(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 December 2016 (15.12.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/200413 A1

(51) International Patent Classification:
G06F 11/36 (2006.01) *G06F 11/30* (2006.01)

(21) International Application Number:
PCT/US2015/050970

(22) International Filing Date:
18 September 2015 (18.09.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/172,598 8 June 2015 (08.06.2015) US

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **EGOZI LEVI, Efrat**; Altalef st. No. 5, Shabatzi st. No. 26, 56100 Yehud (IL). **ASSULIN, Ohad**; Altalef st. No. 5, Shabatzi st. No. 26, 56100 Yehud (IL).

(74) Agents: **WOODS, Ariana** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 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,

[Continued on next page]

(54) Title: APPLICATION SESSION ANALYSIS AND RECOMMENDATION SYSTEM

(57) Abstract: An application session analysis and recommendation system (100) finds similar first sessions (29,30,31) of recorded events for each session of a set of second sessions (39,40,41) of recorded events to create a set of respective matched sessions (32,33). What is different and similar with respect to the recorded events between each second session and the set of respective matched sessions is analyzed (204) and overall similarity and usage scores are produced (206) along with a real user coverage score for each second session.

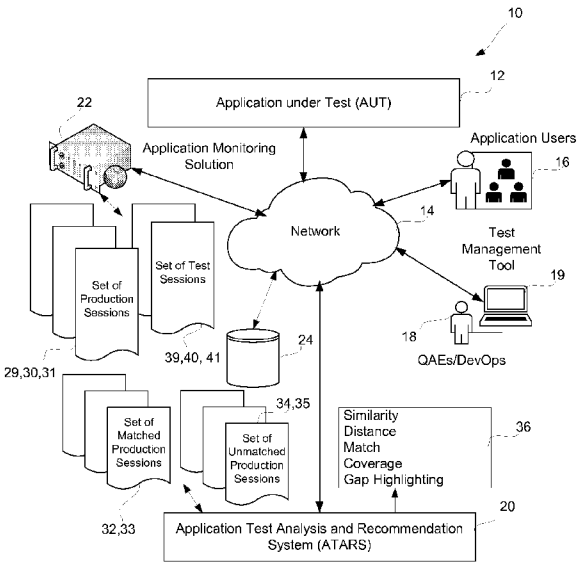

Fig. 1

WO 2016/200413 A1

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))*

APPLICATION SESSION ANALYSIS AND RECOMMENDATION SYSTEM

RELATED APPLICATIONS

[0001] This application claims the benefit of priority to U.S. Provisional Application Serial No. 62/172, 598, filed June 8, 2015, entitled “DevOps and Test Recommendation with User Coverage and Usage”, which is herein incorporated by reference in its entirety.

BACKGROUND

[0002] Applications and services are monitored with various software testing tools with unpredictable degrees of success. Software testing is the process of evaluating a software application, whether a stand-alone application, a web-based application, or software-as-a-service, to detect differences between given input and expected output and/or to assess various feature of the software application. Software testing is a process that should be done during the development process but often times that development testing is difficult and time consuming for quality assurance engineers. To ensure ongoing quality, QAEs often continue testing after an application is released to production. Accordingly, testing is used to assess the quality of the software application, both before release and after release to production. In other words software testing is a generally both a verification and a validation process.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] The disclosure is better understood with reference to the following drawings. The elements of the drawings are not necessarily to scale relative to each other. Rather, emphasis has instead been placed upon clearly illustrating the claimed subject matter. Furthermore, like reference numerals designate corresponding similar parts through the several views.

[0004] Fig. 1 is an example environment diagram for an Application Session Analysis and Recommendation System (ASARS);

[0005] Fig. 2 is an example overall flow diagram for an ASARS;;

[0006] Fig. 3 is a block diagram of a computer based system for implementing an ASARS;

[0007] Fig. 4 is an example flow chart showing only a minimal set of modules to implement an ASARS;

[0008] Fig. 5 is an example non-transitory computer readable medium (CRM) storing instructions to implement an ASARS;

[0009] Fig. 6 is an example block diagram that illustrates how a similarity and match module may be implemented;

[0010] Fig. 7 is an example flow chart of one particular possible implementation of a similarity and match module;

[0011] Fig. 8 is an example simplified flow chart of a gap discovery module;

[0012] Fig. 9 is a flow chart of an exemplary implementation of an alignment technique to create a “super-flow” for a set of similar or matched sessions;

[0013] Fig. 10 is an example flow chart for aligning a super-flow with a respective test session to find the shortest alignment between the two;

[0014] Fig. 11 is an example flow chart of a gap discovery method for unmatched sessions;

[0015] Fig. 12 is an example flow chart of a method for gap discovery for uncovered tuples; and

[0016] Fig. 13 is an example implementation with visualization of the results of an ASARS.

DETAILED DESCRIPTION

Description of the Problem

[0017] The purpose of software application testing and monitoring solutions is to provide an application's owner, software developer operations (DevOps), and quality assurance engineering (QAE) team with a better understanding of customer (user) usage patterns. However, developing and testing an application that "best fits" a user's usage pattern requires both understanding of what the user's usage patterns are and how to embed this understanding into a session of application test flows. Classic application analysis tools only provide an event level based analysis. Contrarily, this disclosure describes a software "Application Session Analysis and Recommendation System" (ASARS) (20, Fig. 1) that improves application testing of the application on a "session level" based analysis, which previously such session level testing/analysis was confusing and hard to use. A session level analysis uses a sequence of events (which may include other actions) or a combination of events). Classic application analysis tools also only collected data from production tests and lacked the ability for comparison and analysis of the tests to the production data as this disclosure will describe and make known.

Description of the Environment

[0018] Fig. 1 is an example environment diagram showing how the Application Session Analysis and Recommendation System (ASARS) 20 may be incorporated into a development environment 10. An application under test (AUT) 12 is connected to a network 14 or other communication service, for instance the Internet or a corporate intranet, a local area network, a virtual private network, a virtual software network, or the like. The network 14 provides a communication vehicle for the AUT 10 to communicate with one or more application monitoring solution 22, such as "Google Analytics"™, "Dynatrace"™, HEWLETT-PACKARD™'s "RUM"™, and "AppPulse"™ as just a few examples. Application users 16 (also referred to herein at times as customers) typically interact with the AUT 12 via a client device such as a terminal, personal computer, workstation, laptop and notebook computers, tablet computers, Smart or feature cell-phones, or other handheld devices as just some examples. In other examples, the Application user may also be connected to

the AUT 12 directly, such as if it were an application running on their computing device locally.

[0019] The application monitoring solution 22 is set up by the Application owner, DevOPs, and/or a QA department to capture sessions of an application user's 16 use of the AUT 12 to create a first set 29 of one or more session events, such as set of production sessions 30. In another example, the first set 29 of sessions may be a first set 29 of one or more session events from application monitoring solution 22, such an "A" set. The first set 29 of sessions may be any set of sessions used as a basis or 'base' set 31 for comparison.

[0020] The application monitoring solution 22 can also be set up similarly to capture a second set 39 of one or more reference session events such as a set of test session 40 which are typically developed by the DevOps or QA Engineers (QAE) 18 using a test management tool 19. In another example, the second set 39 of one or more sessions may also be another set of session events from application monitoring solution 22, such as a "B" set. The second set of sessions 39 can be any set of sessions used as a reference set 41 to compare against the 'base' set 31 of sessions. The test management tool 19 may be server or client based and can be a standalone application or part of the application monitoring solution 22 or integrated as part of other application testing tools. The first 29 and second set 39 of session events can be stored locally on a server running the application monitoring solution 22, remotely on a separate storage system 24 connected to the network, remotely on the QAE/DevOps 18 workstation, or server, the test management tool 19, and/or a server hosting an ASARS 20. In some examples, the ASARS 20 may be done either on the AUT 12 or on a SaaS (software as a service) service that receives the original/unprocessed session's data collected by the App Monitoring Tool.

Definitions

[0021] To help ease understanding, clarity, and conciseness, the following definitions will be used for discussion of additional details of the ASARS 20 unless context requires otherwise or the term is defined explicitly differently. The following definitions are not meant to be a complete definition of the respective terms but rather is intended to help the reader in understanding the following discussion.

[0022] Production Environment – Generally a time in the application product life cycle where the application is released for user testing and use, such as alpha,

beta, and release to manufacturing stages as well as fully release applications. For this disclosure, the production environment also includes internal use testing, ongoing Q/A, application monitoring, and A/B testing stages of the product life cycle.

[0023] Events – A set of events are actions or occurrences detected by an application and monitored by an application monitoring solution 22. Events can be user actions, such as clicking a mouse button or pressing a key, or system occurrences, such as running out of memory. A sequence of events may be a single event such as a Graphical User Interface (GUI) command in the application or a combination of GUI events, sometimes referred to as a "tuple." However, other user events or actions than GUI events may be captured as well, such as text input, delays or pauses, eye tracking, etc.

[0024] Tuple – A finite ordered list of elements in a sequence. In this disclosure, a tuple is an event, a set of consecutive events, or several consecutive events that appear consecutively in collected sequence of events, such as a session based on the session alphabet. The maximum length of the tuple can be modified but one particular example is a tuple of three events.

[0025] Sessions – A session is a period of activity of a user is interacting with an application. For this application, a "session" is a sequence of events as monitored and collected by an application monitoring solution 22, of which many are known to those skilled in the art.

[0026] Session Alphabet – An encoding of events and tuples in a session into a numeric or alphanumeric set of text symbols. Generally, the alphabet is created from encoding event output from the application monitoring solution 22 and typically the same application monitoring solution 22 is used for both the test and production sessions. However, different application monitoring solutions 22 can be used and transcoded as needed via lookup tables, encoding tables, encoding cyphers, etc. as needed into a common textual alphabet for determining similarity and matching.

[0027] For example, a first sequence of events (seq1) may include the following events:

1. Login
2. Open chat
3. Attach
4. Write message
5. Write message

6. Write message
7. Attach
8. Attach
9. Write message
10. Send

[0028] This can also be transposed into a comma separated list of events such as {Login, Open chat, Attach, Write message, Write message, Write message, Attach, Write message, Send}. This list can then be recoded into a series or string of unique numeric (or alphanumeric) representations or alphabet such as {1, 2, 3, 4, 4, 4, 3, 3, 4, 5}. A second sequence of four events (seq2) could be for example:

1. Login
2. Open chat
3. Write message
4. Send

Which could be represented as {1, 2, 4, 5} using the same alphabet as for the first sequence.

[0029] Location - The location of the coded event within the sequence or string of events is used to help determine distance scores. For instance for the first and second strings:

[0030] $\frac{Seq1}{location}\frac{\{1,2,3,4,4,4,3,3,4,5\}}{\{1,2,3,4,5,6,7,8,9,10\}}$ and $\frac{Seq2}{location}\frac{\{1,2,4,5\}}{\{\ 1,2,3,4\}}$

[0031] Preprocessing - Preprocessing of the sequences seq1 with respect to seq2 may be done to use one string for the basis for similarity of the other. In preprocessing, the sequence that is not the base is processed so that events that do not appear in the base sequence are replaced. Assume seq2 is the base, then seq1 is preprocessed to collapse all events that do not appear in seq2 as follows:

Each event that appears in seq1 but not in seq2 is transformed to a “null” wildcard (also known simply as a wildcard or joker) event, e.g. “#”. The null event is an event that is not part of the alphabet of events and doesn’t appear in any sequence including seq2. Consecutive null events are reduced into a single null event.

This preprocessing reduces seq1 into the essential that are used to determine similarity while maintaining the original order. In this example of sequences, seq1 is transformed as follows:

$$Seq1: \{12\#444\#\#45\} - replace\ events\ not\ in\ seq2\ with\ "\#'$$
$$Seq1: \{12\#444\#45\} - collapse\ consequtive\ "\#'s"$$

More examples and detail is discussed in the description of Figs. 6 and 7.

[0032] Recorded sessions – A transcript of a user or test session, preferably in the alphabet used in the similarity and matching module, but at least translatable into that alphabet.

[0033] Production session – User sessions recorded from a user's usage of an application in a production environment during a session. However, a production session in some examples may be a test session or other created session for comparison to the test session. For this disclosure, the production session is the "base" session or the session that is the "basis" for comparison with the test session.

[0034] Test session – Application sessions that are typically created by QAEs and DevOps to test and debug applications or to simulate user usage of the application. For this disclosure, the test session is a 'reference' session that is compared to the "base" or production session for comparison and therefore the result is "how similar the test session is compared to the production session." In some examples, the test and production sessions may be swapped to provide "how similar the production session is compared to the test session."

[0035] Base session – See Production Session, a base session is the basis that the test session is being compared against. In some examples, the base session is a 'production' session and in other examples, the base session is an 'A' session. In yet another example the 'base' session is a 'test' session. The similarity calculation is asymmetrical since it is calculated with respect to the Base session.

[0036] Reference session – See Test Session, a reference session is the session being compared to the Base session. In some examples, the reference session is a 'test' session and in other examples, the reference is a 'B' session. In yet another example, the reference session may be a 'production' session.

[0037] Real user coverage – A measure of the similarity and distance scores between each pair of matched sessions to compute how well the test session has

coverage over the respective production session monitored and recorded from a real customer user. There may be one or more real user coverage scores generated and/or presented.

[0038] Potential coverage – A measure of the delta between the real user coverage score and the best possible coverage used to help determine potential improvement of the test sessions.

[0039] Gap Discovery – A technique for determining gaps between a test session and the set of matched production sessions. The gap may be an event or set of events that appear (or are missing) in a particular test session sequence but are missing (or appear) in the sequence of events in the matched production sessions. Alternatively, the gap may be an event or set of events that appear (or are missing in a particular production session but missing (or appear) in the sequence of events in the matched test session.

[0040] Alignment – Refers to the relative and absolute location of events in a sequence based on position of the event within the sequence. The alignment of one sequence in reference to another sequence provides each event from each sequence a location in a super-flow aligned sequence.

[0041] Super-flow – For this disclosure, a "super-flow" is an alignment of similar base or test sessions that typically aligns all the similar base or test sessions, respectively, into one session which results in an aligning and merging of 'n' similar base or test sessions. In some examples, a dual direction Levenshtein Distance alignment process is used and the resultant super-flow chosen is that direction which requires the smallest number of modifications as well as optimizes the relative location similarity.

[0042] Levenshtein Distance– is one method for computing the edit distance between two strings. However, it may require a search of possible alignment starting point or initialization and therefore may not be optimal in some instances when there are repetitions and multiple starting alignment options in a sequence. The Levenshtein Distance is sensitive to length of the sessions under comparison as well as repetitions and multiple starting alignment positions.

Challenges

[0043] The ASARS 20 provides an automated solution for analyzing test sessions (also referred to as flows) in comparison to sets of user production

sessions/flows such as for creating a “best fit” with customer usage patterns. However, there are several challenges to making an application a “best fit” to customer usage patterns which the ASARS 20 addresses.

[0044] A first challenge is to create a useful collection of user sessions that are a sequence of events in a production environment by requiring the capturing and recording into a set of production sessions 30 such as a first set 29 of sessions. Also, the application tests used to exercise the application require capturing and recording as well into a set of test sessions 40, such as a second set 39 of sessions. Principally, the production 30 and test 40 sessions are a sequence of events monitored and collected by the application monitoring solution 22. A sequence of events may include single events such as a Graphical User Interface (GUI) command in the AUT 12 or a combination of GUI events, sometimes referred to as a “tuple.” However, other user events or actions than GUI events may be captured as well, such as text input, time in an event, delays or pauses, eye tracking, etc. In some examples, the same application monitoring solution 22 may also be used for both the recording of the production and test sessions in order to have the events coded into the same “alphabet”. However, in other examples, there may be multiple application monitoring solutions 22 used and their recording of events and actions translated or coded into a common alphabet or event coding scheme.

[0045] A second challenge is to create a comparison between the two sets of sessions, production 30 and test 40 (alternatively either base 31 and reference 41 or first set 29 and second set 39), which requires a definition of matching between a pair of sessions. This comparison of matching can also be used in a more generalized manner such as in the case of A/B testing. A/B testing is the comparison of two applications or different versions of the same application, such as a web page design, to see which performs better, usually in a controlled experiment. A/B testing may often be considered as a set of semi-controlled experiments as the A/B applications are released into production gradually. For instance, what sequence of events or actions are usually carried out by a customer/user in a released product or “production” environment is captured or recorded into information entailing of sets of session event flows. This type of information is helpful in understanding how well a particular set of application tests simulate the actual customer/user sessions (or flows of events) from the production environment. The information is also helpful to be able to improve the application tests in order to

optimize the coverage and similarity of the application test sessions with regard to the captured or recorded customer/user sessions from the production environment. Additionally, the information may be used to help improve an application user's experience by allowing the application's owner to create and deploy an application that is a "best fit" for the user's usage pattern.

[0046] For ease of understanding in this disclosure, the first set 29 of sessions is referred to herein as the set of production sessions 30 but it may also be referred to as a base session 31 such as a different copy of production sessions 30, a sub-set of production sessions 30, a set or sub-set of test sessions 40, or any other set of sessions which is to be compared to the test sessions 40. Likewise, the second set 39 of sessions for ease of understanding will be referred to as the set of test sessions 40 but it also may be referred to as a reference set 41 of sessions such as a copy of one or more of the production sessions 30, a set or sub-set of the test sessions 40, or any other set of sessions which is to be compared to the first set 29 of sessions, the set of production sessions 30. For instance, the comparison of the user production sessions 30 with the test sessions 40 may create a set of matched production sessions 33 (matched base sessions 32) within the set of production sessions 30 to the test sessions 40, and a set of unmatched production sessions 35 (unmatched base sessions 34) to the test sessions 40.

[0047] Lastly, a remaining challenge is to analyze the matched and unmatched sessions in order to discover and highlight gaps of events between the two sets of sessions, the production sessions 30 and the test sessions 40. This gap discovery provides the application owner with several important pieces of information 36 in which to make informed and actionable decisions. For instance:

1. A coverage score for the tests under investigation in the second set of sessions captures how well one or more tests simulate their respective similar production sessions. There are other instances as well for the purpose of planning, such as simulate a new session which one may want to deploy and receive its potential usage and coverage based on the above analysis. This analysis may provide the potential impact of introducing this new sequence of events. The gap discovery module may also provide a better understanding of how this new sequence of events deviates from current user behavior and may reflect whether this new sequence is a positive/negative change (e.g.

whether this new sequence makes it shorter to access some app functionality);

2. Highlighted gaps over the matched sessions provide which tests cover the production sessions (or vice versa) and how these test sessions can be modified to increase coverage and similarity to real user sessions;
3. Highlighted gaps over the unmatched sessions provide information on "blind spots" that are not covered in the tests and enable new tests or modified tests to cover these blind spots;
4. Highlighted gaps, in the case of A/B testing, allow for the discovery of real issues in production and for improved "best fit" of the application to actual real user usage; and
5. Easy to understand visualizations of the discovered gaps and coverage scores.

[0048] To summarize, the disclosed technique for ASARS 20 provides information 36 and recommendations at the session level and not at the event level as with other application analysis tools. The technique discloses how to enhance tests based on real user sessions from the application in the production environment and can choose one or more 'best representative' production sessions for each test session based on the match score. The technique can be extended to the field of application monitoring and A/B testing.

Description of the Overall Flow of the System

[0049] Fig. 2 is an example overall flow diagram 50 for the ASARS 20 of Fig. 1 to generate a coverage score for each test 40 or reference 41 session with respect to each production 30 or base 31 session. Each of a first set 29 of event sessions, the set of base session 31 and each of a second set 39 of event sessions, the set of reference sessions 41 are respectively compared in "match" unit 52 to calculate similarity scores, distance (between relative locations of events) scores and match scores for each pairing of base 31 and reference 41 sessions. A "compare" unit 54 produces a set of matched base sessions 32 (or matched production sessions 33) and a set of unmatched base sessions 34 (or unmatched production session 35) or each session in the set of reference sessions 41 from respective similarity, distance, and match scores. Thus, a compare unit 54 is able to find similar base sessions 31 of recorded event for each reference session 41, such as a set of matched base

sessions 32 for a set of reference sessions 41. Compare unit 54 is also able to analyze what is different and similar with respect to the recorded events between each reference session 41 and each of the set of respective matched base sessions 31. For each reference session 41, a “coverage” unit 56 produces an overall similarity and usage score along with a real user coverage score with two flavors using both unique and actual (non-unique) matched base sessions 32 with respect to respective reference sessions 41. Unique sessions are those matched base sessions 32 which may be copies of other base sessions 31 and thus are only counted once. Actual sessions include all the matched base sessions 31. The real usage score calculation uses the similarity score between each pair of sequences and a sensitivity threshold to identify matched pairs of sequences. More detail for one example is presented below with respect to Figs. 6 and 7.

[0050] Besides real user usage coverage, the analysis can continue to analyze gaps in the test flows in comparison to the matched base sessions 32. A first step is to align a particular reference session 41 in comparison to the base sessions that provides the smallest number of misalignments as well as optimizing the similarity of the order of events, thus providing the application owner and/or QAE/DevOps testers an immediate understanding of how well the reference session 41 simulates the base sessions 31 as well as highlight the gaps between the reference session 41 and similar base sessions 31. A second step in the gap analysis may find the best one or more representative session(s) of the set of similar or matched sessions that best covers the matched base set 32 based on the match score. A third step in the gap analysis may provide a coverage score for each reference session 41 under investigation. This coverage score may be used in many other settings, such as A/B testing, to discover unexpected issues at the flow level (small series of recorded events) and not just the session level (which may have a large series of recorded events). Accordingly, the disclosed technique for the ASARS 20 provides an easy to understand coverage score along with gap discovery and highlighting to provide an application owner, QAEs, and/or DevOps with actionable information to develop test and deploy a reliable “best fit” application as well as allow for feature planning when there is a need to understand whether expected user usage patterns are indeed occurring in the real session data and to what degree.

Description of the Overall Method

[0051] Fig. 3 is a block diagram of a computer based system 100 for implementing an ASARS 20. The system 100 includes a processor 102 which may be one or more central processing unit (CPU) cores, hyper threads, or one or more separate CPU units in one or more physical machines. For instance, the CPU may be a multi-core Intel™ or AMD™ processor or it may consist of one or more server implementations, either physical or virtual, operating separately or in one or more datacenters, including the use of cloud computing services. The processor 102 is communicatively coupled with a communication channel 132, such as a processor bus, optical link, etc. to one or more communication devices such as network 104, which may be a physical or virtual network interface, many of which are known to those of skill in the art, including wired and wireless mediums, both optical and radio frequency (RF) for communication. The processor 102 may be also communicatively coupled in some examples to a graphics interface 106 to allow for visualization of the results and recommendations to the ASARS 20 user.

[0052] Processor 102 is also communicatively coupled to non-transient computer readable memory (CRM) 108 which includes a set of instructions organized in modules 110 which when the instruction are read and executed by the processor to cause the processor to perform the functions of the respective modules. While a particular example module organization is shown for understanding, those of skill in the art will recognize that the software may be organized in any particular order or combinations that implements the described functions and still meet the intended scope of the claims. The CRM 108 may include a storage area for holding programs and/or data and may also be implemented in various levels of hierarchy, such as various levels of cache, dynamic random access memory (DRAM), virtual memory, file systems of non-volatile memory, and physical semiconductor, nanotechnology materials, and magnetic/optical media or combinations thereof. In some examples, all the memory may be non-volatile memory or partially non-volatile such as with battery backed up memory. The non-volatile memory may include magnetic, optical, flash, EEPROM, phase-change memory, resistive RAM memory, and/or combinations as just some examples.

[0053] The CRM 108 includes a session handling module 112 to receive a first set of sessions, the set of production sessions 30, and a second set of sessions, the set of test sessions 40. The session handling module 112 may collect the first and

second set of sessions form the application monitoring solution 22, storage 24, the QAE/DevOps 18, the test management tool 19, from storage within the ASARS 20, or by a SaaS implementation of ASARS 20. A similarity and match module 114 is used to analyze what is different and similar between each test session and their respective matched production sessions. The analysis may be done using a common session alphabet. The analysis uses a matching score to decide whether compared sessions are a match or not. The result of this module is a set of matched production sessions 33 to each respective test session 40 along with their similarity scores and distance scores.

[0054] A unique/actual usage module 116 provides a usage measure between two sets of sessions. There are two flavors of usage measures which may be provided, a unique usage measure, and an actual (non-unique) usage measure. The unique usage measure is the number of unique matched base 32 (production 33) sessions that have been found as matched with a specific test 40 session or a set of reference sessions 41 out of the total number of base 31 (production 30) sessions. The actual usage measure is the actual number of non-unique matched base sessions 31 matched with a specific test session 40, or a set of reference sessions 41, out of the total number of bae 31 (production 30) sessions.

[0055] A real user coverage score module 118 use the similarity scores to calculate a coverage measure between sets of sessions. There may be three flavors of coverage measure provided by the module and each of these flavors may be applied both in the unique and actual settings. The three flavors are 1) weighted S_W, 2) optimal S_O, and 3) 1-way S_N. Each base session 31 is assigned with a similarity score (e.g. a matched session from the set of base sessions). When the coverage score is computed for a single reference session in comparison with the respective base session then each base session has a single similarity score assigned in each of the three flavors. These similarity scores are then summed over the entire base set of sessions in both the unique and actual settings and used to compute a percentage of coverage.

[0056] For a given pair test and base (production) sessions, *ref1* and *base1*, respectively, the weighted flavor S_W is the distance score outputted by the similarity and matching solution, i.e.:

$$S_W(base1) = \max(S(ref1, base\ 1), S(base1, ref1))$$

For the optimal flavor S_O, it is similar to the weighted flavor but with one exception. An unmatched base session is given a zero (0) similarity score, i.e.:

$$S_O(base1) = \begin{Bmatrix} \max(S(ref1, base1), S(base1, ref1) \;\; if \; M(ref1, base1) > threshold \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad else \end{Bmatrix}$$

The 1-way flavor $S_N$is an asymmetric similarity score (which is asymmetric as opposed to the distance score) of the reference session to the base session, i.e.:

$$S_N(base1) = S(ref1, base1)$$

When the coverage score is computed for a set of reference sessions then it is possible that a base session will be matched to more than one reference session. In that case, the base session will be assigned with the maximum similarity score among the reference sessions, i.e.:

$$S_W = \frac{max}{t_i \in reference\ set\ session} \{\max(S(t_i, base1), S(base1, t_i))\}$$

[0057] The real user coverage score module 118 may also calculate a potential coverage measure. This potential coverage measure is the delta between the coverage calculation and the best possible coverage. This measure uses the coverage and usage measures to estimate the max possible coverage improvement potential by computing usage minus coverage. Also, this module will recalculate the coverage measure assuming the discovered test gaps modification (for gap discovery module below) have been implemented and calculate the delta of the two coverage calculations.

[0058] A gap discovery module 120 has two main sub-modules, a “super-flow” alignment sub-module 122, and a gap discovery sub-module 124. The gap discovery module 120 allows for detection of a gap between similar sessions such as between a single test or reference session and the set of its respective matched production or base sessions. A gap between a test session and the set of matched or similar production sessions can be an event, a set of events, including tuples and sets of tuples, that appear (or are missing) in the test session sequence but are missing (or appear) in the sequence of events of the matched production sessions. The alignment sub-module 122 creates a “super-flow” session of a set of matched

production sessions and then in the gap discovery sub-module 124, the "super-flow" session is aligned with the respective test session in order to discover the gaps and their alignment with the test sequence.

[0059] Finally, a visualization module 130 may be included to help the AUT 12 QAE/DevOps or owner easily visualize the various measures, session sequences, and detected gaps. In some examples, the gaps are highlighted to more vividly call out their locations.

Description of the Method and CRM claim

[0060] In some examples, not all modules may be present or various modules may be combined or executed in different orders. For instance, Fig. 4 is an example flow chart 200 showing only a minimal set of modules. For instance, block 202 finds similar base (first or production) sessions for each reference (second or test) session to create a sub-set of matched base sessions. In block 204, an analysis is performed to analyze what is different and similar between each reference session and respective matched base sessions from block 202. In block 206, an overall similarity and usage scores are produced along with a real user coverage score for each reference session. Block 206 may also find gaps within the set of matched base sessions for each reference session.

[0061] Fig. 5 is an example non-transitory computer readable medium (CRM) which is a physical media 310 that stores instructions that can be executed by a processor to perform the various functions. The instructions may be organized in one or more modules on media 310 and in various orders as desired by DevOps in order to meet particular software engineering objectives and different software language approaches. In module 312, a set of instructions are used to cause the processor to handle the various sessions under test, both the first (production or base) and second (test or reference) sessions and thus is a session handling module. It may retrieve the sessions from the application monitoring solution 22, online storage 24, the QAE/DevOps workstation(s) 18, directly from the AUT 12, from within local storage in the ASARS 20 or a SaaS version of ASARS 20. A similarity and match module 314 produces a set of similarity scores, distance measures, and matching scores as well as organizing the set of first sessions into sets of matched first session and unmatched first sessions. A unique/actual usage module 316 provides one or more coverage scores based on actual and unique

usage of the various first sessions. A real user coverage score module 318 provides a set of measures of how well one or more particular second sessions are able to replicate one or more of the first sessions. The coverage score module 318 may also provide a set of potential coverage scores or measure to reflect how well the second session may be improved to better cover one or more respective first sessions.

Description of Modules

Similarity and Match module

[0062] Fig. 6 is an example block diagram that illustrates how a similarity and match module may be implemented. In Fig. 6 a first sequence 430, seq1, representing a first session and a second sequence 440, seq2, representing a second session are processed by a similarity and match module 450 for each pair (seq1, seq2) to compute a set of scores 420 such as similarity scores {S(seq1, seq2), S(seq2, seq1)}, distance score {D(seq1, seq2)}, and matching score {M(seq1, seq2)}. In one example, the provided similarity scores are in a range of [0,1] though other ranges may be used. In this example, a "1" means that the pair of sessions are substantially identical and '0' means that the pair are substantially completely not identical. A sensitivity threshold that may be adaptive based on the length of the compared sequences or otherwise adaptive may be used to determine similarity yes or no. For instance, a short session sequence would typically have a lower threshold value than a longer session sequence in one example. If a QAE/DevOPs were only interested in information on very similar sessions, the threshold may be increased.

The distance score for a pair of sessions utilizes an asymmetric similarity score whereas the similarity score calculations returns two similarity scores for a pair of sessions, namely how similar seq1 is to seq2 and how similar seq2 is to seq1. The distance score is the max value of these two similarity scores.

[0063] Fig. 7 is an example flow chart of one particular possible implementation of the similarity and match module 450. In block 452, seq1 is preprocessed to remove events not in seq2 and replaced with a "null" wildcard (joker) event. Consecutive null events are collapsed into a single null event to reduce seq1 into the essential that are used for the similarity measure calculations as noted earlier. For example, assume seq2 is a sequence of 2 events {1,5}:

1. When seq1 is the sequence {1,2,3,4,5} the preprocessed seq1 is {1,#,5}
2. When seq1 is the sequence {1, 2, 5, 3, 4} the preprocessed seq1 is {1,#,5,#}
3. When seq1 is the sequence {1,2,3,4} the preprocessed seq1 is {1, #}

[0064] In block 454, seq1 and seq2 are converted to vectors to capture order, the average location (L) of the events in the sequences. For instance, similarity calculations between the first preprocessed session (seq1) and the second unprocessed base session (seq2) are designed to take into account the relative locations of the event within each session. The closer the relative event locations are the more similar the two sessions and thus more easy to align. For example, the two sequences (preprocessed seq1 and unprocessed seq2) are converted into a vector L1, and L2 respectively, to capture the location (order) of the events in the sequence. This conversion is designed to capture for each event its order in the session. The length of these vectors is the number of unique events in seq2 and preprocessed seq1 put together. For example if seq2 is the sequence of length n: { s_1 , s_2, … ,s_i , … , s_n }, and the number of the unique events in seq2 is k denoted by { e_1, e_2, …, e_k}, then the preprocessed seq1 is now constructed of at most these k events and maybe also the null event "#". Therefore the number of unique events per sequence is either k or k + 1. Now for each sequence and for each event in each sequence, the event's average relative locations in the sequence is computed. The relative location is the event's order in the sequence divided by the length of the sequence, e.g. the relative location of event "3" in a sequence "1, 2, 3" is 3/3. Thus if event e_i appears in location j and k in sequence seq2 then $s_k = s_j = e_i$. For event e_i the average location in seq2 would be the average (j/n, k/n). Now for each unique event in L2 and L1 there is its average relative locations in that sequence. If an event does not exist in the sequence then the value for that non-existent event is set to zero. For example, considering the above example the unique set of events is {1,#,5}. The event vector L1 for processed sequence seq1 {1, #, 5, #} would be [1/4, avg(2/4,4/4), ¾], thus L1=[0.25, 0.75, 0.75] and L2=[0.5, 1, 0].

[0065] In block 456, the similarity scores are computed between the two sequences seq1 and seq2 based on the distance of each unique event's average location in each sequence. For each event in L1 and L2 apply the norm-L1 notion,

i.e. abs(L1-L2). First, a location distance is calculated for each event that is the distance between its average locations in the sequences. Next, the similarity per event is computed as the location distance of the event from the maximum. Since the locations of the events are relative locations then the maximum distance relative location possible is 1 and the minimum relative location possible is 0. Therefore, the maximum distance of relative locations is 1. The similarity for the sequence is the average of these similarity scores per event. For example, if the sequences are identical then the average locations for each event per sequence is the same. Therefore, for each event the distance between the average locations will be zero and the distance from the maximum will be 1. Identical sequences will get a maximum similarity score for each event of "1" and the overall similarity score of sequence will be "1". The equation for calculating the similarity score given the average locations of the events can be described as:

[0066] $S(seq1, seq2) = sum(\ abs(1 - abs(L1 - L2))\ ) / length(L1)$; and

[0067] $S(seq2, seq1) = sum(\ abs(1 - abs(L2 - L1))\ ) / length(L2)$

[0068] Each similarity score is asymmetric in nature in that it assumes that one sequence is the basis for comparison. Accordingly for each pair of sequences measure S(seq1, seq2), preprocesses seq1 with respect to seq2 and measures how similar is the first session to the second session, whereas S(seq2, seq1) preprocesses seq2 with respect to seq1 and computes how similar is the second session to the first session. The resultant two similarities are not necessarily identical. For example, seq2 is a sequence of 2 events: {1, 5} and seq1 is the sequence: {1, 2, 3, 4, 5} then S(seq1, seq2) = 0.72 and S(seq2, seq1) = 0.58.

[0069] In block 458, the distance measure is computed. The distance measure should be symmetric and transitive therefore the distance between two sessions is defined as the max between the two similarity scores. Accordingly, the distance measure is computed as:

$$d(seq1, seq2) = d(seq2, seq1) = max(S(seq1, seq2), S(seq2, seq1))$$

[0070] In block 460, a matching mechanism is computed based on the distance. Since the distance and similarity scores are always a number between [0, 1] in a manner proportional to the similarity independent of the session lengths and

alignment, a matching mechanism is provided based on this distance. Basically, the matching mechanism is based on a threshold. An example default threshold is 0.75, however, this can be modified either manually or automatically. For short sequences this default may be too strict, therefore, one may automatically apply different thresholds to different lengths. Moreover, given a new manual setting, the matching mechanism may learn the new preferences of the user, by remembering past settings and extrapolating for the future.

[0071] The match criteria between two sequences, seq1 and seq2 uses the similarity score and a threshold as follows:

$$M(seq1, seq2) = max\big(S(seq1, seq2), S(seq2, seq1)\big) \geq threshold$$

[0072] Where the threshold can be tuned automatically or manually. The threshold is a value between [0,1]. In one example, the default threshold for sequences of length larger than 10 may be 0.75. For increased sensitivity the threshold should be higher for lower sensitivity or short sequence length the recommended can be decreased. It is recommended in most examples to set the threshold at least at 0.5. When the match criteria is greater than the threshold, a match is determined as 'yes'. If the match criteria is less than the threshold, a match is determined as 'no'. If the match criteria is equal to the threshold, the DevOps developer can choose either 'yes' or 'no' depending on the level of matching desired. In block 460, as an example, when the threshold is equal to the match criteria, it is also accorded a 'yes'.

[0073] The system output for a pair of sessions as noted in block 420 of Fig. 6 for seq1 and seq2, includes information 36 of at least four measures or scores:

$$S(seq1, seq2)$$
$$S(seq2, seq1)$$
$$D(seq2, seq1)$$
$$M(seq1, seq2)$$

Gap Discovery

[0074] Fig. 8 is a simplified flow chart 500 of the gap discovery module 120 of Fig. 3. Gap discovery module looks for 'gaps' of events between similar sessions, such as between a single reference (test) session and the set of its similar base

(production) sessions. In block 510, a "super-flow" of a set of similar production sessions for each reference session is created using an alignment technique. A gap between a reference session and the set of similar base sessions can be an event or set of events that appear (or are missing) in the reference session sequence but are missing (or appear) in the sequence of events of the similar sessions. In block 520, the super-flows of similar or matched base sessions are aligned with the respective reference session in order to discover the gaps and gap alignment in the reference sequence.

[0075] Fig. 9 is a flow chart 600 of an exemplary implementation of an alignment technique to create a "super-flow" for a set of similar or matched sessions. Thus, one assumption that helps assure the success of this technique is that the aligned sessions have already been found to be similar in the similarity and match module 114, 314. The purpose of this technique is to find the shortest "super-flow" that aligns all these similar sessions into one symbolic session. The multiple alignment of the similar base sessions is run according to their length, starting from the two most longest and aligning the shorter one to the longer similar base session using a Levenshtein distance computed measure. Accordingly, in block 602, the 'n' sessions are sorted according to their length. A merged alignment of these two sessions is created in block 604 where the shorter is aligned with respect to the longer flow. Then, the next longest similar base session is iteratively aligned with the newly merged alignment. In block 606, if two sessions, seq1 and seq2, are of the same length this alignment is applied at least twice; 1) once with seq1 in respect to seq2 and 2) vice versa and the technique proceeds with the shorter merged alignment of the two options. In block 608, the next longest session is iteratively aligned with the previously created merged alignment. In block 610, when there are no more 'n' sessions, a "super-flow" is created from the last remaining alignment which is the result of aligning and merging the 'n' similar base sessions.

[0076] By the order of length before aligning the sessions, this ordering improves the "super-flow" length by more than 30% than when ordering by length is not done. The more sessions to align the larger the improvement. The order of alignment process utilizes the trait that the aligned sessions are similarly matched as defined by the similarity and matching solution, i.e. that there is a similarity in the events and order of events. The alignment process in another example may also align from shortest to longest to generate the shortest super-flow. In yet another

example, an even better and shorter super-flow may be created when also applying reverse alignment, i.e. from end to start.

[0077] Accordingly, an alignment may be done by using a single Levenshtein distance alignment process. However, the use of a Levenshtein distance alignment process may be sensitive to multiple alignment options. In one example, an improvement to the alignment resulting from aligning by order of session length may be achieved by executing the Levenshtein distance alignment process at least twice: 1) once aligning the sessions from start to end and 2) once running the alignment process on the reversed sessions. Next the number of modifications (deletions or insertion) and the relative locations of the aligned events are computed and the alignment process proceeds with the "super-flow" that requires the smallest number of modifications as well as optimizes the relative location similarity. In some examples, the alignment may occur times total for portions of the sessions: Once aligning the sessions from start to end and on the reversed sessions in order to find possible alternative starting points with improved similarity distance and then the sessions are further aligned given the new starting point (which means that the event session portions preceding the chosen starting point are not included in third alignment).

[0078] These additional steps, ordering and executing Levenshtein distance alignment at least twice, may assure that the session alignment for super-flows better handles the issue of multiple alignments options while optimizing the relative location similarity. Another outcome of this alignment process is that for each event in the "super-flow" and each base session there is also the order (location of the event in the base session) of the event that matched from the base session.

[0079] Fig. 10 is an example flow chart 700 for aligning the resultant super-flow with a respective test (second or reference) session 40 to find the shortest alignment between the two. In block 702, for each set of matched sets of matched production (first or base) sessions 33, a respective "super-flow" is created. In block 704, a test session (seq1) is aligned with the respective "super-flow: (seq2) by aligning seq1 with respect to seq2 and seq2 with respect to seq1 and taking the shortest alignment to create a base "super-flow". Therefore, the same alignment technique described in Fig. 9 of aligning the shorter session to the longer session is used. Generally, the longer session would be the base "super-flow" however this may not always be the case.

[0080] In block 706, for each event in the "base super-flow" aligned with the test session the percentage of the event's occurrence among the base sessions as well as each event's locations among the test sessions is determined. In block 708, based on a threshold it is determined for each event whether it is a major gap or a minor gap. The default for an event occurrence to be a major gap is 30% of all base sessions in the set of similar sessions. This threshold may be adaptively changed based on the size of the set of similar sessions. Finally, in block 710, consecutive minor gaps are collapsed into a single node of gaps.

[0081] When the test set is not a single session but multiple sessions one can similarly create with a test "super-flow" for the test sessions and compare that with the base "super-flow" as described above for Fig. 10.

[0082] In some examples, what is desired is to determine gaps in unmatched sessions to find what user sequences of events are not being covered by the test sessions. Fig. 11 is an example flow chart 800 of a gap discovery method or module for unmatched sessions. For instance, in either of the session data sets there may be sessions that have no similar session matched from the respective set. In block 802, unmatched production (first or base) sessions are found for each of the test (second or reference) sessions. These are the unmatched sessions and there is a lot of important value that can be discovered and highlighted here for the QAE/DevOps and owners of the AUT 12 to investigate. In the production sessions, these are the user sessions that are completely uncovered by any test. These are the application's 'blind spots' in testing. In other words, this example is the case where the test session is an empty group and therefore the production sessions in the set of unmatched sessions are not clustered into subsets of similar sessions, as in the examples described above. In block 804, subsets of similar sessions are discovered within the set of unmatched sessions using the above similarity and matching module of Figs. 6 and 7. In block 806, the unmatched sessions set are used as the "test" set and matched to the base production sessions set. The process of similarities and match needs to be run only once. The similarities and matching scores described above are used for online clustering the unmatched sessions into subsets of unmatched sessions. In block 808, for each of these subsets, the unique/actual usage and coverage scores or measures as described above are computed. In this case the usage is the size of the subset. Block 810, enables an application owner to prioritize these subsets based the calculated usage

and coverage that reflects the size of the blind spot. Furthermore, for each session in the subset there is a coverage calculation as described above. In block 812, the session with the highest coverage may be chosen as the class representative. Also in block 814, for each of these subsets, a "super-flow" is created as described in Fig. 9 as a representative of the merged sessions in this subset. Further, gap discovery may proceed based on the class representative session and the set of production sessions.

[0083] Gap discovery may also be done for uncovered tuples of consecutive events. A tuple of consecutive events is an event or several consecutive events that appeared consecutively in collected sequence of events (i.e. a session). For example, given a set of these 3 sessions:

1. A, b, c, d
2. A, b
3. A

There are four possible tuples that are limited to a length of two consecutive events: "A", "A,b", "b,c", "c, d". The respective usage score values are: 100%, 66.67%, 33.33%, 33.33%, 33.33%, 33.33%, 33.33%.

[0084] For each event or short sequence of events an analysis of usage and best representatives of production sessions may be provided. In this example, for each tuple of events, an example solution provides its real usage score in the production sessions as well as in the test sessions, where a tuple of events is a short subsequence of consecutive events of length at least 1 and as small or long as the app owner sets it. Note that the real user usage score here is different than the frequency. For each such tuple there is a set of production sessions in which it appears. These are the events of a set of matched sessions upon which example implementations apply the clustering and best representative technique.

[0085] The maximum length of a tuple can be modified (e.g., manually by the application owner). The default tuple limit length is three. For each of these tuples of events in the base sessions example solutions have the percent of unique/actual sessions in which it appeared in the non-base sessions set as well as a in the base sessions, each such list is considered a subset of matched base sessions. For each of these subsets the technique will provide its unique/actual usage calculation in the test subsets as well as in the production subsets, along with a best representative for of the subsets (highest coverage).

[0086] These tuple of events will be sorted according to the lowest usage in test and highest usage in production. This prioritization puts at the top of the list sessions that appear in production but are missing from any of the test sessions, these are referred to as uncovered event tuples.

[0087] Fig. 12 is an example flow chart 900 of a method or module for gap discovery for uncovered tuples. In block 902, single events as well as tuples of consecutive events that appear only in the base (production) sessions are considered that are infrequent or do not appear in any of the reference (test) sessions. The consecutive events may be such that they are frequent in one set but rare in the other set. Accordingly, the definition of uncovered in this context is that they are "rare" or infrequent. This definition may include or mean none or infrequent with a very low frequency with some preset threshold. This 'consecutive' threshold may also be set dynamically as a ratio of the frequency in the base set.

[0088] The maximum length of a tuple can be modified. In one example, the default tuple limit length is three. For each of these tuples of events in the base sessions, in block 906, the percent of unique/actual sessions in which it appeared in the base sessions set as well as a list of the base sessions in which it appears is known. Each such list is considered a subset of base sessions which may be chosen to investigate. For each of these subsets, its unique/actual usage calculation is provided, as well as one or more representative(s) of the subset with the highest coverage similar to the gap discovery in unmatched sessions described above. In block 908, the gap discovery for each subset of base sessions for each tuple/combination of events can be analyzed.

[0089] Further, in many cases different uncovered events may appear in the same sessions or vice versa, such as with combinations of uncovered tuples of events. The purpose of this additional module is to allow the application owner to choose any subset of the uncovered tuples of events and this module will provide it with best representative/s sessions with the highest coverage. This will be done by using the unified list of base sessions in which these tuples of events appear as the set of "unmatched" sessions. Then a gap discovery process similar to the unmatched sessions described above is applied in order to select the base sessions' best representatives ordered by the real user coverage score.

Further Recommendations Based on Production Data

[0090] Fig. 13 is an example implementation 1000 of an ASARS 20 with visualization of the analysis and recommendations along with linking of tests to an application's product area or feature. Example ASARS 20 implementations may rely on the captured sessions from production and test session's data. Principally, example implementations may generate, for each session a sequence of events as monitored and collected by the application monitoring solution 22. In some examples, sets of sessions may be generated using the same applications monitoring solution 22 in different environments for the same application.

[0091] Example ARS 20 implementations may include data collection and preprocessing preparation, an analysis that includes session matching, analysis & recommendation through session comparison and analysis, as well as a visualization of the findings along with recommendations.

[0092] Example data collection from the test and production and preparation environments may convert the collected events into unique events and clean parameter values and reconstruct sessions (sequence of clean unique events). In some examples, abnormally long sessions may be split into smaller sessions based on the time lap between events. The data collection and preparation phase may also include a link between the collected test sessions and their respective product areas and features.

[0093] Example data collection and preparation steps may include:

1. Data collection. This step may include the sessions' data collection from production or test environments. It also may include the collection of necessary data from a test management tool 19 (see Fig. 1) controlled by the QAE/DevOps 18. The collection 1010 of sessions' data may include two short scripts, the first script for receiving the data collected by the application monitoring solution 22 both from the production environment 1016 that is displayed in production window 1028 as well as a test environment 1012 that is displayed in test window 1018. The second script 1014 is specific for the test environment where it injects into the session data the test id or name and steps and display in injected test window 1022. This data may be used later on to link 1020 the product areas and feature of an application to the test-production related analysis from similarity and

match modules 1026 and perhaps gap discovery results via test name and displayed in the results window 1024. The test name and link to product area or features may be received from the test management tool 19.

2. Data processing & reconstruct user sessions. There may be a data processing phase to generate the unique events from all sets of sessions' data as well as the unique sessions constructed from the unique events for each of the data sets. The collected data may contain a unique session id, e.g. a cookie, along with a timestamp from each event in the session. These two collected values may allow the reconstruction of the sequence of events per session. Many times an event contains the event type or name or identifier and possibly additional parameter values, e.g. a response time measurement per event in this session. The task at hand may be to identify and clean out these parameter values and create a dictionary of the unique event name as given in the collected data. The immediate by product would be to clean these parameter values from the reconstructed sessions.

[0094] Example:

The collected data may include the following events:
"AgM/release/release_backlog*FullLogin*15seconds"
"AgM/release/release_backlog*FullLogin*23seconds"
"AgM/release/release_backlog*FullLogin*54seconds"
...

[0095] These are all the same event of "AgM/release/release_backlog* FullLogin*seconds" with a parameter value of time that varies, e.g. 15, 23, 54 etc. In order to reconstruct the unique events and sessions, example implementations may need to identify and clean these parameters. For this end example implementations may analyze strings identifying the unique patterns and parameters and cleaning them out. For example, in the above example there are 3 events. All 3 events are the same event, i.e. "AgM/release/release_backlog*FullLogin*15seconds". The event is "AgM/release/release_backlog*FullLogin*seconds" and the parameter values for this event in this session is 15. So the clean session may include the event without the parameter value. The parameter value may be stored for future use

with its reference to session and event. The output may be a list of unique events ("dictionary") collected across all session data sets. Each collected event may be mapped to a unique "clean event". Now each session may be cleaned of the parameter values by mapping its original events to the respective "clean events".

[0096] Example Implementation Steps:

a) Receive collected raw data
b) Sort collected events by session identifier and time stamp to reconstruct sessions.
c) For each session collected from the test environment extract the injected test id or name and test step.
d) Convert collected events to unique clean events by using the solution to identify and remove parameter values in the collected events
e) For each reconstructed session map the original events to the unique "clean events" from the previous step
f) Create a unique set of "clean sessions". For each clean session count the number of its occurrences in the data set. A unique "clean sessions" is unique in the order and sequence of its "clean events".
g) Abnormally long sequences may be split into smaller sessions. The "normal" length of the sessions is defined by the length of the sessions in the test set. The test sessions are not split. The production sessions may be extremely longer than the normal test session length and may need to be split into smaller sessions. In which case it will be done by analyzing the time differences between consecutive events in sessions for normal and abnormal time lapses.

[0097] Continuing with the example data collection and preparation steps, the steps may further include:

3. Link test data to product areas and features. This step may use the extracted test id or name extracted from the collected test sessions and finds it in the data received from the test management tool. The linking may be performed based on the commonalities.

4. Fill in the missing test steps that exist in the actual test as appears in the test management tool but are not collected by the application monitoring solution 22 due to the, naturally occurring, partial instrumentation and collection. Use, for example, the test name extracted per test session in the above steps to identify the actual test and complete test steps in the test management tools and reconstruct the test session with the missing test steps. These missing test steps are used for a part of the analysis regarding test steps analysis, and may not be considered as part of the test session.

[0098] Example steps performed by an ARS 20 implementation may include at least one analysis and recommendation module 1030, for instance:

1. For each pair of sessions from test and production example sessions applying a method of measuring similarity 1026 between user sessions as described, for example, above. For example, each pair of sessions now has the similarities, distance, and match scores. The provided similarities and distance scores are in the range of [0,1]. Where 1 means this pair of sessions is nearly completely identical and 0 means that it is completely un-identical. The default sensitivity threshold is 0.75. This sensitivity factor can be modified given the length of the compared sessions. When the compared sessions are short then a lower sensitivity threshold may be desired. When the application owner is interested in information on only the very similar sessions the threshold should be increased and lowered vice versa.

2. Discover the set of unmatched production sessions 1032 and provide the best representing sessions. For example, this step may use the above results of the matched score between test and production sessions. Production sessions that have no match to any of the test sessions may be considered as the set of unmatched sessions. This set of unmatched production sessions is then analyzed as described above to iteratively find the clusters of sessions and the best representative for each cluster.

3. For each product area or feature (this refers also to the overall product) compute the unique real user coverage and usage scores 1033. This step

includes assigning each product area or feature with its linked tests as captured in the test management tool.

[0099] The next step 1034 is to assign each product area or feature with the production sessions that were assigned to any of the linked test sessions. In this step not all test sessions are valid for the linking of production sessions. Example solutions may apply an adaptive sensitivity threshold regarding the number of matched production sessions per test session. For example, if the default matched session sensitivity threshold is five and the linked test session has less than five matched production sessions then it will not be used in this linking step of production sessions to product area. This adaptive threshold can be modified based on the total volume of production sessions. This may be followed by computing the coverage and usage scores using the unified set of matched production sessions and their respective similarities and distance scores. Each production session may have more than one test session matched. In which case, example solutions use the max scores of the matched test sessions for each of the production sessions in this set. There may exist other test sessions that are matched to this production session but assigned to a different product area/feature, in which case they will not be used in this coverage and usage calculations.

[0100] For instance, this is an explanation of how an example session looks in the real user coverage and usage scores per product area described below:

Sequence 1 (seq1) is a sequence of nine events

1. Login
2. Open Chat
3. Attach
4. Write Msg
5. Write Msg
6. Write Msg
7. Attach
8. Write Msg
9. Send

For simplicity, the sequence of coma separated events can be written as, e.g. "Login, Open Chat, Attach, Write Msg, Write Msg, Write Msg, Attach, Write Msg, Send". Also, each unique event can be converted to a unique numeric representation or alphabet in which case the above sequence can be described as "1, 2, 3, 4, 4, 4, 3, 4, 5"

Given a second sequence (seq2), e.g. a sequence of four events

1. Login
2. Open Chat
3. Write Msg
4. Send

they may be written in the common alphabet as:
"1, 2, 4, 5".

[0101] Below in Table 1 is an example of a set of test sessions with their linked product areas/features associated with the AUT 12 and a set of recorded production sessions to demonstrate the selection process of assigning each product area/feature of the AUT 12 with a set of matched production sessions via the test sessions. For this example the adaptive threshold for using the linked test is four.

First, assume that the set of test sessions include the following sessions:

1. Test flow1 is "1, 2, 3, 4, 5" and assigned to feature Y
2. Test flow2 is "1, 2, 3, 3, 3, 4, 5" and assigned to feature Y
3. Test flow3 is "1, 2, 5, 3, 4" and assigned to feature Y
4. Test flow4 is "1, 2, 3, 4" and assigned to feature X

Second, assume that the production set of sessions includes the following sessions:

- Production flow1 is "1, 5"
- Production flow2 is "1, 5"
- Production flow3 is "1, 5"
- Production flow4 is "1, 5"
- Production flow5 is "1, 5"
- Production flow6 is "1, 5"

- Production flow7 is “1, 5, 6”

[0102] The similarity and matching results between the unique productions sessions and the four test session flows are tabulated in Table 1.

Test Session Flows	Total number of matched production sessions per test	Unique Production Session flow “1, 5”	Unique Production Session flow “1, 5, 6”
“1, 2, 3, 4, 5” (feature Y)	5	Match=1; Similarity = [0.58, 0.72] Distance = 0.72	Match=0; Similarity = [0.46, 0.5] Distance = 0.5
“1, 2, 3, 3, 3, 4,5” (feature Y)	5	Match=1; Similarity = [0.58, 0.72] Distance = 0.72	Match=0; Similarity = [0.46, 0.5] Distance = 0.5
“1, 2, 5, 3, 4” (feature Y)	0	Match=0; Similarity = [0.58, 0.42 Distance = 0.58]	Match=0; Similarity = [0.52, 0.43 Distance = 0.52]
“1, 2, 3, 4” (feature X)	0	Match=0; Similarity = [0.33, 0.3 Distance = 0.33]	Match=0; Similarity = [0.29, 0.3] Distance =0.3

Table 1 - Example of sets of test sessions and unique recorded production sessions checked for Match, Similarity, and Distance

[0103] In this example feature Y has three test sessions that are linked in the system but only two of those sessions cover more than five production sessions and therefore only these two will be used for discovering the set of matched production sessions per feature.

[0104] In order to calculate the overall real user coverage and usage an example solution applies the same logic for discovering the unified set of matched production sessions over the complete set of test sessions regardless of its link to any product area or feature, since there maybe tests that may not be linked in the QAE set of application tests and but may be relevant to the application quality and coverage.

[0105] Example analysis and providing visual and actionable recommendations for test enhancement accordingly may also include at least one of the following, but not limited to, functionalities:

a. Discover the uncovered production sessions 1032 by computing the similarities and match between the test sessions and the production

sessions. Note that the sensitivity threshold can, in some examples, be tuned to fit the need for more relaxed matching.

b. Compute a real user coverage and expected usage scores 1033 for each product area and feature by computing the similarities and match between the test sessions and the production sessions. Note that the sensitivity threshold can be tuned to fit the need for more relaxed matching.

c. Compute an expected usage score for synthetically generated user sessions under design. Note that the sensitivity threshold can be tuned to fit the need for more relaxed matching.

d. Provide a unique event based analysis which provides per event or short sequence of events the best session representatives from production.

e. Execute the test analysis and enhancements process 1034 (as described in above) over the collected test sessions and the production matched and unmatched sessions.

f. For each production session provide a frequency score 1035 in production along with a best test coverage score and enhancements per test vs. production session recommendations.

g. Find the uncovered events and/or tuples from production sessions 1036 that are not included in the set of test sessions.

h. Find uncovered test steps from production sessions 1038 to allow for improving the set of test sessions.

[0106] Many different ways of visualizing the analysis and recommendations may be chosen from selecting from example visualization options 1040, including but not limited to, Venn diagrams 1042 and product tree maps which represent the various scores by size and colors based on product features or other application usage areas. In some examples, there may be a coverage matrix 1046 that displays the relative coverage between the test sessions and the unique production sessions along with a matching result. This matrix visualization may be helpful in determining where an adaptive threshold setting should be set by quickly visualizing all scores while the threshold is manually changed. Another example visualization is providing suggested test coverage improvements 1048 to help the user of the ASARS 20 understand how the tests may be modified. Yet another visualization might be to provide a list of which production tuples 1050 of common event sequences are not

covered by the test flows but used often in the real user production sessions. Several other possible visualizations are possible and fall within the intended scope of the claims.

[0107] In summary, the ASARS 20 may include several functionalities, providing real user coverage and usage scores per session and product area/feature by discovering the matched and unmatched sessions from production, applying the solution described above for providing test enhancements based on real users' sessions, as well as linking and aggregating these results at the product area level.

[0108] While the claimed subject matter has been particularly shown and described with reference to the foregoing examples, those skilled in the art will understand that many variations may be made therein without departing from the intended scope of subject matter in the following claims. This description should be understood to include all novel and non-obvious combinations of elements described herein, and claims may be presented in this or a later application to any novel and non-obvious combination of these elements. The foregoing examples are illustrative, and no single feature or element is essential to all possible combinations that may be claimed in this or a later application. Where the claims recite “a” or “a first” element of the equivalent thereof, such claims should be understood to include incorporation of one or more such elements, neither requiring nor excluding two or more such elements.

What is claimed is:

CLAIMS

1. A method (200) for application session analysis and recommendation, comprising:

finding (202) similar sessions of events in a first set of sessions (29,30,31) for each session with respect to each session of events of a second set of sessions (39,40,41) to create a sub-set of matched sessions (32,33) from the first set of sessions for each of the second set of sessions;

analyzing (204) what is different and similar between each session of events in the second set of sessions and respective sub-set of matched sessions; and

producing (206) an overall similarity score and usage score along with a real user coverage score for each session of events in the second set of sessions.

2. The method of claim 1, further comprising providing (812) at least one best representative session from the respective sub-set of matched sessions for each session of events in the set of second sessions.

3. The method of claim 1, further comprising:

creating a super-flow (600) of the sub-set of matched sessions for each session of events in the second set of sessions; and

finding gaps (500) of the first set of sessions for each session of events of the second set of sessions.

4. The method of claim 3, further comprising aligning (200) the respective super-flows for each session of events in the second set of sessions in order to discover the gaps and gap alignment for each respective session of events of the second set of sessions.

5. The method of claim 4, wherein the aligning (700) includes using a Levenshtein Distance measure (604) at least twice to create a respective first and second merged alignment for each direction and computing the number of modifications of events and relative locations of aligned events for each direction and proceeding with the merged alignment with the smallest number of modifications while optimizing relative location similarity.

6. The method of claim 3, further comprising creating (704) a super-flow of the second set of sessions and comparing the super-flow of the second set of sessions with a base super-flow created from the super-flows of each of the sub-set of matched sessions in the first set of sessions.

7. The method of claim 1, wherein the events in the first set of sessions include tuples of events which do not occur (902) in the second set of sessions.

8. A system (100) for application session analysis and recommendation, comprising:

a processor (102) ; and

a processor readable memory (108) including instructions (110) and coupled to the processor to allow the processor to execute the instructions to:

find (202) similar base sessions (29,30,31) of recorded events for each reference session of a set of reference sessions (39, 40, 41) of recorded events to create a set of respective matched base sessions;

analyze (204) what is different and similar with respect to the recorded events between each reference session and the set of respective matched base sessions; and

produce (206) an overall similarity and usage score along with a real user coverage score for each reference session.

9. The system of claim 8, wherein the set of reference sessions (39, 40, 41) is comprised of a set of base sessions (29, 30, 31) that are not matched to any other base sessions.

10. The system of claim 8, wherein the instructions further cause the processor to:

create (510) a super-flow alignment of recorded events of a set of similar base sessions for each reference session with an alignment technique using a Levenshtein Distance measure (604) at least twice; and

find gaps (520) of recorded events in the set of base sessions for each reference session.

11. The system of claim 10, wherein the instructions further cause the processor (102) to align the super-flows of recorded events of the set of respective matched base sessions with the respective reference session.

12. The system of claim 10 wherein the instructions further cause the processor (102) to create (510) a super-flow of recorded events for each reference session and comparing (520) the super-flow of each reference session with a base super-flow of recorded events created from the super-flow of recorded events of each of the similar base sessions for each reference session.

13. A non-transitory computer readable memory (300) comprising instructions (310) in modules that when executed by a processor (102) cause the processor to execute the code in the modules for application session analysis and recommendation, the memory comprising:

a sessions handling module (312) to receive a first set of sessions (29, 30, 31) of recorded events and a second set of sessions (39, 40, 41) of recorded events;

a similarity and match module (314) to process the first and second set of sessions to find similar first set of sessions of recorded events for each second set of sessions of recorded events to create a set of respective matched sessions (32, 33); and

a usage module (316) to analyze what is different and similar with respect to the recorded events between each session in the second set of sessions and the set of respective matched sessions and produce (206) an overall similarity and usage score for each session in the second set of sessions.

14. The computer readable memory (300) of claim 13, the memory further comprising a gap discovery module (120) to find gaps of recorded events in the set of respective matched sessions for each session of the second set of sessions of recorded events, wherein the gap discovery module further includes instructions to create a super-flow alignment (600) of recorded events of the respective matched sessions for each session of the second set of sessions with an alignment technique using a Levenshtein Distance measure at least twice.

15. The computer readable memory (300) of claim 13, the memory further comprising a visualization module (130) to display the real user coverage score for each session in the second set of sessions along with the discovered gaps with actionable information to improve the overall similarity and usage score for each session in the second set of sessions.

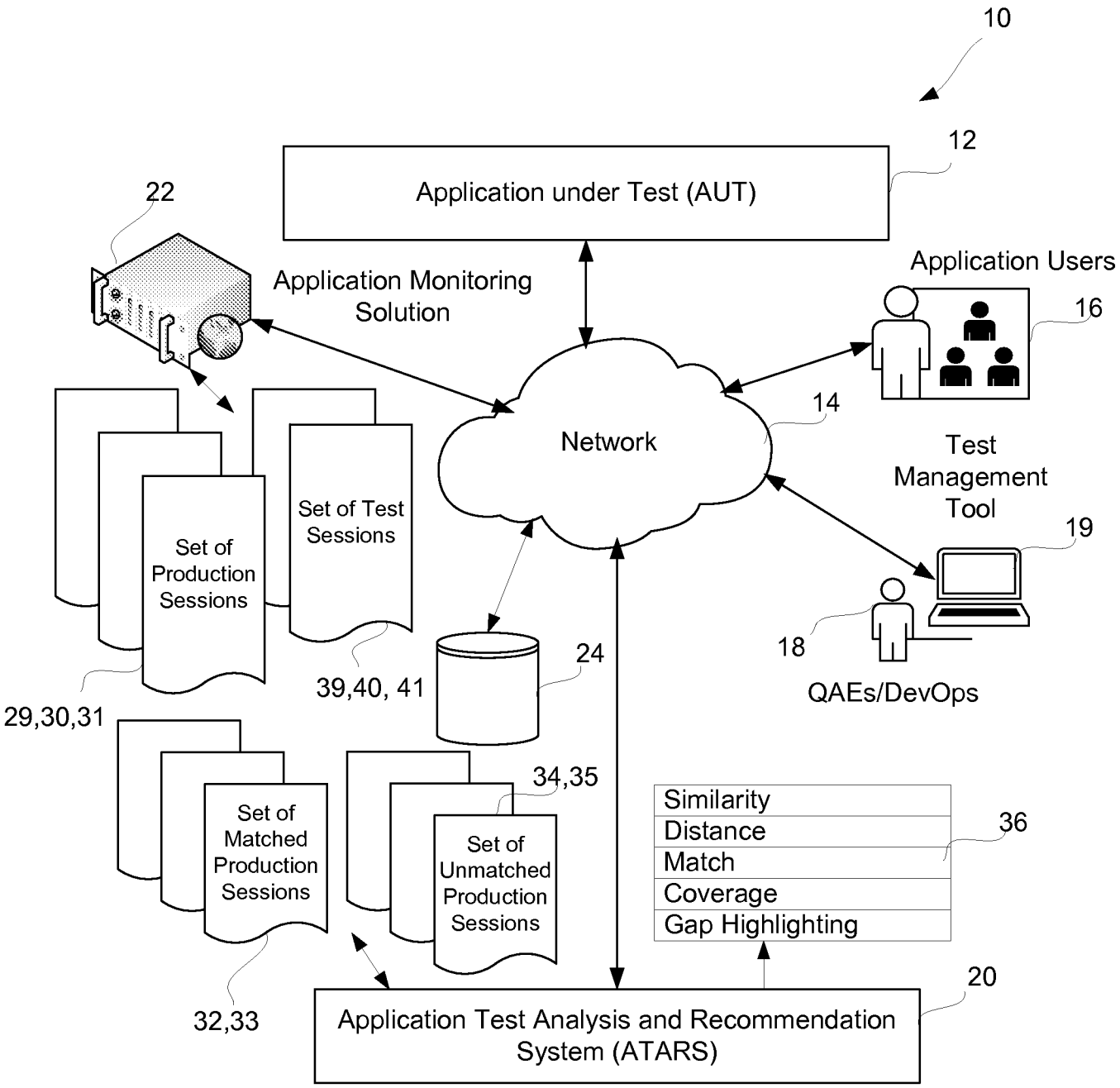
10
12
Application under Test (AUT)
22
Application Monitoring Solution
Application Users
16
14
Network
Test Management Tool
19
Set of Production Sessions
Set of Test Sessions
24
18
QAEs/DevOps
29,30,31
39,40, 41
34,35
Set of Matched Production Sessions
Set of Unmatched Production Sessions
Similarity
Distance
Match
Coverage
Gap Highlighting
36
20
32,33
Application Test Analysis and Recommendation System (ATARS)

Fig. 1

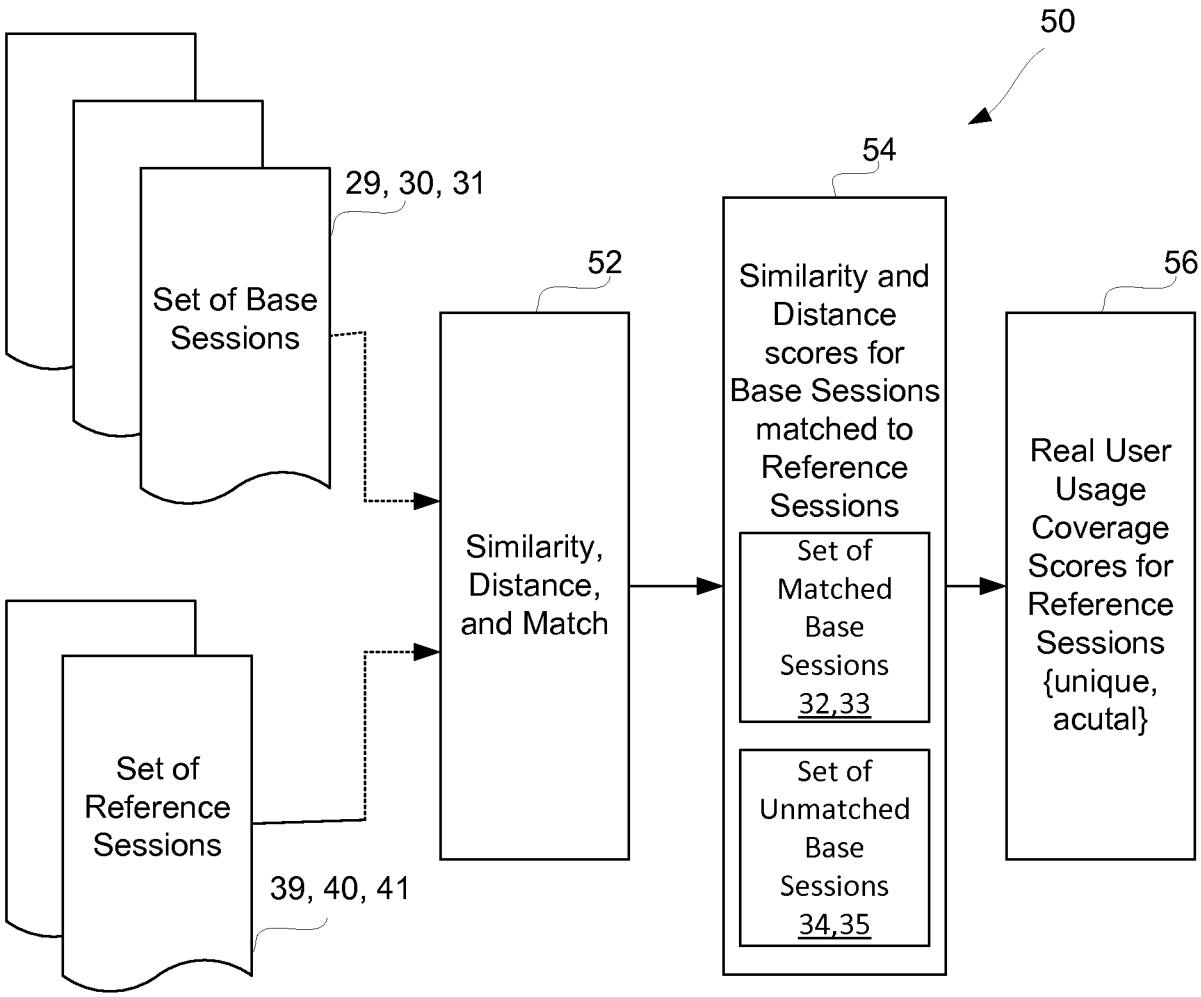
50
54
29, 30, 31
Set of Base Sessions
52
Similarity and Distance scores for Base Sessions matched to Reference Sessions
56
Similarity, Distance, and Match
Set of Matched Base Sessions 32,33
Real User Usage Coverage Scores for Reference Sessions {unique, acutal}
Set of Unmatched Base Sessions 34,35
Set of Reference Sessions
39, 40, 41

Fig. 2

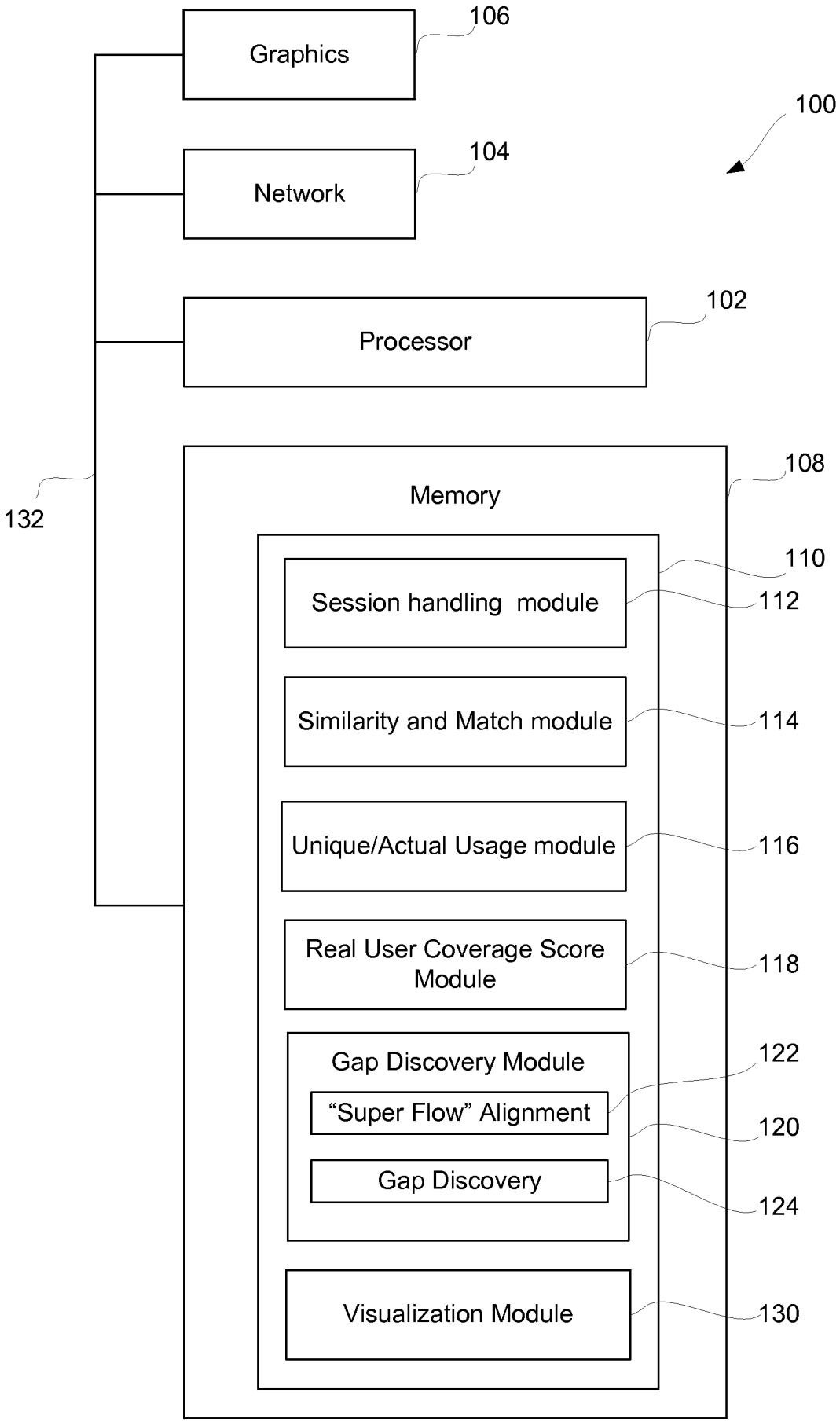
Graphics
106
100
Network
104
Processor
102
132
Memory
108
110
Session handling module
112
Similarity and Match module
114
Unique/Actual Usage module
116
Real User Coverage Score Module
118
Gap Discovery Module
122
"Super Flow" Alignment
120
Gap Discovery
124
Visualization Module
130

Fig. 3

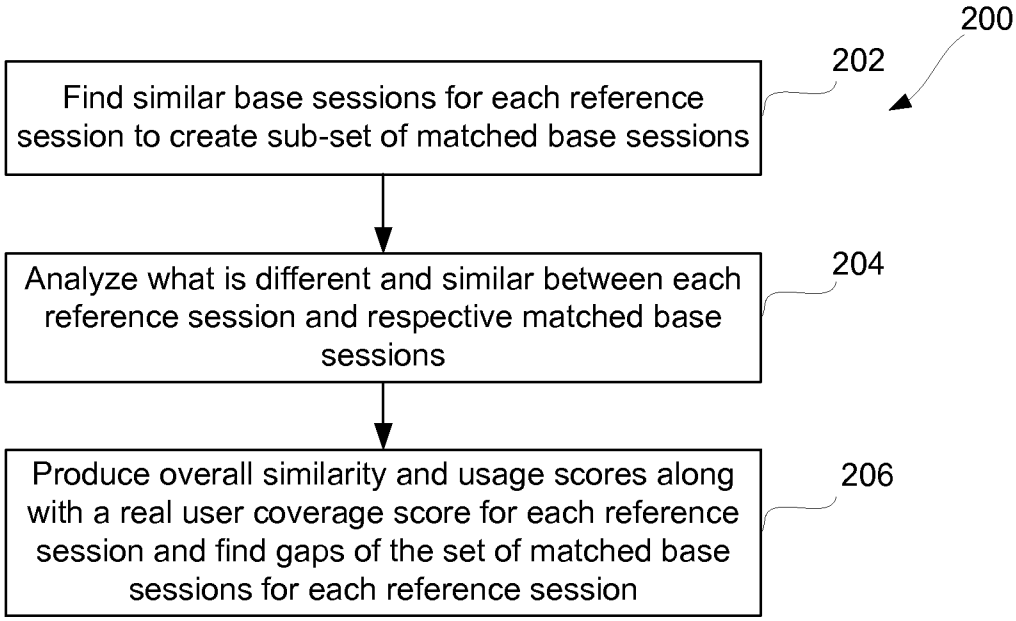
200
202
Find similar base sessions for each reference session to create sub-set of matched base sessions
204
Analyze what is different and similar between each reference session and respective matched base sessions
206
Produce overall similarity and usage scores along with a real user coverage score for each reference session and find gaps of the set of matched base sessions for each reference session

Fig. 4

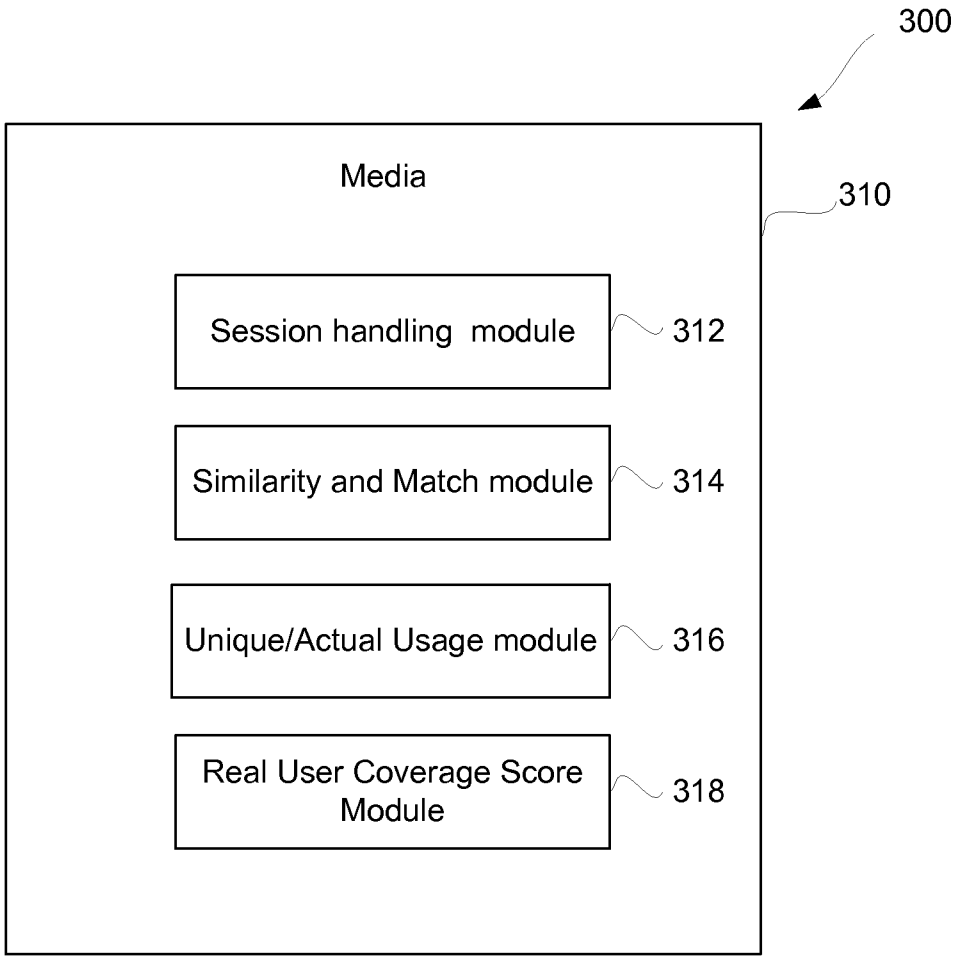
300
Media
310
Session handling module
312
Similarity and Match module
314
Unique/Actual Usage module
316
Real User Coverage Score Module
318

Fig. 5

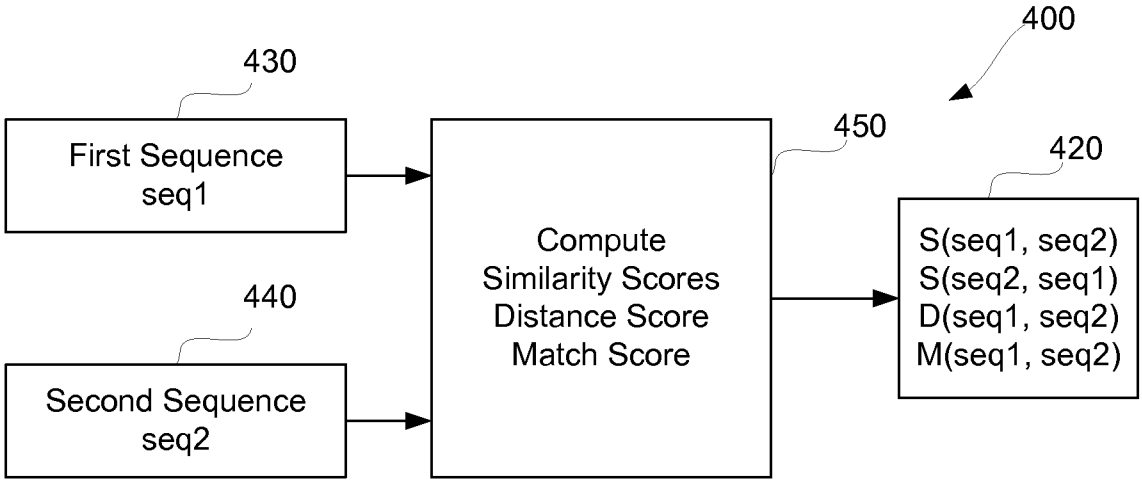
400
430
First Sequence
seq1
440
Second Sequence
seq2
450
Compute
Similarity Scores
Distance Score
Match Score
420
S(seq1, seq2)
S(seq2, seq1)
D(seq1, seq2)
M(seq1, seq2)

Fig. 6

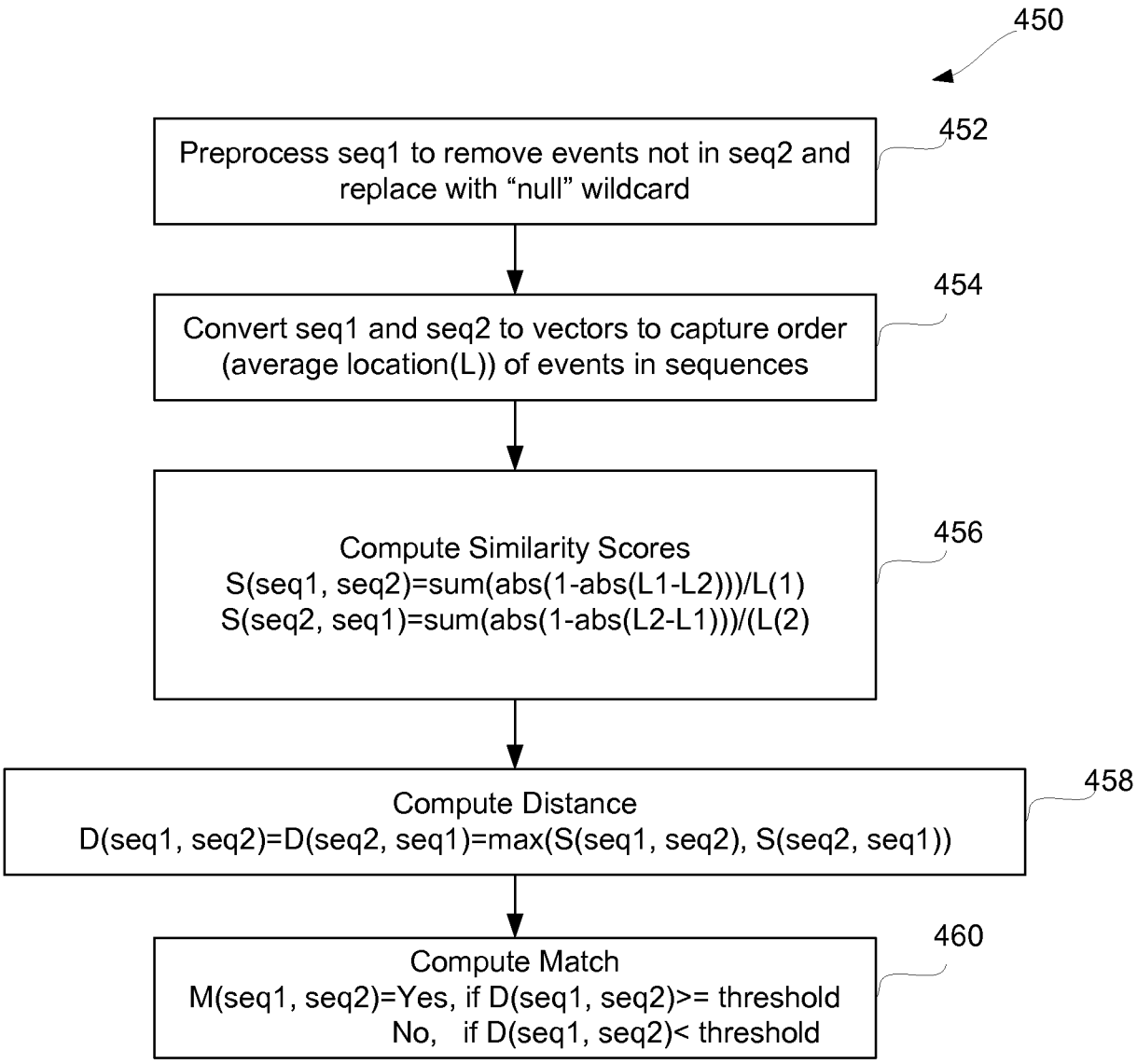
450
Preprocess seq1 to remove events not in seq2 and replace with “null” wildcard
452
Convert seq1 and seq2 to vectors to capture order (average location(L)) of events in sequences
454
Compute Similarity Scores
S(seq1, seq2)=sum(abs(1-abs(L1-L2)))/L(1)
S(seq2, seq1)=sum(abs(1-abs(L2-L1)))/(L(2)
456
Compute Distance
D(seq1, seq2)=D(seq2, seq1)=max(S(seq1, seq2), S(seq2, seq1))
458
Compute Match
M(seq1, seq2)=Yes, if D(seq1, seq2)>= threshold
No, if D(seq1, seq2)< threshold
460

Fig. 7

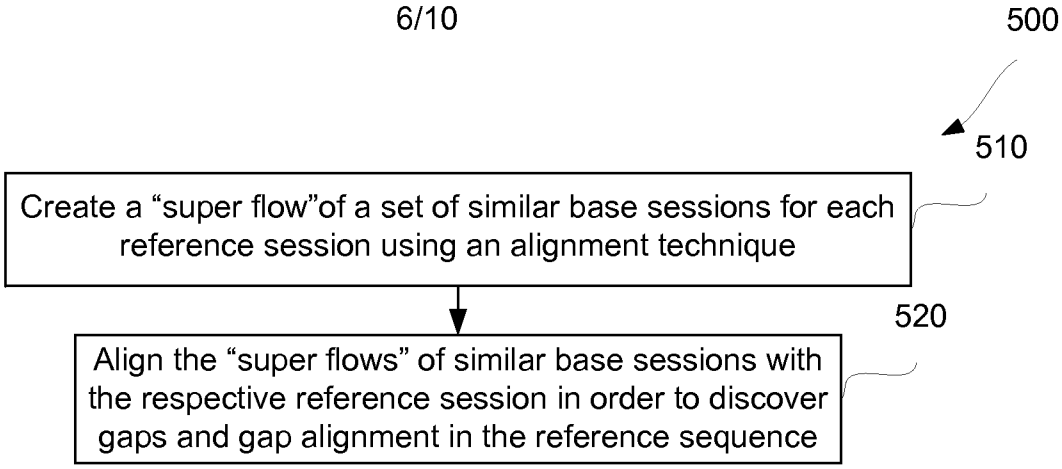
500
510
Create a “super flow”of a set of similar base sessions for each reference session using an alignment technique
520
Align the “super flows” of similar base sessions with the respective reference session in order to discover gaps and gap alignment in the reference sequence

Fig. 8

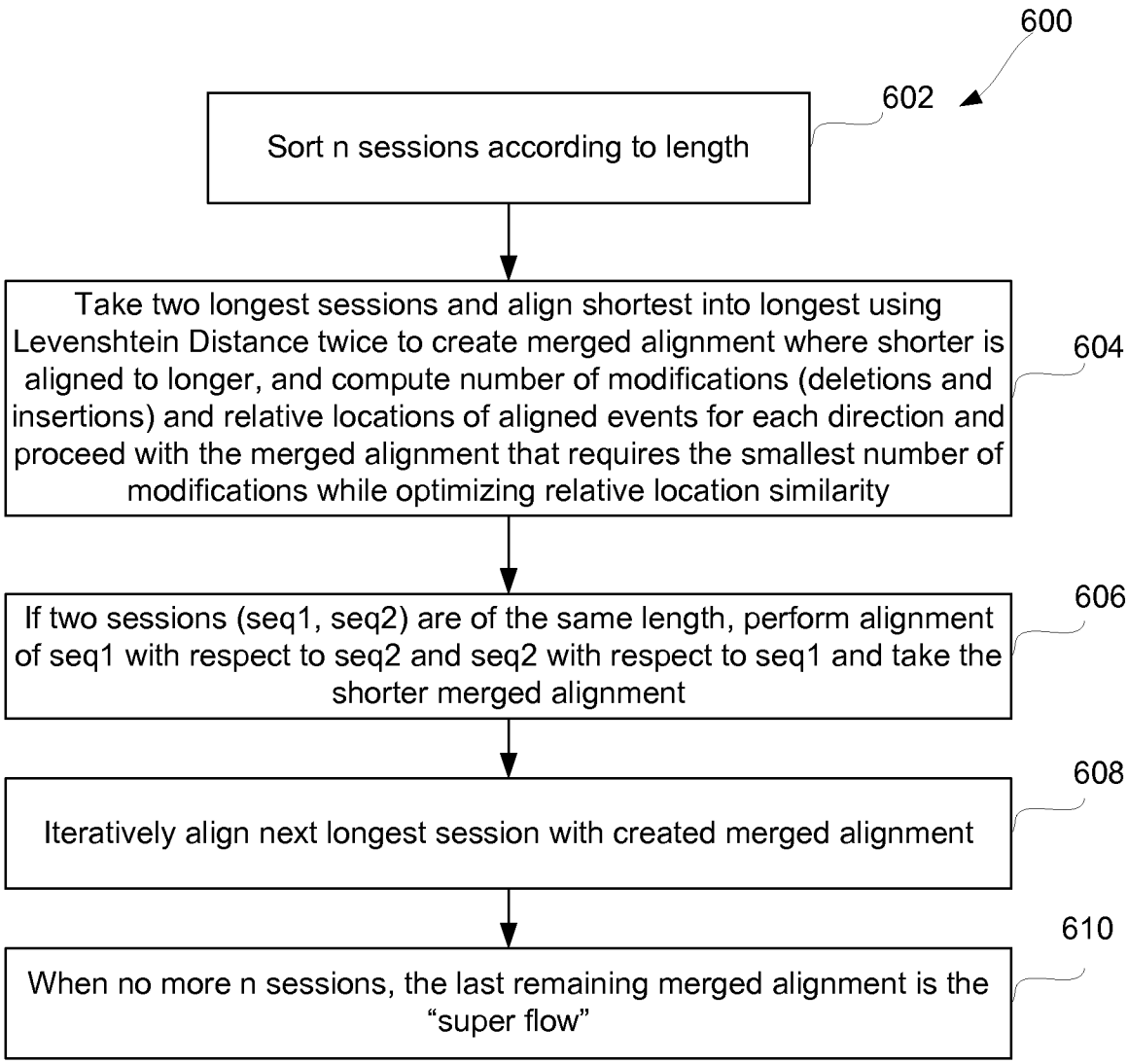
600
602
Sort n sessions according to length
Take two longest sessions and align shortest into longest using Levenshtein Distance twice to create merged alignment where shorter is aligned to longer, and compute number of modifications (deletions and insertions) and relative locations of aligned events for each direction and proceed with the merged alignment that requires the smallest number of modifications while optimizing relative location similarity
604
If two sessions (seq1, seq2) are of the same length, perform alignment of seq1 with respect to seq2 and seq2 with respect to seq1 and take the shorter merged alignment
606
Iteratively align next longest session with created merged alignment
608
When no more n sessions, the last remaining merged alignment is the “super flow”
610

Fig. 9

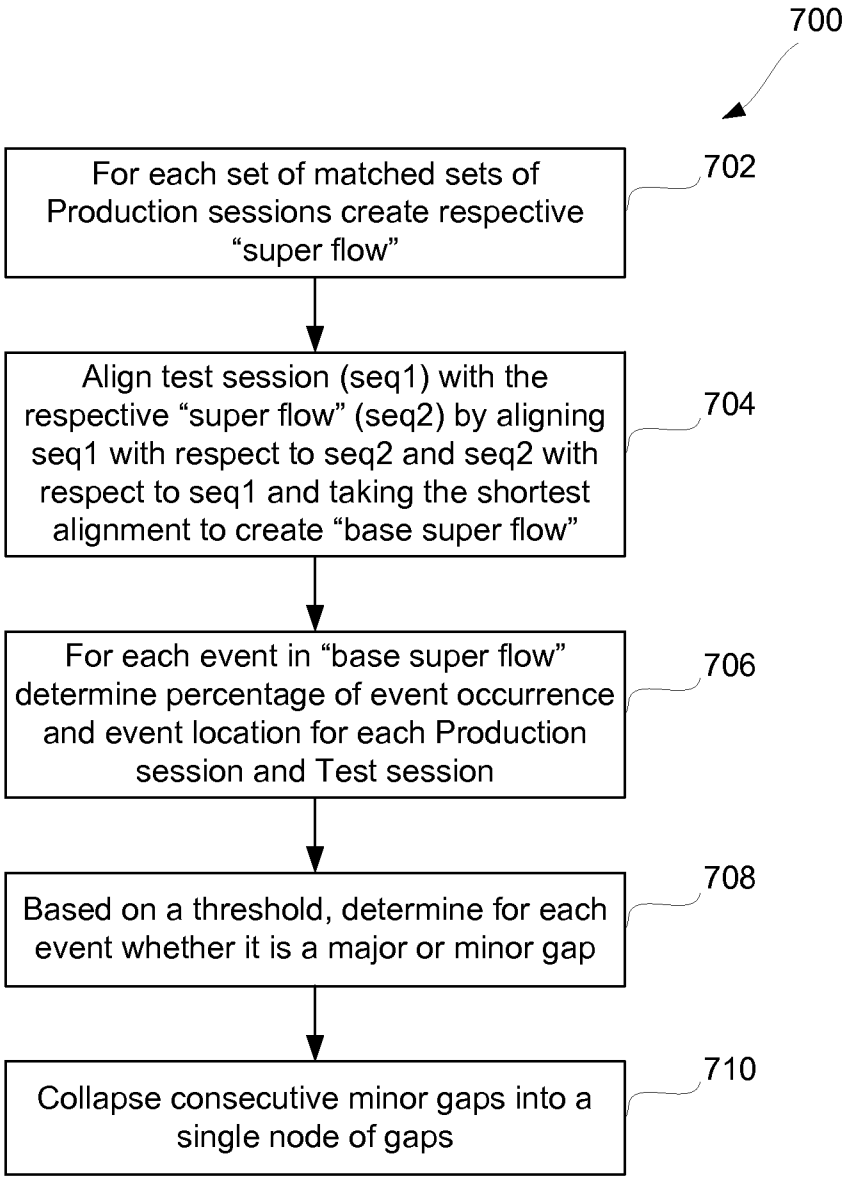
700
For each set of matched sets of Production sessions create respective "super flow"
702
Align test session (seq1) with the respective "super flow" (seq2) by aligning seq1 with respect to seq2 and seq2 with respect to seq1 and taking the shortest alignment to create "base super flow"
704
For each event in "base super flow" determine percentage of event occurrence and event location for each Production session and Test session
706
Based on a threshold, determine for each event whether it is a major or minor gap
708
Collapse consecutive minor gaps into a single node of gaps
710

Fig. 10

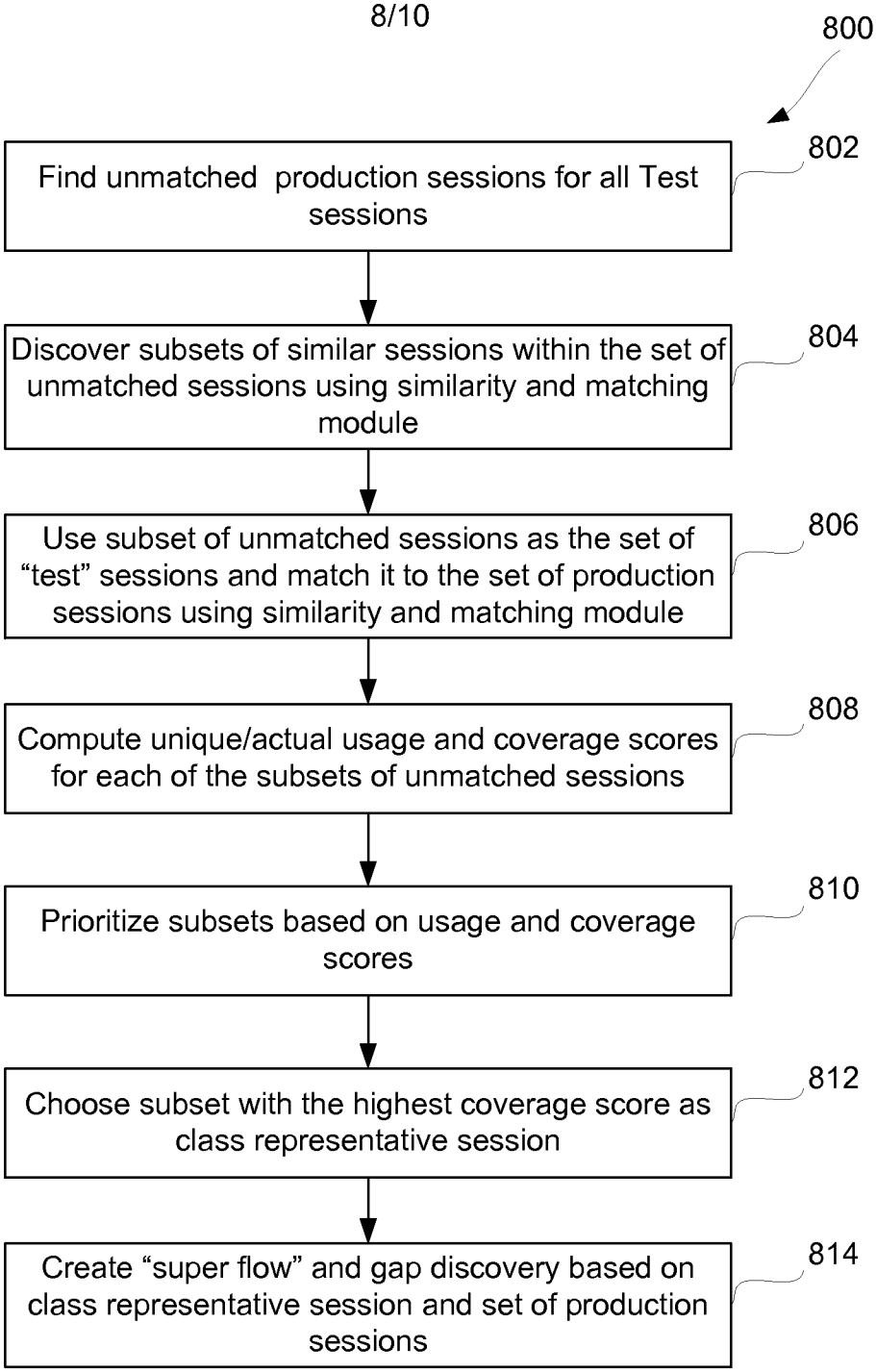
800
Find unmatched production sessions for all Test sessions
802
Discover subsets of similar sessions within the set of unmatched sessions using similarity and matching module
804
Use subset of unmatched sessions as the set of “test” sessions and match it to the set of production sessions using similarity and matching module
806
Compute unique/actual usage and coverage scores for each of the subsets of unmatched sessions
808
Prioritize subsets based on usage and coverage scores
810
Choose subset with the highest coverage score as class representative session
812
Create “super flow” and gap discovery based on class representative session and set of production sessions
814

Fig. 11

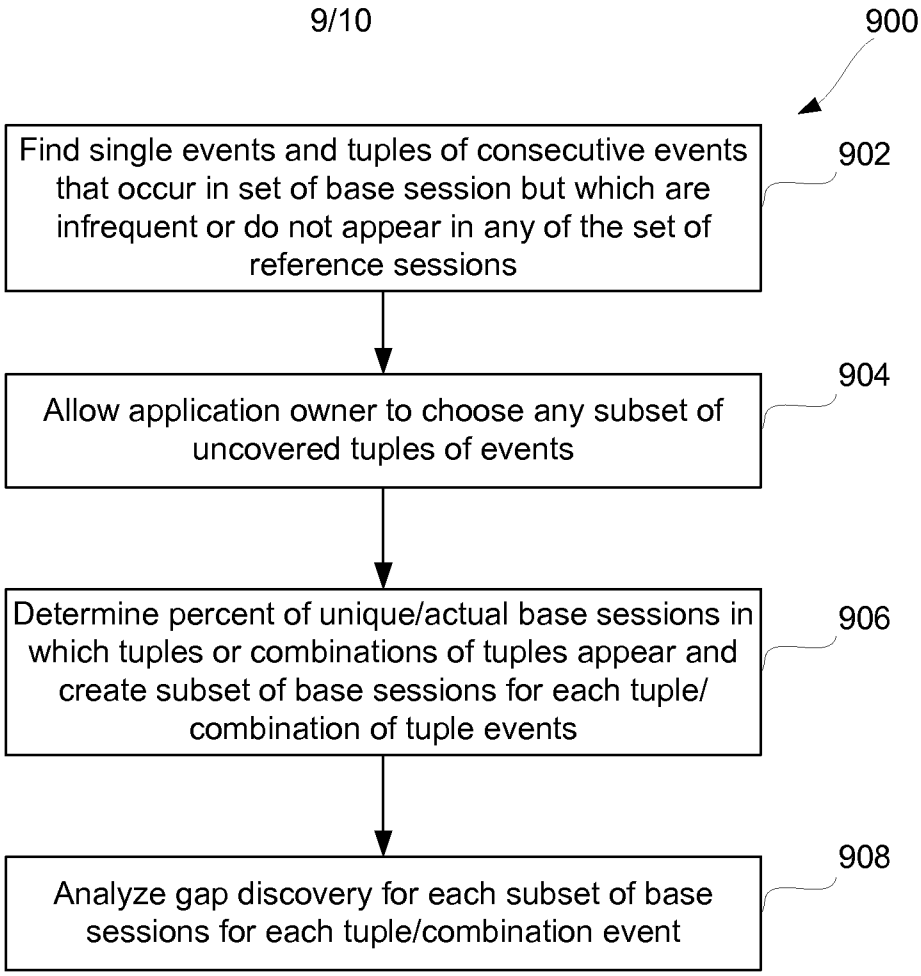
900
Find single events and tuples of consecutive events that occur in set of base session but which are infrequent or do not appear in any of the set of reference sessions
902
Allow application owner to choose any subset of uncovered tuples of events
904
Determine percent of unique/actual base sessions in which tuples or combinations of tuples appear and create subset of base sessions for each tuple/ combination of tuple events
906
Analyze gap discovery for each subset of base sessions for each tuple/combination event
908

Fig. 12

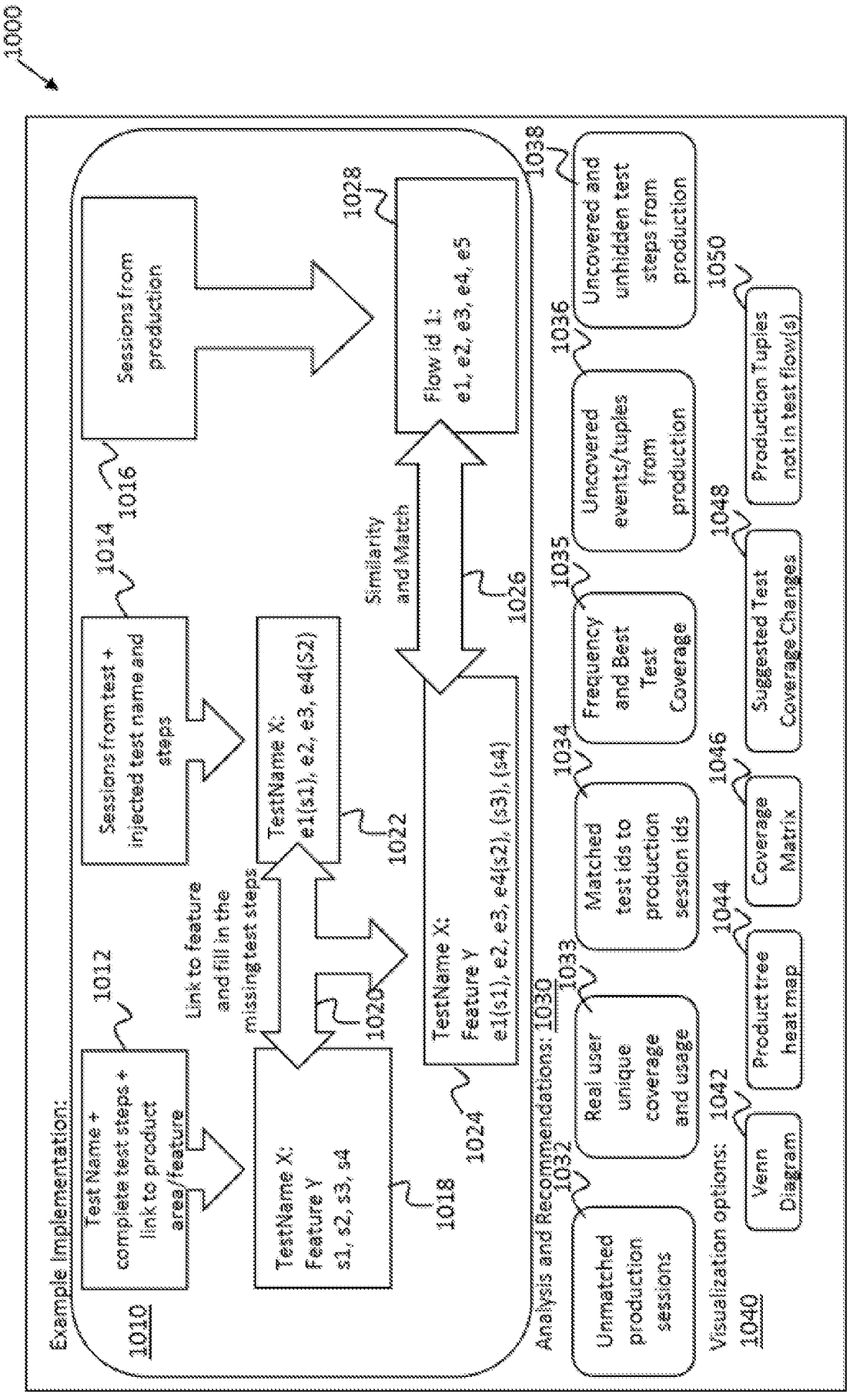
1000
Example Implementation:
1010
Test Name + complete test steps + link to product area/feature
1012
Link to feature and fill in the missing test steps
TestName X: Feature Y s1, s2, s3, s4
1018
1020
Sessions from test + injected test name and steps
1014
1016
TestName X: e1(s1), e2, e3, e4(S2)
1022
TestName X: Feature Y e1(s1), e2, e3, e4(s2), (s3), (s4)
1024
Similarity and Match
1026
Sessions from production
1028
Flow id 1: e1, e2, e3, e4, e5
1038
Analysis and Recommendations: 1030
Unmatched production sessions
1032
Real user unique coverage and usage
1033
Matched test ids to production session ids
1034
Frequency and Best Test Coverage
1035
Uncovered events/tuples from production
1036
Uncovered and unhidden test steps from production
1050
Suggested Test Coverage Changes
1048
Production Tuples not in test flow(s)
Visualization options: 1042
1040
Venn Diagram
Product tree heat map
1044
Coverage Matrix
1046

Fig. 13

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/050970

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/36(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 11/36; G06F 11/07; G06F 9/44; H04L 12/26; 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: application session analysis, base sessions, reference session, similarity score, usage score, real user coverage score, Levenshtein Distance, 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.
A	US 7,339,891 B2 (ROBERT BINDER et al.) 04 March 2008 See column 4, line 34 - column 10, line 1; claim 1; and figure 2.	1-15
A	US 2013-0290786 A1 (SHAY ARTZI et al.) 31 October 2013 See paragraphs [0031], [0043], [0047], [0118], and [0132]; and claim 12.	1-15
A	US 2013-0132933 A1 (SADAGOPAN RAJARAM et al.) 23 May 2013 See paragraphs [0001], [0015], [0029], [0033], and [0038].	1-15
A	US 2011-0131450 A1 (GRAHAM WHEELER et al.) 02 June 2011 See paragraphs [0006], [0013], [0015], [0023], and [0046].	1-15
A	US 2005-0204343 A1 (MICHAEL T. KISAMORE et al.) 15 September 2005 See paragraphs [0009], [0011], [0037], and [0056]-[0057].	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	Date of mailing of the international search report
26 February 2016 (26.02.2016)	**26 February 2016 (26.02.2016)**

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. +82-42-472-7140

Authorized officer
NHO, Ji Myong
Telephone No. +82-42-481-8528

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/050970

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 7339891 B2	04/03/2008	AU 2003-300182 A1	10/08/2004
		AU 2003-300182 A8	10/08/2004
		US 2003-0156549 A1	21/08/2003
		WO 2004-064431 A2	29/07/2004
		WO 2004-064431 A3	07/10/2004
US 2013-0290786 A1	31/10/2013	US 9135147 B2	15/09/2015
US 2013-0132933 A1	23/05/2013	None	
US 2011-0131450 A1	02/06/2011	None	
US 2005-0204343 A1	15/09/2005	US 7398469 B2	08/07/2008
		WO 2005-096153 A2	13/10/2005
		WO 2005-096153 A3	17/08/2006