



- (51) **International Patent Classification:**
G06F 9/44 (2006.01) *G06F 11/30* (2006.01)
- (21) **International Application Number:**
PCT/US2015/064667
- (22) **International Filing Date:**
9 December 2015 (09.12.2015)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (71) **Applicant:** HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP [US/US]; 11445 Compaq Center Drive West, Houston, Texas 77070 (US).
- (72) **Inventors:** SHMAYA, Daniel; Altalef st No 5, Shabazi st. No 26, 56100 Yehud (IL). BEN-YEHUDA, Itay; Altalef st No 5, Shabazi st. No 26, 56100 Yehud (IL). BUSI, Eliraz; Altalef st No 5, Shabazi st. No 26, 56100 Yehud (IL).
- (74) **Agents:** LEE, Rachel Jeong-Eun et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, Colorado 80528 (US).
- (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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, 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 the identity of the inventor (Rule 4.17(i))
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))

Published:

- with international search report (Art. 21(3))

(54) **Title:** SOFTWARE DEVELOPMENT MANagements

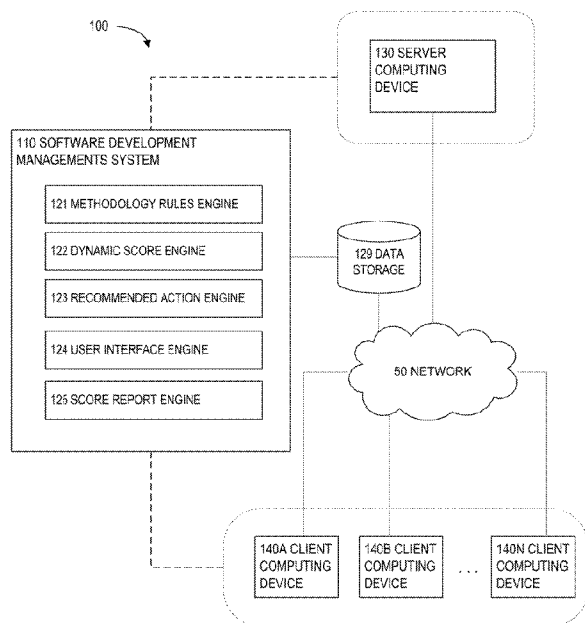


FIG. 1

(57) **Abstract:** Examples disclosed herein relate to software development managements. Some of the examples enable identifying a set of rules related to a software development methodology. Each rule of the set of rules may be associated with a rule-specific score. Some of the examples further enable monitoring a software development process, and determining, during the monitoring, whether at least one rule of the set of rules is invoked by an action performed by a user. If determined that a particular rule of the set of rules is invoked, a software development management score may be dynamically adjusted based on the rule-specific score associated with the particular rule. Some of the examples further enable providing a recommended action to increase the software development management score.

SOFTWARE DEVELOPMENT MANAGEMENTS

BACKGROUND

[0001] Developing software programs is a complex process that involves problem definition, requirements development, design, coding and debugging, and unit testing, integration, and system testing and maintenance, for example. A variety of software development methodologies exist to structure, plan, and manage this process of software development. Examples of such software development methodologies include agile software development, waterfall, lean development, crystal method, dynamic systems development model (DSDM), extreme programming (XP), and feature driven development (FDD).

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The following detailed description references the drawings, wherein:

[0003] FIG. 1 is a block diagram depicting an example environment in which various examples may be implemented as a software development managements system.

[0004] FIG. 2 is a block diagram depicting an example software development managements system.

[0005] FIG. 3 is a block diagram depicting an example machine-readable storage medium comprising instructions executable by a processor for software development managements.

[0006] FIG. 4 is a block diagram depicting an example machine-readable storage medium comprising instructions executable by a processor for software development managements.

[0007] FIG. 5 is a flow diagram depicting an example method for software development managements.

[0008] FIG. 6 is a flow diagram depicting an example method for software development managements.

DETAILED DESCRIPTION

[0009] The following detailed description refers to the accompanying drawings. Wherever possible, the same reference numbers are used in the drawings and the following description to refer to the same or similar parts. It is to be expressly understood, however, that the drawings are for the purpose of illustration and description only. While several examples are described in this document, modifications, adaptations, and other implementations are possible. Accordingly, the following detailed description does not limit the disclosed examples. Instead, the proper scope of the disclosed examples may be defined by the appended claims.

[0010] Developing software programs is a complex process that involves problem definition, requirements development, design, coding and debugging, and unit testing, integration, and system testing and maintenance, for example. Because of this, many organizations rely on a well-defined software development methodology to structure, plan, and manage this process of software development. Examples of a software development methodology may include, but not being limited to, agile software development, waterfall, lean development, crystal method, dynamic systems development model (DSDM), extreme programming (XP), and feature driven development (FDD).

[0011] A particular software development methodology sets forth and/or requires a particular set of rules to be followed by a user and/or a team of users. A “user,” as used herein, may refer to an administrator, a programmer, a product owner, a business analyst, a customer, and/or any other users who engage in the software development process. Wrong implementations of the required rules, due to lack of experience or misunderstanding of the process, can reduce the velocity and quality of the development. Moreover, in large enterprise environments, it may be technically challenging to manage and track the development progress of multiple development teams and their improvements over time.

[0012] Examples disclosed herein provide technical solutions to these technical challenges by providing a comprehensive guiding tool that helps users to follow a

set of rules required by a particular software development methodology to improve their compliance to the set of rules. Some of the examples enable identifying a set of rules related to a software development methodology. Each rule of the set of rules may be associated with a rule-specific score. Some of the examples further enable monitoring a software development process, and determining, during the monitoring, whether at least one rule of the set of rules is invoked by an action performed by a user. If determined that a particular rule of the set of rules is invoked, a software development management score may be dynamically adjusted based on the rule-specific score associated with the particular rule. Some of the examples further enable providing a recommended action to increase the software development management score.

[0013] The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting. As used herein, the singular forms “a,” “an,” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. The term “plurality,” as used herein, is defined as two or more than two. The term “another,” as used herein, is defined as at least a second or more. The term “coupled,” as used herein, is defined as connected, whether directly without any intervening elements or indirectly with at least one intervening elements, unless otherwise indicated. Two elements can be coupled mechanically, electrically, or communicatively linked through a communication channel, pathway, network, or system. The term “and/or” as used herein refers to and encompasses any and all possible combinations of one or more of the associated listed items. It will also be understood that, although the terms first, second, third, etc. may be used herein to describe various elements, these elements should not be limited by these terms, as these terms are only used to distinguish one element from another unless stated otherwise or the context indicates otherwise. As used herein, the term “includes” means includes but not limited to, the term “including” means including but not limited to. The term “based on” means based at least in part on.

[0014] FIG. 1 is an example environment 100 in which various examples may be implemented as a software development managements system 110. Environment 100 may include various components including server computing

device 130 and client computing devices 140 (illustrated as 140A, 140B, ..., 140N). Each client computing device 140A, 140B, ..., 140N may communicate requests to and/or receive responses from server computing device 130. Server computing device 130 may receive and/or respond to requests from client computing devices 140. Client computing devices 140 may be any type of computing device providing a user interface through which a user can interact with a software application. For example, client computing devices 140 may include a laptop computing device, a desktop computing device, an all-in-one computing device, a thin client, a workstation, a tablet computing device, a mobile phone, an electronic book reader, a network-enabled appliance such as a "Smart" television, and/or other electronic device suitable for displaying a user interface and processing user interactions with the displayed interface. While server computing device 130 is depicted as a single computing device, server computing device 130 may include any number of integrated or distributed computing devices serving at least one software application for consumption by client computing devices 140.

[0015] The various components (e.g., components 129, 130, and/or 140) depicted in FIG. 1 may be coupled to at least one other component via a network 50. Network 50 may comprise any infrastructure or combination of infrastructures that enable electronic communication between the components. For example, network 50 may include at least one of the Internet, an intranet, a PAN (Personal Area Network), a LAN (Local Area Network), a WAN (Wide Area Network), a SAN (Storage Area Network), a MAN (Metropolitan Area Network), a wireless network, a cellular communications network, a Public Switched Telephone Network, and/or other network. According to various implementations, software development managements system 110 and the various components described herein may be implemented in hardware and/or a combination of hardware and programming that configures hardware. Furthermore, in FIG. 1 and other Figures described herein, different numbers of components or entities than depicted may be used.

[0016] Software development managements system 110 may comprise a methodology rules engine 121, a dynamic score engine 122, a recommended action engine 123, a user interface engine 124, a score report engine 125, and/or other engines. The term "engine", as used herein, refers to a combination of

hardware and programming that performs a designated function. As is illustrated respect to FIGS. 3-4, the hardware of each engine, for example, may include one or both of a processor and a machine-readable storage medium, while the programming is instructions or code stored on the machine-readable storage medium and executable by the processor to perform the designated function.

[0017] Methodology rules engine 121 may identify a set of rules related to a software development methodology. As discussed above, a particular software development methodology sets forth and/or requires a particular set of rules to be followed by a user and/or a team of users. One example of software development methodology includes “waterfall” methodology. The waterfall methodology comprises several phases of development (e.g., requirements, analysis, design, coding, testing, acceptance, etc.) and is designed such that moving to the next phase of development requires the completion of the preceding phase. Development progress flows in a downward fashion; hence the name “waterfall” was conferred onto this model. The development phases, events, activities, actions, tasks, and/or items under each of the phases, and/or other rules may be defined as a set of rules associated with the waterfall methodology.

[0018] Another example of software development methodology includes “agile” methodology. This type of development is iterative and incremental (e.g., an iteration is called a “sprint” in scrum methodology, which is one type of agile methodology). In this methodology, the software is developed in small batches such that changes can be easily be introduced into the software product. In the example of scrum methodology, a product backlog refers to a list of product backlog items (PBIs) that are managed for any changes to be made to a particular software product. The product backlog may be used to capture requests for modifying a product (e.g., adding, replacing, and/or deleting features). Each PBI may be represented in the form of a user story that describes a desired feature in narrative form. In this case, a set of rules associated with the scrum methodology may include the following example rules: “a product backlog is defined before a sprint planning meeting,” “a new user story is not added to an active sprint,” “the number of defects that were opened during the sprint is less than a predefined threshold,” and “user stories are completed within a predefined timeframe.”

[0019] The set of rules related to a particular software development methodology may be identified in several ways: In some implementations, the set of rules may be pre-defined or otherwise provided by the particular software development methodology. The pre-defined set of rules may be obtained by methodology rules engine 121 and/or further defined and/or customized (e.g., a rule added, deleted, and/or modified) by a user. In other implementations, the set of rules may be manually defined by a user via methodology rules engine 121 by adding, deleting, and/or modifying any portion of the set of rules.

[0020] Each rule of the set of rules may be associated with a rule-specific score. The associated score may be given when the rule is invoked (e.g., satisfied or violated) by an action performed by a user (and/or an inaction by the user). A rule-specific score may be numerical (e.g., +5 points, -2 points, etc.), category-based (e.g., satisfied, violated, etc.), and/or of another form. Returning to the example above, the rule requiring that “a product backlog is defined before a sprint planning meeting” may be satisfied by a user’s action of defining the product backlog before the sprint planning meeting, but may be violated if the user fails to define the product backlog before the sprint planning meeting. To encourage users to satisfy the rule, a positive rule-specific score (e.g., +3 points) may be associated with this rule, which may be given if the user satisfies the rule. In some implementations, another score may be associated with the same rule, which may be given if the user violates the rule. For example, a score of zero may be given if the user violates the rule, or in some situations, a negative score (e.g., -3 points) may be given.

[0021] In another example, the rule requiring that “a new user story is not added to an active sprint” may be violated if a user adds a new user story to an active sprint, but may remain satisfied if the user does not perform this action. To discourage users from violating the rule, a negative rule-specific score (e.g., -5 points) may be associated with this rule, which may be given if the user violates the rule. In some implementations, another score may be associated with the same rule, which may be given if the user does not violate the rule. For example, a score of zero may be given if the user does not violate the rule, or in some situations, a positive score (e.g., +1 point) may be given.

[0022] Such rule-specific scores may be pre-defined or otherwise provided by the particular software development methodology, and/or further defined and/or customized (e.g., score modified) by a user via methodology rules engine 121. In some implementations, a rule-specific score for a particular rule may be manually defined by a user via methodology rules engine 121.

[0023] Note that a different set of rules and/or different rule-specific scores may apply to different users (or teams). For example, a first user (or a first team) may be expected to comply with a first set of rules while a second user (or a second team) may be expected to comply with a second set of rules that is different from the first set of rules. In another example, both the first and second users (or the first and second teams) may be expected to comply with the same set of rules, but some rules may be associated with different scores for different users (or teams).

[0024] Dynamic score engine 122 may monitor a software development process. For example, dynamic score engine 122 may monitor actions performed by users (and/or their inactions) during the software development process. In some implementations, the monitoring may continue for a particular time period (e.g., a particular iteration, sprint, etc.). Dynamic score engine 122 may determine, during the monitoring, whether at least one rule of the set of rules (e.g., identified by methodology rules engine 121) is invoked (e.g., satisfied or violated) by an action performed by a user (and/or by an inaction of a user). For example, dynamic score engine 122 may, during the monitoring, detect that a new user story has been added to an active sprint by a user, and thus determine that the rule requiring that “a new user story is not added to an active sprint” has been invoked (e.g., violated) by the user’s action.

[0025] Dynamic score engine 122 may determine and/or dynamically adjust a software development management score based on which rules of the set of rules are invoked. A “software development management score,” as used herein, may indicate a measure of compliance with a set of rules associated with a particular software development methodology. A software development management score may represent a measure of compliance by a user and/or a team of users. In some instances, a software development management score may be determined for the particular time period for which the monitoring took place. In this way, a

software development management score determined for a user (or a team of users) may be easily compared with a software development management score determined for another user (or another team of users). Moreover, a software development management score determined for one iteration or sprint may be compared to a software development management score determined for another iteration or sprint.

[0026] For example, if a first rule of the set of rules is invoked, the rule-specific score associated with the first rule may be added to and/or subtracted from the software development management score. Similarly, if a second rule of the set of rules is invoked, the rule-specific score associated with the second rule may be added to and/or subtracted from the software development management score.

[0027] Recommended action engine 123 may identify a current status of the software development process (e.g., as being monitored by dynamic score engine 122). A “current status of the software development process,” as used herein, may comprise information related to: current progress or rate thereof (e.g., a percentage of software development being completed to date, a percentage of software development completed per a unit of time such as 10% completion per week); current development work distribution, current status of development tasks (e.g., user stories) completed, yet to be completed, delayed, etc., current development team membership, and/or any other information related to the software development process.

[0028] Recommended action engine 123 may use the current status to determine a recommended action to increase the software development management score. For example, the recommended action, if performed by a user, may increase the software development management score. If the current status indicates that “user stories are not assigned equally among the sprint members” (e.g., one of the set of rules may require the user stories to be assigned equally among the sprint members), the recommended action may be generated to suggest the user to move certain user stories to one member to another member.

[0029] In some implementations, recommended action engine 123 may determine which rules of the set of rules are predicted to be violated based on the

current status. Recommended action engine 123 may generate an alert message that notifies the user of the rules that are predicted to be violated and/or provide a recommended action that, if performed by a user, satisfies at least one of the rules that are predicted to be violated. For example, this determination (of which rules of the set of rules are predicted to be violated) may be based on projecting the current progress rate forward. Consider the following example: a particular rule of the set of rules is defined that 100% content should be delivered until the end of the 8-week long sprint. Recommended action engine 123 may identify the current status (e.g., at the 4-week mark, 40% of the content is completed) and/or the current progress rate (e.g., 10% completion per week). Recommended action engine 123 may then determine that the particular rule is predicted to be violated (e.g., meaning that 100% content cannot be delivered until the end of the sprint) given the current status (e.g., given that the current progress rate is projected forward). If the current progress rate is projected forward, another 40% of the content can be completed over the next 4 weeks, resulting in 80% completion rather than 100% completion that was originally intended by the particular rule. Note that the rule that is predicted to be violated may be satisfied by performance of a recommended action.

[0030] In some implementations, the recommended action may be provided in narrative form. For example, it may state “user stories are not assigned equally among the sprint members. You can increase your score by 5 points by moving 2 user stories from Daniel to Jack.” Note that this may include the recommended action (e.g., moving 2 user stories from Daniel to Jack), a current status and/or a rule to be satisfied by the recommended action (e.g., user stories are not assigned equally among the sprint members), and a rule-specific score associated with the rule to be satisfied by the recommended action (e.g., 5 points).

[0031] A recommended action may be tailored to a particular user (e.g., different users may have a different set of recommended actions, the same recommended action in different narrative form, etc.) and/or to a particular development team.

[0032] User interface engine 124 may present, via a user interface, the software development management score, a recommended action, a current

status and/or a rule to be satisfied by the recommended action, a rule-specific score associated with the rule to be satisfied by the recommended action (e.g., an increase in the software development management score if the recommended action is performed by the user), etc. In some examples, if a cursor pointer hovers over a graphical user element representing the software development management score, user interface engine 124 may cause a display of the recommended action (e.g., moving 2 user stories from Daniel to Jack), a current status and/or a rule to be satisfied by the recommended action (e.g., user stories are not assigned equally among the sprint members), and/or a rule-specific score associated with the rule to be satisfied by the recommended action (e.g., 5 points).

[0033] In this way, if a user (e.g., development team lead) is dissatisfied with the current software development management score, the user may easily receive several recommended actions to increase the level of compliance with the set of rules imposed by the given software development methodology.

[0034] Score report engine 125 may generate a score report that includes a detailed breakdown of the software development management score. The report may include rules that have been invoked, rule-specific scores associated with those invoked rules, any statistics related to the software development process (e.g., overall development progress rate), recommended actions by recommended action 123, recommended actions that have been performed by a user, and/or other information related to the software development process and/or software development methodology.

[0035] In performing their respective functions, engines 121-125 may access data storage 129 and/or other suitable database(s). Data storage 129 may represent any memory accessible to software development managements system 110 that can be used to store and retrieve data. Data storage 129 and/or other database may comprise random access memory (RAM), read-only memory (ROM), electrically-erasable programmable read-only memory (EEPROM), cache memory, floppy disks, hard disks, optical disks, tapes, solid state drives, flash drives, portable compact disks, and/or other storage media for storing computer-executable instructions and/or data. Software development managements system

110 may access data storage 129 locally or remotely via network 50 or other networks.

[0036] Data storage 129 may include a database to organize and store data. The database may reside in a single or multiple physical device(s) and in a single or multiple physical location(s). The database may store a plurality of types of data and/or files and associated data or file description, administrative information, or any other data.

[0037] FIG. 2 is a block diagram depicting an example software development managements system 210. Software development managements system 210 may comprise a methodology rules engine 221, a dynamic score engine 222, a recommended action engine 223, and/or other engines. Engines 221-223 represent engines 121-123, respectively.

[0038] FIG. 3 is a block diagram depicting an example machine-readable storage medium 310 comprising instructions executable by a processor for software development managements.

[0039] In the foregoing discussion, engines 121-125 were described as combinations of hardware and programming. Engines 121-125 may be implemented in a number of fashions. Referring to FIG. 3, the programming may be processor executable instructions 321 stored on a machine-readable storage medium 310 and the hardware may include a processor 311 for executing those instructions. Thus, machine-readable storage medium 310 can be said to store program instructions or code that when executed by processor 311 implements software development managements system 110 of FIG. 1.

[0040] In FIG. 3, the executable program instructions in machine-readable storage medium 310 are depicted as recommended action instructions 321. Instructions 321 represent program instructions that, when executed, cause processor 311 to implement engines 123, respectively.

[0041] FIG. 4 is a block diagram depicting an example machine-readable storage medium 410 comprising instructions executable by a processor for software development managements.

[0042] Referring to FIG. 4, the programming may be processor executable instructions 421-425 stored on a machine-readable storage medium 410 and the

hardware may include a processor 411 for executing those instructions. Thus, machine-readable storage medium 410 can be said to store program instructions or code that when executed by processor 411 implements software development managements system 110 of FIG. 1.

[0043] In FIG. 4, the executable program instructions in machine-readable storage medium 410 are depicted as methodology rules instructions 421, dynamic score instructions 422, recommended action instructions 423, user interface instructions 424, and score report instructions 425. Instructions 421-425 represent program instructions that, when executed, cause processor 411 to implement engines 121-125, respectively.

[0044] Machine-readable storage medium 310 (or machine-readable storage medium 410) may be any electronic, magnetic, optical, or other physical storage device that contains or stores executable instructions. In some implementations, machine-readable storage medium 310 (or machine-readable storage medium 410) may be a non-transitory storage medium, where the term “non-transitory” does not encompass transitory propagating signals. Machine-readable storage medium 310 (or machine-readable storage medium 410) may be implemented in a single device or distributed across devices. Likewise, processor 311 (or processor 411) may represent any number of processors capable of executing instructions stored by machine-readable storage medium 310 (or machine-readable storage medium 410). Processor 311 (or processor 411) may be integrated in a single device or distributed across devices. Further, machine-readable storage medium 310 (or machine-readable storage medium 410) may be fully or partially integrated in the same device as processor 311 (or processor 411), or it may be separate but accessible to that device and processor 311 (or processor 411).

[0045] In one example, the program instructions may be part of an installation package that when installed can be executed by processor 311 (or processor 411) to implement software development managements system 110. In this case, machine-readable storage medium 310 (or machine-readable storage medium 410) may be a portable medium such as a floppy disk, CD, DVD, or flash drive or a memory maintained by a server from which the installation package can be

downloaded and installed. In another example, the program instructions may be part of an application or applications already installed. Here, machine-readable storage medium 310 (or machine-readable storage medium 410) may include a hard disk, optical disk, tapes, solid state drives, RAM, ROM, EEPROM, or the like.

[0046] Processor 311 may be at least one central processing unit (CPU), microprocessor, and/or other hardware device suitable for retrieval and execution of instructions stored in machine-readable storage medium 310. Processor 311 may fetch, decode, and execute program instructions 321, and/or other instructions. As an alternative or in addition to retrieving and executing instructions, processor 311 may include at least one electronic circuit comprising a number of electronic components for performing the functionality of at least one of instructions 321, and/or other instructions.

[0047] Processor 411 may be at least one central processing unit (CPU), microprocessor, and/or other hardware device suitable for retrieval and execution of instructions stored in machine-readable storage medium 410. Processor 411 may fetch, decode, and execute program instructions 421-425, and/or other instructions. As an alternative or in addition to retrieving and executing instructions, processor 411 may include at least one electronic circuit comprising a number of electronic components for performing the functionality of at least one of instructions 421-425, and/or other instructions.

[0048] FIG. 5 is a flow diagram depicting an example method 500 for software development managements. The various processing blocks and/or data flows depicted in FIG. 5 (and in the other drawing figures such as FIG. 6) are described in greater detail herein. The described processing blocks may be accomplished using some or all of the system components described in detail above and, in some implementations, various processing blocks may be performed in different sequences and various processing blocks may be omitted. Additional processing blocks may be performed along with some or all of the processing blocks shown in the depicted flow diagrams. Some processing blocks may be performed simultaneously. Accordingly, method 500 as illustrated (and described in greater detail below) is meant to be an example and, as such, should not be viewed as limiting. Method 500 may be implemented in the form of executable instructions

stored on a machine-readable storage medium, such as storage medium 310, and/or in the form of electronic circuitry.

[0049] In block 521, method 500 may include identifying a set of rules related to a software development methodology. Each rule of the set of rules may be associated with a rule-specific score. Referring back to FIG. 1, methodology rules engine 121 may be responsible for implementing block 521.

[0050] In block 522, method 500 may include monitoring a software development process. Referring back to FIG. 1, dynamic score engine 122 may be responsible for implementing block 522.

[0051] In block 523, method 500 may include determining, during the monitoring, whether at least one rule of the set of rules is invoked by an action performed by a user. In response to determining that a first rule of the set of rules is affected, method 500 may proceed to block 524. Otherwise, method 500 may return to block 522. Referring back to FIG. 1, dynamic score engine 122 may be responsible for implementing block 523.

[0052] In block 524, method 500 may include dynamically adjusting a software development management score based on the rule-specific score associated with the first rule. Referring back to FIG. 1, dynamic score engine 122 may be responsible for implementing block 524.

[0053] In block 525, method 500 may include providing a recommended action to increase the software development management score. Referring back to FIG. 1, recommended action engine 123 may be responsible for implementing block 525.

[0054] FIG. 6 is a flow diagram depicting an example method 600 for software development managements. Method 600 as illustrated (and described in greater detail below) is meant to be an example and, as such, should not be viewed as limiting. Method 600 may be implemented in the form of executable instructions stored on a machine-readable storage medium, such as storage medium 310, and/or in the form of electronic circuitry.

[0055] In block 621, method 600 may include identifying a set of rules related to a software development methodology. Each rule of the set of rules may be

associated with a rule-specific score. Referring back to FIG. 1, methodology rules engine 121 may be responsible for implementing block 621.

[0056] In block 622, method 600 may include identifying a current status of a software development process. Referring back to FIG. 1, recommended action engine 123 may be responsible for implementing block 622.

[0057] In block 623, method 600 may include determining which rules of the set of rules are predicted to be violated based on the current status. Referring back to FIG. 1, recommended action engine 123 may be responsible for implementing block 623.

[0058] In block 624, method 600 may include providing a recommended action that, when performed by a user, satisfies at least one of the rules that are predicted to be violated. Referring back to FIG. 1, recommended action engine 123 may be responsible for implementing block 624.

[0059] The foregoing disclosure describes a number of example implementations for software development managements. The disclosed examples may include systems, devices, computer-readable storage media, and methods for software development managements. For purposes of explanation, certain examples are described with reference to the components illustrated in FIGS. 1-4. The functionality of the illustrated components may overlap, however, and may be present in a fewer or greater number of elements and components.

[0060] Further, all or part of the functionality of illustrated elements may co-exist or be distributed among several geographically dispersed locations. Moreover, the disclosed examples may be implemented in various environments and are not limited to the illustrated examples. Further, the sequence of operations described in connection with FIGS. 5-6 are examples and are not intended to be limiting. Additional or fewer operations or combinations of operations may be used or may vary without departing from the scope of the disclosed examples. Furthermore, implementations consistent with the disclosed examples need not perform the sequence of operations in any particular order. Thus, the present disclosure merely sets forth possible examples of implementations, and many variations and modifications may be made to the

described examples. All such modifications and variations are intended to be included within the scope of this disclosure and protected by the following claims.

CLAIMS

1. A method for software development managements, the method comprising:
 - identifying a set of rules related to a software development methodology, each rule of the set of rules associated with a rule-specific score;
 - monitoring a software development process;
 - determining, during the monitoring, whether at least one rule of the set of rules is invoked by an action performed by a user;
 - in response to determining that a first rule of the set of rules is invoked, dynamically adjusting a software development management score based on the rule-specific score associated with the first rule; and
 - providing a recommended action to increase the software development management score.
2. The method of claim 1, further comprising:
 - in response to determining that a second rule of the set of rules is invoked, dynamically adjusting the software development management score based on the rule-specific score associated with the second rule.
3. The method of claim 1, further comprising:
 - identifying a current status of the software development process; and
 - determining which rules of the set of rules are predicted to be violated based on the current status.
4. The method of claim 3, further comprising:
 - generating an alert message that notifies the user of the rules that are predicted to be violated.
5. The method of claim 3, further comprising:
 - providing the recommended action that, if performed by the user, satisfies at least one of the rules that are predicted to be violated.

6. The method of claim 1, wherein the software development methodology comprises: agile software development, waterfall, lean development, crystal method, dynamic systems development model (DSDM), extreme programming (XP), or feature driven development (FDD).

7. A non-transitory machine-readable storage medium comprising instructions executable by a processor of a computing device for software development managements, the machine-readable storage medium comprising:

instructions to identify a current status of a software development process that is managed using a software development methodology, the software development methodology having a set of rules, the current status including information related to a current progress rate;

instructions to determine which rules of the set of rules are predicted to be violated, the determination being based on projecting the current progress rate forward; and

instructions to generate a recommended action that, if performed by a first user of a first software development team, avoids violation of at least one of the rules that are predicted to be violated.

8. The non-transitory machine-readable storage medium of claim 7, further comprising:

instructions to monitor the software development process for a first time period;

instructions to determine, during the monitoring, whether a first rule of the set of rules is invoked by an action performed by a second user of the first software development team, the first rule associated with a first rule-specific score;

in response to determining that the first rule is invoked, dynamically adjusting a software development management score based on the first rule-specific score; and

instructions to provide the recommended action to increase the software development management score.

9. The non-transitory machine-readable storage medium of claim 8, further comprising:

instructions to provide a comparison between the software development management score associated with the first software management team and a software development management score associated with a second software management team.

10. The non-transitory machine-readable storage medium of claim 8, further comprising:

instructions to provide a comparison between the software development score determined for the first time period and a software development score determined for a second time period.

11. The non-transitory machine-readable storage medium of claim 7, wherein the set of rules comprises at least one of: a first set of positive rules and a second set of negative rules, each positive rule of the first set associated with a positive score and each negative rule of the second set associated with a negative score.

12. A system for software development managements comprising:
a processor that:

identifies a set of rules related to a software development methodology,
each rule of the set of rules associated with a rule-specific score;

monitors a software development process;

determines, during the monitoring, whether at least one rule of the set of rules is invoked by an action performed by a user;

in response to determining that a first rule of the set of rules is invoked, dynamically adjusts a software development management score based on the rule-specific score associated with the first rule; and

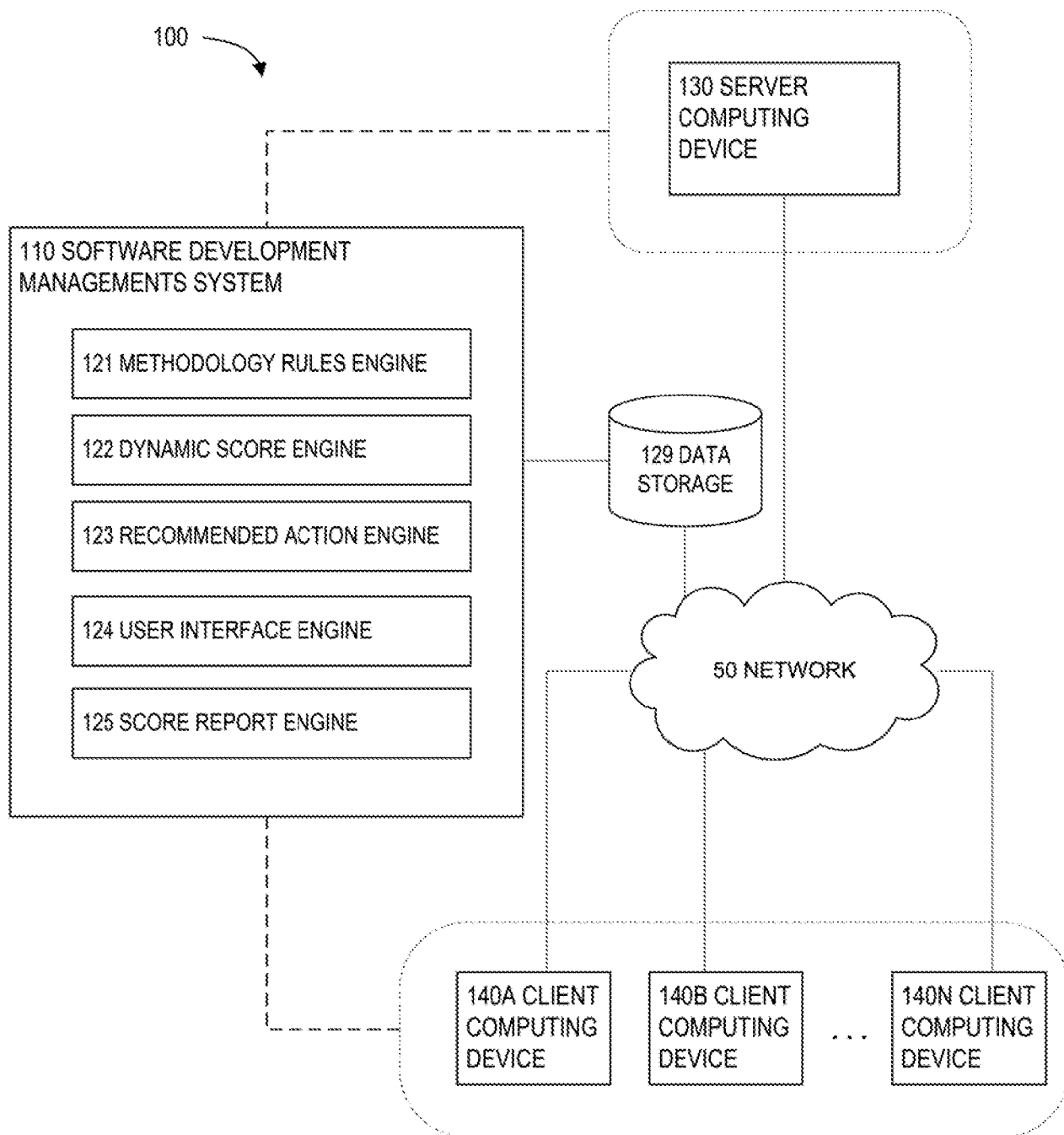
identifies a current status of the software development process; and

determines a recommended action based on the current status, wherein the recommended action, if performed by the user, increases the software development management score.

13. The system of claim 12, the processor that:
presents, via a user interface, the software development management score, the recommended action, and an increase in the software development management score if the recommended action is performed by the user.

14. The system of claim 12, the processor that:
in response to determining that a second rule of the set of rules is invoked, dynamically adjusts the software development management score based on the rule-specific score associated with the second rule; and
generates a score report that includes the first rule, the rule-specific score associated with the first rule, the second rule, and the rule-specific score associated with the second rule.

15. The system of claim 12, the processor that:
monitors the software development process for a time period, wherein the time period comprises an iteration or a sprint.

**FIG. 1**

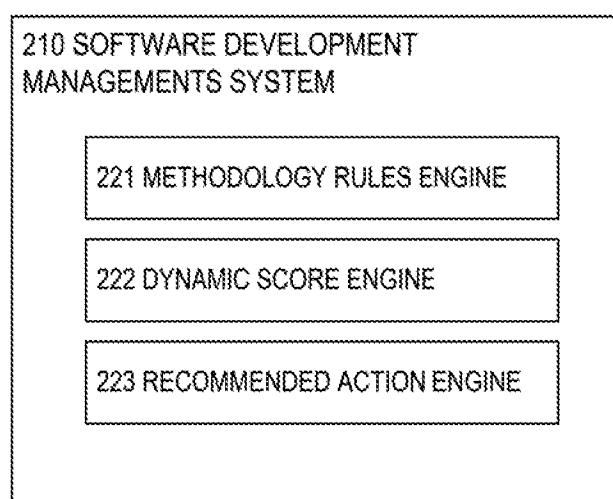
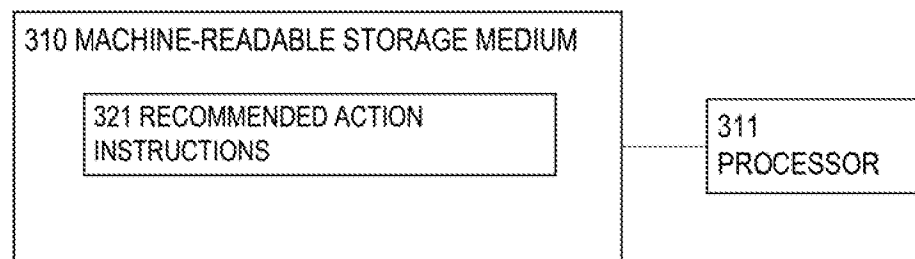
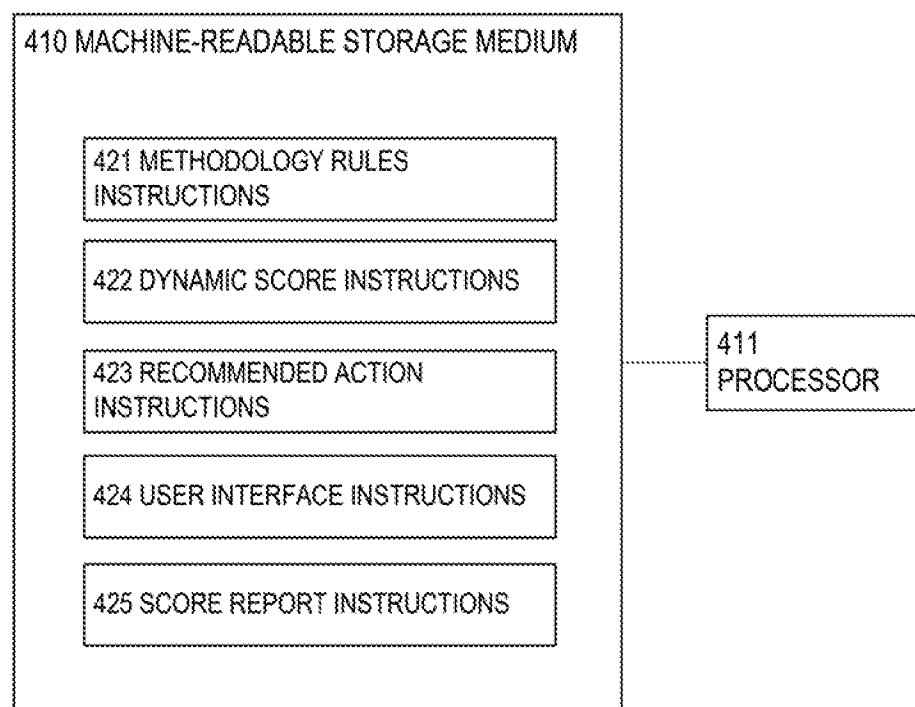
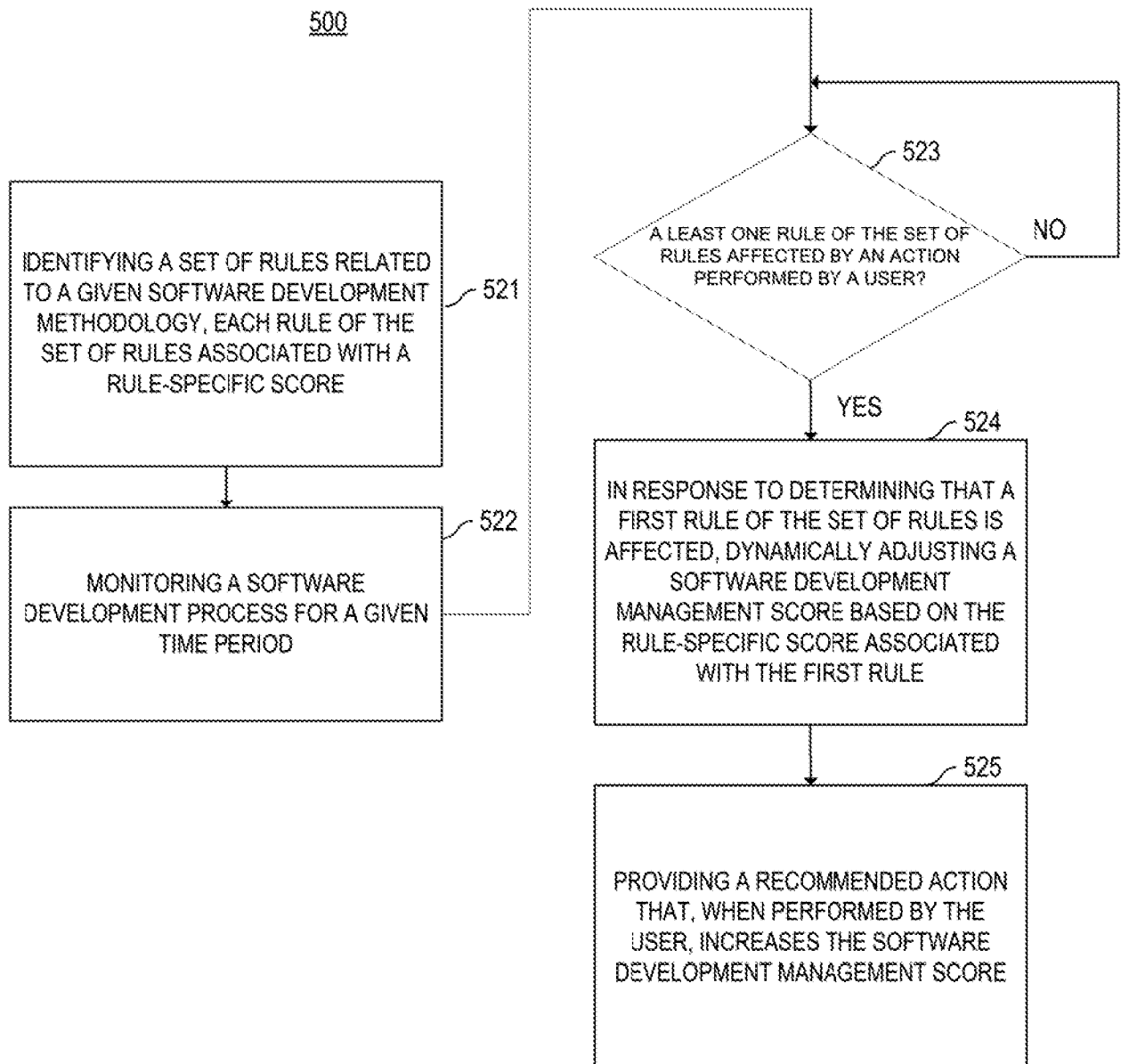
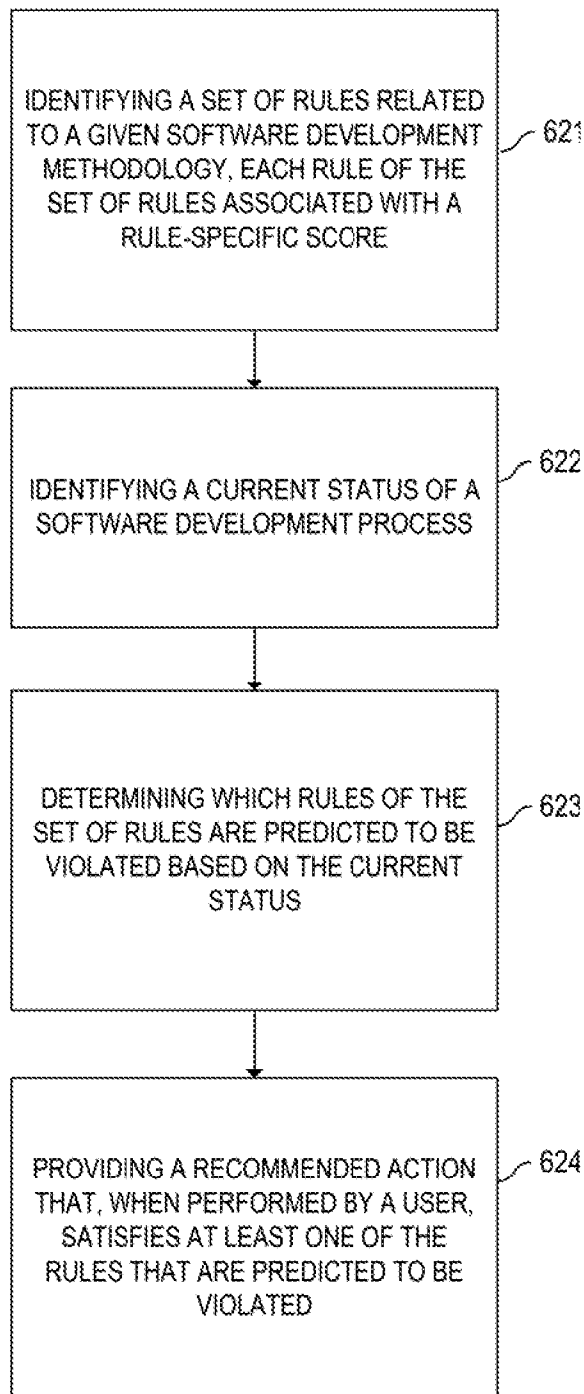


FIG. 2

**FIG. 3****FIG. 4**

**FIG. 5**

5/5

600**FIG. 6**

A. CLASSIFICATION OF SUBJECT MATTER**G06F 9/44(2006.01)i, G06F 11/30(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 9/44; G06N 5/00; G06F 11/30

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 development management, rule, score, monitoring, recommend, violation, alert message, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2011-0314450 A1 (MORAN SHOCHAT et al.) 22 December 2011 See paragraphs [0016], [0023], [0036]–[0038]; claims 1–6, 8–15, 20; and figure 3.	1–5, 7–8, 11–12, 15
Y		6, 9–10, 13–14
Y	ABHISHEK BORAD et al., "Software Engineering – Agile Methodology using SCRUM", IJIRST – International Journal for Innovative Research in Science & Technology, May 2015, Vol. 1 (http://www.ijirst.org/articles/IJIRSTV1I12106.pdf) See pages 175, 185–186.	6, 9–10, 13–14
A	US 2012-0150782 A1 (GREG BENSON et al.) 14 June 2012 See paragraphs [0041]–[0059]; claim 1; and figure 2.	1–15
A	US 2011-0041120 A1 (NACHIAPPAN NAGAPPAN et al.) 17 February 2011 See paragraphs [0057]–[0060]; claim 1; and figure 6.	1–15
A	US 2009-0070734 A1 (MARK DIXON et al.) 12 March 2009 See paragraphs [0056]–[0062]; claim 1; and figure 4.	1–15



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

"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

17 August 2016 (17.08.2016)

Date of mailing of the international search report

18 August 2016 (18.08.2016)

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

LEE, EUN KYU

Telephone No. +82-42-481-3580



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/064667

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0314450 A1	22/12/2011	US 8589882 B2	19/11/2013
US 2012-0150782 A1	14/06/2012	US 2006-0291657 A1	28/12/2006
		US 2007-0002139 A1	04/01/2007
		US 2007-0002140 A1	04/01/2007
		US 2007-0008410 A1	11/01/2007
		US 2007-0011105 A1	11/01/2007
		US 2007-0011106 A1	11/01/2007
		US 2007-0011107 A1	11/01/2007
		US 2007-0011108 A1	11/01/2007
		US 2007-0022057 A1	25/01/2007
		US 2007-0022079 A1	25/01/2007
		US 2007-0030143 A1	08/02/2007
		US 2007-0182544 A1	09/08/2007
		US 2009-0210378 A1	20/08/2009
		US 2010-0090822 A1	15/04/2010
		US 2013-0120135 A1	16/05/2013
		US 2014-0058990 A1	27/02/2014
		US 7512583 B2	31/03/2009
		US 7526455 B2	28/04/2009
		US 7609159 B2	27/10/2009
		US 7656286 B2	02/02/2010
		US 8515895 B2	20/08/2013
		US 8830053 B2	09/09/2014
		WO 2006-119323 A2	09/11/2006
		WO 2006-119323 A3	22/02/2007
		WO 2007-027239 A1	08/03/2007
		WO 2007-027239 A8	26/04/2007
US 2011-0041120 A1	17/02/2011	None	
US 2009-0070734 A1	12/03/2009	WO 2007-041242 A2	12/04/2007