release label or release TAG **420** to baseline the project artifacts with SHA1 **422**. At step **418**, if the project artifacts miss the planned date, the date is re-planned and updated in the delivery checklist. The report generator **610** receives the release label from the version repository **612** (shown in FIG. 6) and generates an audit report **613**.

[0042] FIG. 5 illustrates an exemplary block diagram **500** of generating an audit report, in accordance with an embodiment of the present invention. The release label or release TAG **506** along with the SHA1 **508** retrieved from the version repository **612**. The hash value **402** along with the public key **404** are retrieved from the version repository **612** to verify the hash value at **410**. Then the process checks if there is any discrepancy in the hash value. At step **510**, if the hash value stored in the temporary storage (TAG folder) equals to the hash value stored in the version repository, the release label is uploaded for report generation. If the hash value stored in the temporary storage (TAG folder) does not equal to the hash value stored in the version repository, an exception message is

prompted, and the process returns to the steps performed by block diagram 400 of FIG. 4. The release label is uploaded in the report generator 610 to generate the audit report 613.

[0043] FIG. 6 illustrates a system 600 for generating hash values for a project artifact 601 before being stored in a version repository 612, in accordance with an embodiment of the present invention. The system 600 comprises a hash generator 602, a temporary storage 604, a version repository 612, a comparison generator 606, a label generator 608, and a report generator 610. The hash generator 602 generates one or more hash values corresponding to each project artifacts. The temporary storage 604 receives the generated hash values and the project artifacts from the hash generator 602. The version repository 612 stores the project artifacts 601 with the corresponding hash values.

[0044] The comparison generator 606 retrieves the project artifacts 601 and the hash values from the version repository 612 and matches the project artifacts 601 with the hash values. The comparison generator 606 further compares the matched project artifacts 601 against a delivery checklist to determine an existence of the project artifacts 601 in the version repository 612 during a labeling operation. The comparison generator 606 examines a date of the project artifacts in the version repository 612 against a baseline date during the labeling operation.

[0045] The label generator 608 performs the labeling operation on the project artifacts 601 matched with the hash values to obtain a release label. The label generator 608 uploads the release label in the version repository 612. The report generator 610 receives the release label from the version repository 612 and generates an audit report 613. The report generator 610 generates an exception message on determining a mismatch of the hash values during of the comparison of the hash values. The version repository 612 performs a backend verification of the hash values for each project artifacts 601.

[0046] Thus the present system and method incorporate security features (encryption and decryption) into the current labeling operation and configuration audit report generation. The hash value of each project artifact 601 (requirement book, etc.) will be generated each time the project artifacts 601 are uploaded into or downloaded from the version repository 612 and stored. During the release labeling operation, those values were compared to determine the authenticity and integrity of the project artifacts 601. The ones that have matching values will then be compared to the delivery checklist to determine

their existence and baseline date compliance. Only project artifacts **601** that met those criteria can be labeled for release in the version repository **612**. The releases are audited through the generated audit report **613**. For report preparation, specific release label is uploaded to the report generator **610**. Only release label with the matching hash values
5 are allowed to be uploaded to ensure the authenticity, and the integrity of the package maintained throughout the release process.

[0047] While embodiments of the present invention have been illustrated and described, it will be clear that the invention is not limited to these embodiments only. Numerous modifications, changes, variations, substitutions, and equivalents will be apparent to those
10 skilled in the art, without departing from the scope of the invention, as described in the claims.

CLAIMS:

1. A computer-implemented method to generate hash values for a project artifact, the method comprising steps of:
 - 5 generating (104) one or more hash values corresponding to each project artifacts through a hash generator;
 - uploading (106) the generated hash values and the project artifacts into a temporary storage;
 - storing (108) the project artifacts with the corresponding hash values in a version
10 repository;
 - retrieving (110) the project artifacts and the hash values from the version repository and matches the project artifacts with the hash values through a comparison generator;
 - comparing (112) the matched project artifacts against a delivery checklist through the comparison generator to determine an existence of the project artifacts in the version
15 repository during a labeling operation;
 - performing (114) the labeling operation on the project artifacts matched with the hash values to obtain a release label through a label generator, wherein the label generator uploads the release label in the version repository; and
 - receiving (116) the release label from the version repository through a report generator to
20 generate an audit report.
2. The method according to claim 1, wherein the comparison generator examines a date of the project artifacts in the version repository against a baseline date during the labeling operation.
25
3. The method according to claim 1, wherein the report generator generates an exception message on determining a mismatch of the hash values during of the comparison of the hash values and the project artifacts.
- 30 4. The method according to claim 1, wherein the version repository performs a backend verification of the hash values for each project artifacts.
5. The method according to claim 1, wherein the audit report is stored in the version repository.

6. The method according to claim 1, wherein the project artifacts are ready to be released after generation of the audit report.

5 7. A system (600) to generate hash values for a project artifact and a software package, the system (600) comprises:

a hash generator (602) to generate one or more hash values corresponding to each project artifacts (601);

10 a temporary storage (604) to receive the generated hash values and the project artifacts (601) from the hash generator (602);

a version repository (612) to store the project artifacts (601) with the corresponding hash values;

15 a comparison generator (606) to retrieve the project artifacts and the hash values from the version repository (612) and matches the project artifacts (601) with the hash values, wherein the comparison generator (606) further compares the matched project artifacts (601) against a delivery checklist to determine an existence of the project artifacts (601) in the version repository (612) during a labeling operation;

20 a label generator (608) to perform the labeling operation on the project artifacts (601) matched with the hash values to obtain a release label, wherein the label generator (608) uploads the release label in the version repository (612); and

a report generator (610) to receive the release label from the version repository (612) and generates an audit report.

25 8. The system according to claim 7, wherein the comparison generator (606) examines a date of the project artifacts (601) in the version repository against a baseline date during the labeling operation.

9. The system according to claim 7, wherein the report generator (610) generates an exception message on determining a mismatch of the hash values during of the comparison of the hash values and the project artifacts.

10. The system according to claim 7, wherein the version repository (612) performs a backend verification of the hash values for each project artifacts (601).

1/6

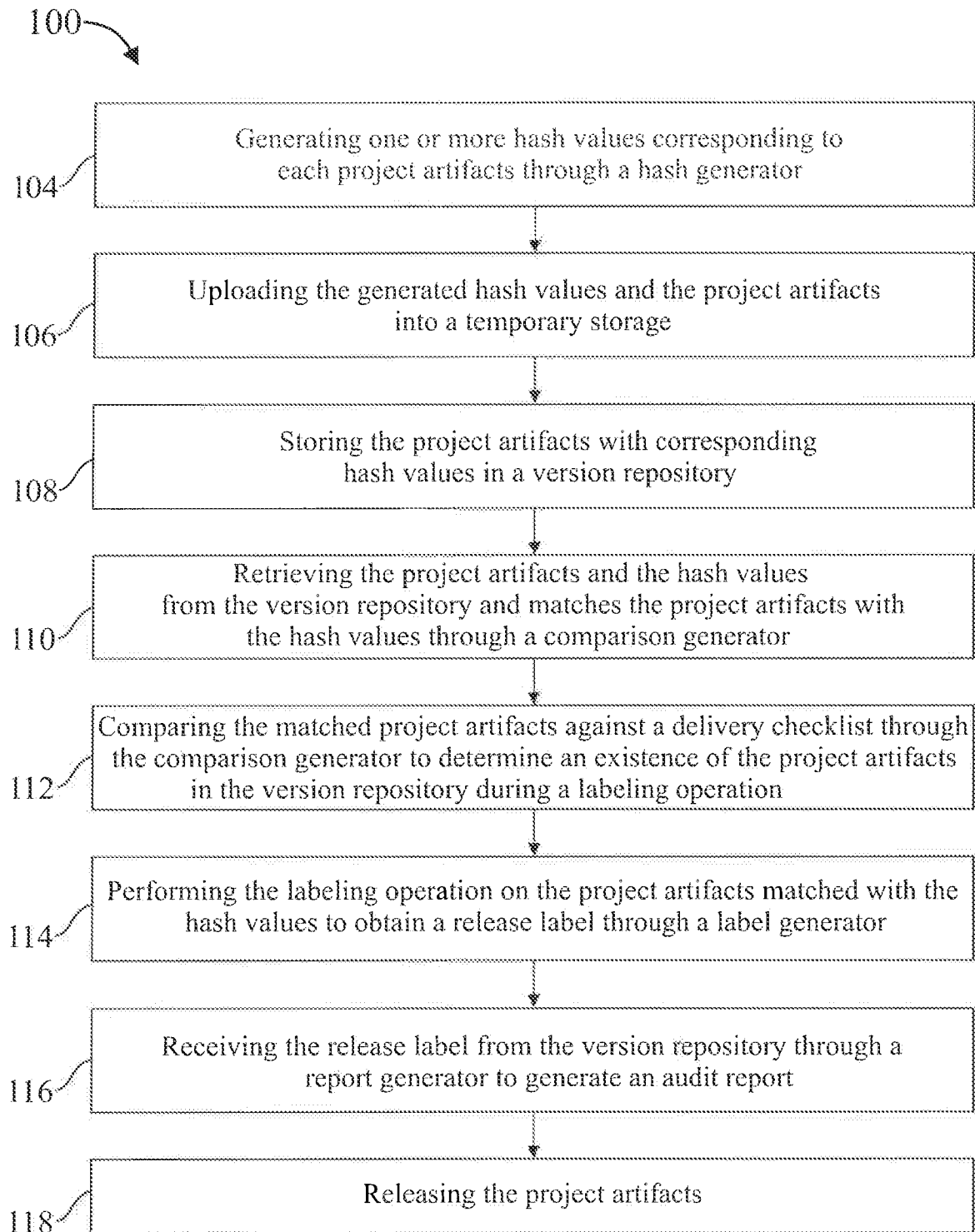


FIG. 1

2/6

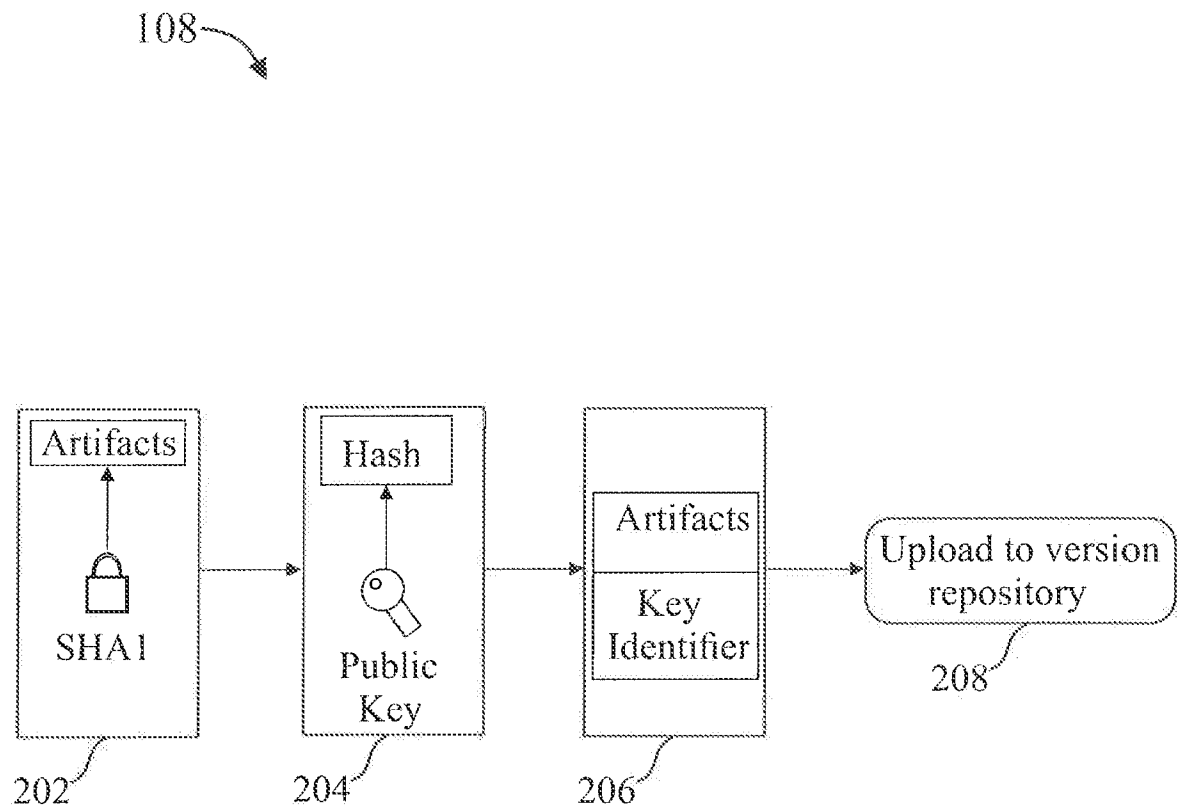


FIG. 2

3/6

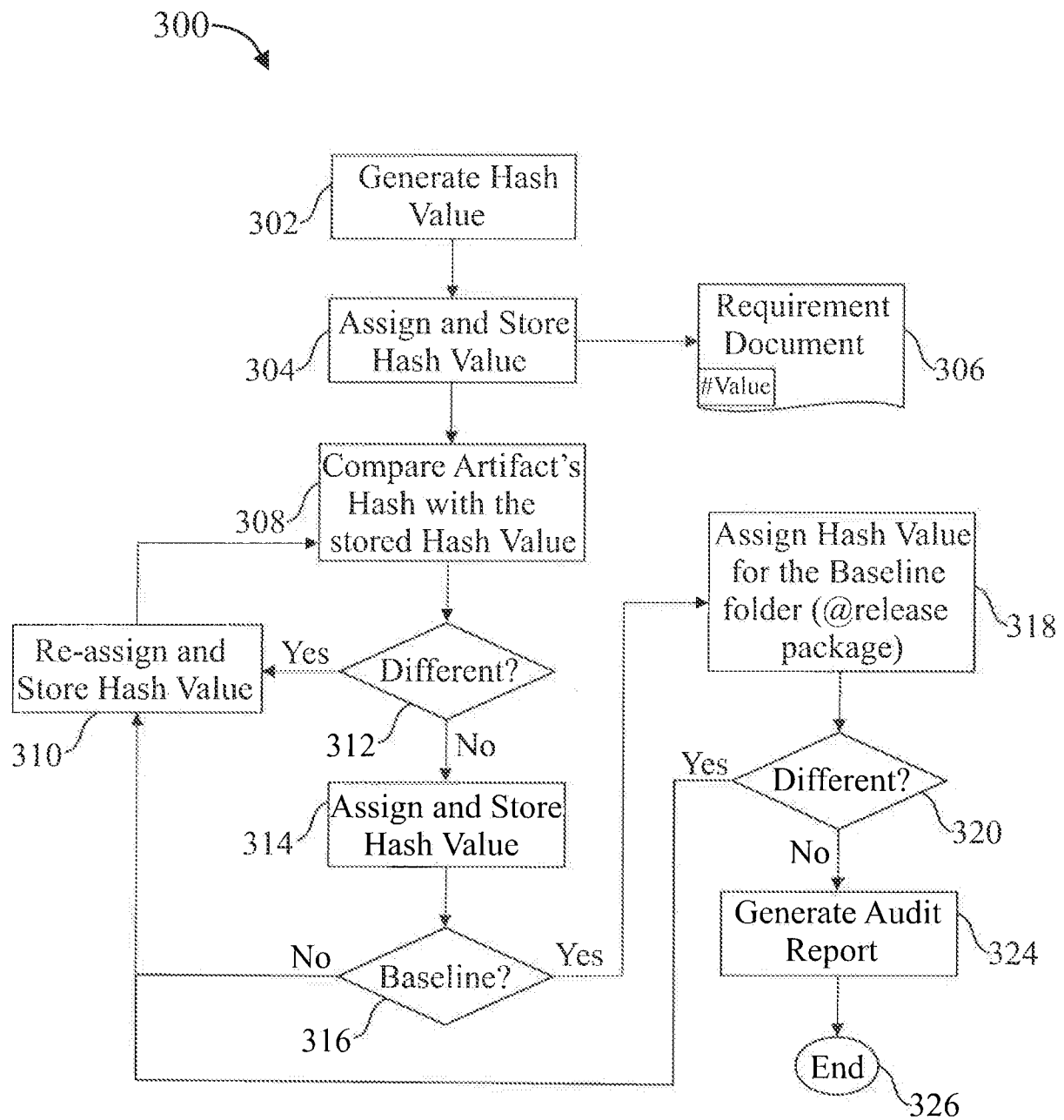
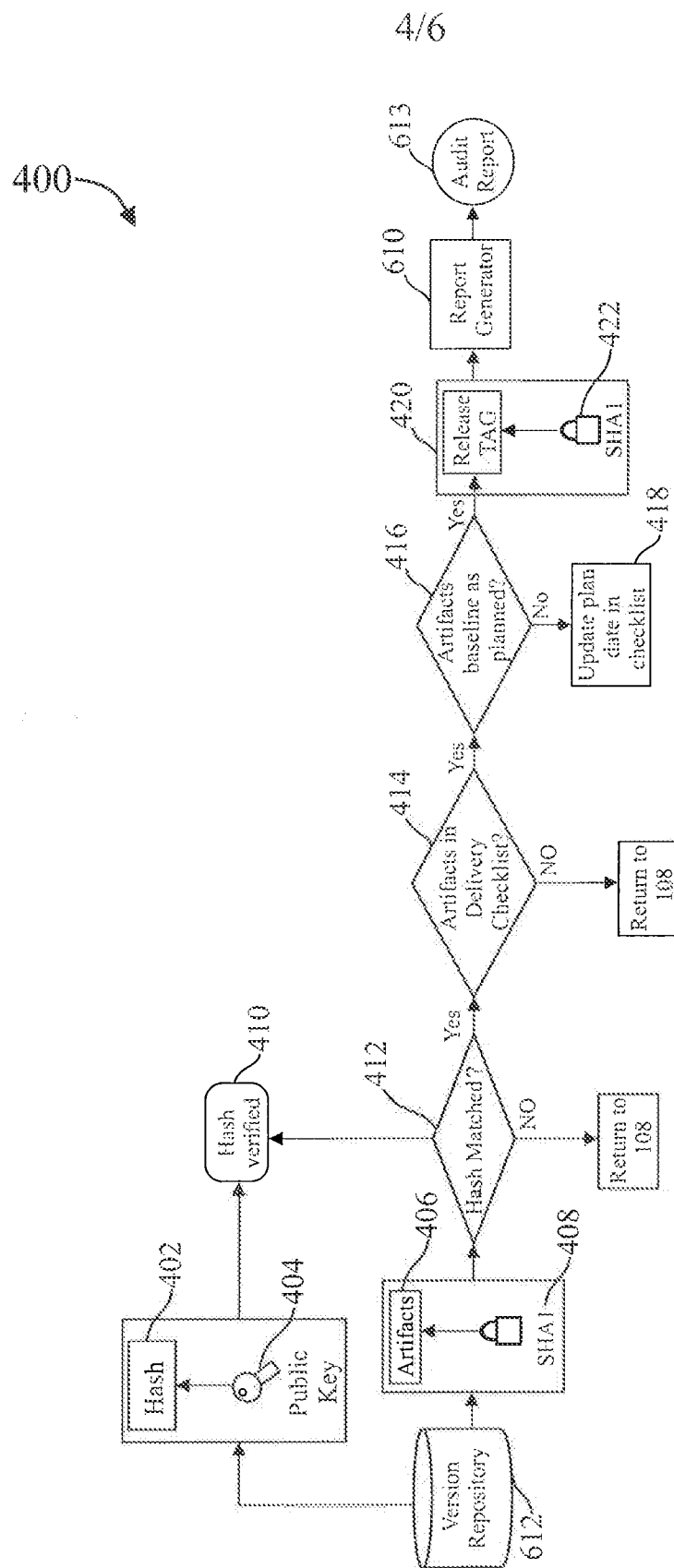


FIG. 3



5/6

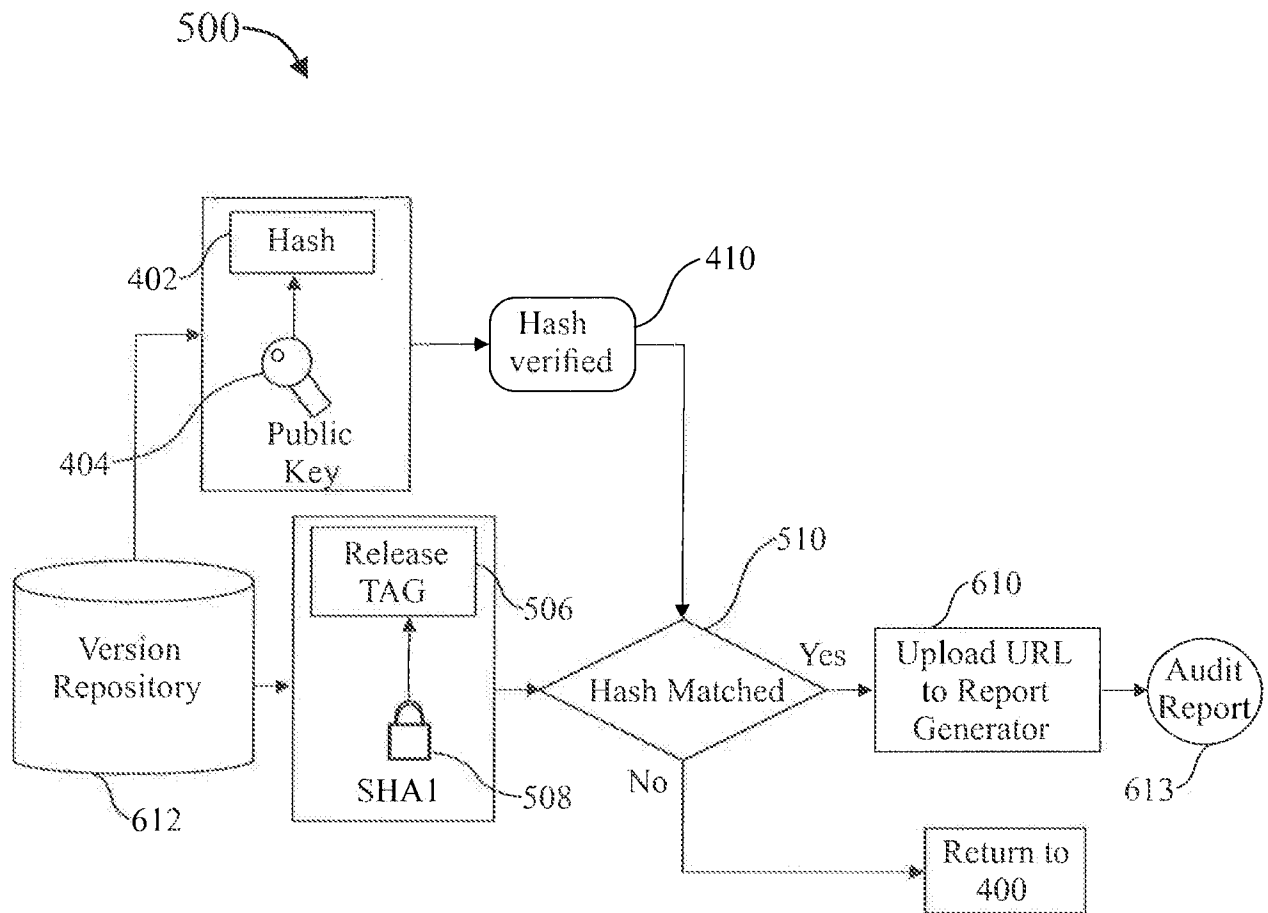


FIG. 5

6/6

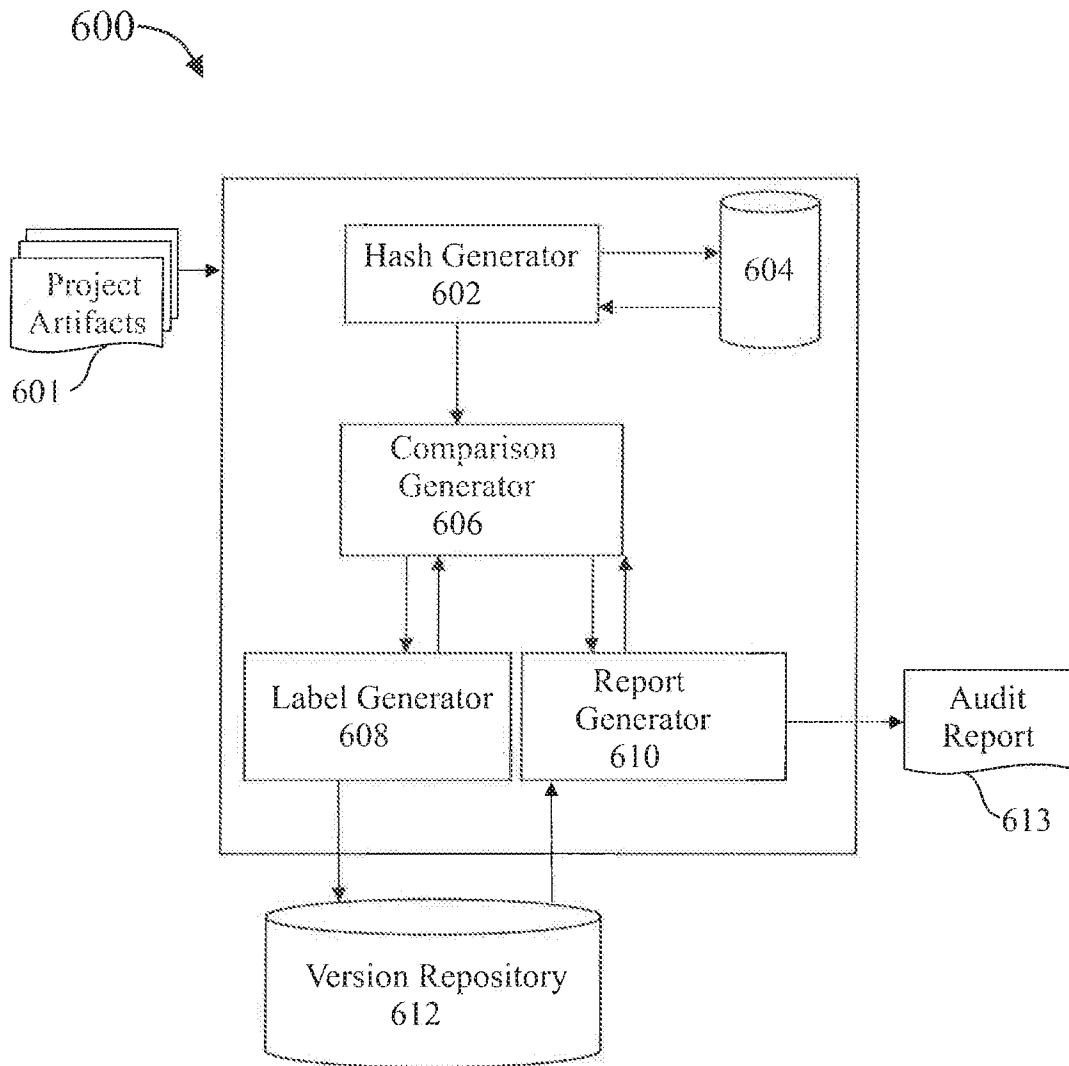


FIG. 6

A. CLASSIFICATION OF SUBJECT MATTER**G06F 21/10(2013.01)i, G06F 21/12(2013.01)i, G06F 21/62(2013.01)i, G06F 21/64(2013.01)i, H04L 29/06(2006.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 21/10; G06F 17/24; G06F 17/30; G06F 9/44; G06F 9/445; H04L 9/32; G06F 21/12; G06F 21/62; G06F 21/64; H04L 29/06

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: software, develop, artifact, valid, hash, release, deliver

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2010-0223469 A1 (RIAZ Y. HUSSAIN et al.) 02 September 2010 paragraphs [0040], [0042], [0045]; claim 1; and figures 6-7	1-10
Y	US 2017-0372247 A1 (INTUIT INC.) 28 December 2017 paragraphs [0055], [0062], [0108]	1-10
A	US 2012-0266154 A1 (JEB STUART THORLEY et al.) 18 October 2012 paragraphs [0105]-[0155]; and figures 1-7	1-10
A	US 2017-0357927 A1 (ACCENTURE GLOBAL SOLUTIONS LIMITED) 14 December 2017 paragraph [0042]; and figure 8	1-10
A	US 2009-0055795 A1 (RONALD D. FINLAYSON et al.) 26 February 2009 paragraph [0086]	1-10



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"D" document cited by the applicant in the international application

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

21 April 2020 (21.04.2020)

Date of mailing of the international search report

21 April 2020 (21.04.2020)

Name and mailing address of the ISA/KR

International Application Division

Korean Intellectual Property Office

189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea



Facsimile No. +82-42-481-8578

Authorized officer

KIM, Sung Hee

Telephone No. +82-42-481-3516



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/MY2019/050128

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2010-0223469 A1	02/09/2010	None	
US 2017-0372247 A1	28/12/2017	WO 2017-222567 A1	28/12/2017
US 2012-0266154 A1	18/10/2012	CA 2637168 A1	11/01/2009
		US 2011-0099543 A1	28/04/2011
		US 2012-0084764 A1	05/04/2012
		US 8117596 B2	14/02/2012
		US 8219985 B2	10/07/2012
		US 8739152 B2	27/05/2014
US 2017-0357927 A1	14/12/2017	None	
US 2009-0055795 A1	26/02/2009	US 9189757 B2	17/11/2015