



- (51) International Patent Classification: *G06F 9/44* (2006.01)    *G06F 15/16* (2006.01)
- (21) International Application Number: PCT/US2015/032065
- (22) International Filing Date: 21 May 2015 (21.05.2015)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data: 62/001,514    21 May 2014 (21.05.2014)    US
- (71) Applicant: QUANTUM FUEL SYSTEMS TECHNOLOGIES WORLDWIDE, INC. [US/US]; 25242 Artic Ocean Drive, Lake Forest, CA 92630 (US).
- (72) Inventor: ZUNIGA-HERNANDEZ, Maria, Eugenia; 25242 Artic Ocean Drive, Lake Forest, CA 92630 (US).
- (74) Agent: KRIETZMAN, Mark, H.; Washington Square, 1050 Connecticut Avenue, N.W., Suite 1100, Washington, DC 20036 (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,

[Continued on next page]

(54) Title: ENHANCED COMPLIANCE VERIFICATION SYSTEM

(57) Abstract: Methods and systems for developing and deploying software applications in a computing environment hosted by multi-user computing services platforms. Web-based user interfaces providing one or more options for accessing a software development project hosted by multi-user computing services platforms for presentation to users. Multi-user computing and network services platforms configured to receive, via the user interface, inputs to software development projects, which may include change requests or work items.

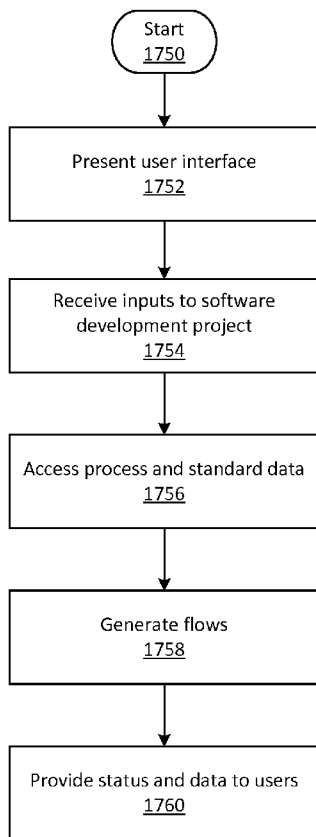
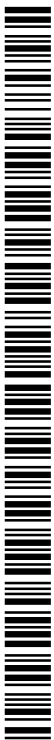


FIG. 17





SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,  
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

**(84) Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE,

DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Published:**

— *with international search report (Art. 21(3))*

## ENHANCED COMPLIANCE VERIFICATION SYSTEM

## BACKGROUND

**[0001]** Many software development projects must comply with multiple requirements. For example, a software development project may need to comply with an industry standard such as ISO 26262, a functional safety standard. Another example of a standard is ISO/IEC 15504, also known as SPICE (Software Process Improvement and Capability Determination), which is a set of standards for computer software development. Additional processes may also be implemented, such as Agile Scrum.

## DISCLOSURE

**[0002]** Disclosed herein are methods and systems for developing and deploying software applications in a computing environment hosted by a multi-user computing services platform. In some embodiments, a web-based user interface providing one or more options for accessing a software development project hosted by the multi-user computing services platform is presented to a developer user. The multi-user computing and network services platform receives, via the user interface, inputs to the software development project. The inputs may include a change request or a work item.

**[0003]** In response to receiving the inputs, data available within the multi-user computing services platform is accessed and used to verify compliance of the change request or the work item to applicable standards and development processes. A plurality of requests from the various users and developers may be processed.

**[0004]** The present disclosure provides aspects of computer-implemented methods for developing and deploying software applications comprising presenting, to a developer user of a multi-user computing platform, a user interface providing one or more options for accessing a software development project hosted by the multi-user computing services platform; receiving, by the multi-user computing services platform via the user interface, inputs to the software development project, wherein the inputs comprise at least one of a change request and a work item; in response to receiving the inputs, accessing data associated with at least one industry standard and at least one software development process; automatically generating, based on the data, one or more user actions consistent with conformance to the at least one industry standard and at least one software development process; and providing, by the multi-user computing services platform, a user interface indicative of the one or more user actions.

[0005] The present disclosure provides aspects of systems configured to develop and deploy software applications hosted by a multi-user computing services platform comprising at least one memory having stored therein computer instructions that, upon execution by one or more processors of the system, cause the system to (i) present, to a developer user of a multi-user computing platform, a user interface providing one or more options for accessing a software development project hosted by the multi-user computing services platform, (ii) receive, by the multi-user computing services platform via the user interface, inputs to the software development project, wherein the inputs comprise at least one of a change request and a work item, (iii) in response to receiving the inputs, access data associated with at least one industry standard and at least one software development process, (iv) automatically generate, based on the data, one or more user actions consistent with conformance to the at least one industry standard and at least one software development process, and (v) provide, by the multi-user computing services platform, a user interface indicative of the one or more user actions. The present disclosure provides aspects of such systems, wherein the at least one memory further comprises computer instructions that, upon execution by one or more processors of the system, cause the system to implement a developer editor configured to present a web-based user interface and a development environment.

[0006] The foregoing summary is illustrative only and is not intended to be in any way limiting. In addition to the illustrative aspects, embodiments, and features described above, further aspects, embodiments, and features will become apparent by reference to the drawings and the following detailed description.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The accompanying drawings, which are included to provide a further understanding of the disclosure, are incorporated in and constitute a part of this specification, illustrate embodiments of the disclosure and together with the detailed description serve to explain the principles of the disclosure. No attempt is made to show structural details of the disclosure in more detail than may be necessary for a fundamental understanding of the disclosure and the various ways in which it may be practiced. All reference numerals, designators, and call-outs in the figures are hereby incorporated by this reference as fully set forth herein. The failure to number an element in a figure is not intended to waive any rights, and unnumbered references may also be identified by alpha characters in the figures.

[0008] **FIG 1** illustrates a high-level system diagram in accordance with some aspects of the disclosure;

- [0009] FIG 2 illustrates a functional block diagram depicting an application development environment in accordance with some aspects of the disclosure;
- [0010] FIG 3 illustrates aspects of a software development process in accordance with some aspects of the disclosure;
- [0011] FIG 4 illustrates aspects of a scrum construction life cycle in accordance with some aspects of the disclosure;
- [0012] FIG 5 illustrates aspects of a software development workflow in accordance with some aspects of the disclosure;
- [0013] FIG 6 illustrates aspects of a user interface in accordance with some aspects of the disclosure;
- [0014] FIG 7 illustrates aspects of a change request workflow for a developer user in accordance with some aspects of the disclosure;
- [0015] FIG 8 illustrates aspects of a change request workflow for a supervisor or approver in accordance with some aspects of the disclosure;
- [0016] FIG 9 illustrates aspects of a user interface showing creation of a work item or a sub-work item in accordance with some aspects of the disclosure;
- [0017] FIG 10 illustrates aspects of a user interface showing manual creation of link associations in accordance with some aspects of the disclosure;
- [0018] FIG 11 illustrates aspects of a flow for a work item or a change request in accordance with some aspects of the disclosure;
- [0019] FIG 12 illustrates aspects of a user interface showing creation of a sub work item in accordance with some aspects of the disclosure;
- [0020] FIG 13 illustrates aspects of a requirements workflow in accordance with some aspects of the disclosure;
- [0021] FIG 14 illustrates aspects of a test case work flow in accordance with some aspects of the disclosure;
- [0022] FIG 15 illustrates aspects of a coding, calibration, defect and supporting task work flow in accordance with some aspects of the disclosure;
- [0023] FIG 16 illustrates a flowchart depicting aspects of a method for providing a software development environment in accordance with some aspects of the disclosure;
- [0024] FIG 17 illustrates aspects of an operational procedure for developing and deploying software applications in accordance with some aspects of the disclosure; and
- [0025] FIG 18 illustrates aspects of computing devices and networks in accordance with some aspects of the disclosure.

## DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

**[0026]** Aspects of the exemplary implementations of the disclosure and the various features and advantageous details thereof are explained more fully with reference to the non-limiting embodiments and examples that are described and/or illustrated in the accompanying drawings and detailed in the following description. It should be noted that the features illustrated in the drawings are not necessarily drawn to scale, and features of one embodiment may be employed with other embodiments as the skilled artisan would recognize, even if not explicitly stated herein. Descriptions of well-known components and processing techniques may be omitted so as to not unnecessarily obscure the embodiments of the disclosure. The examples used herein are intended merely to facilitate an understanding of ways in which the disclosure may be practiced and to further enable those of skill in the art to practice the embodiments of the disclosure. Accordingly, the examples and embodiments herein should not be construed as limiting the scope of the disclosure, which may further be defined by appended claims and applicable law.

**[0027]** Software development is a complex and labor intensive activity that requires rigorous processes to ensure compliance to requirements. As the scale of software development increases and because software development teams need to coordinate activities related to software projects, some parts of the software development process may be performed in a network environment to enable the use of a network of computational resources to process and coordinate the activities of multiple developers and teams. Software development solutions that focus on a typical integrated development environment provide limited direct support for coordinating software development activity. As a result, much of the software development work is implemented in an ad hoc manner with each team recreating its own process, and tools, which can lead to inefficiencies, security issues, and errors. For example, the process of bringing together various components of a software application project developed on different computers or on the same computer but at different times also can be error prone and require an inordinate amount of effort by the developers. Correct builds require complex build environments to be replicated as closely as possible on many different desktops.

**[0028]** Many software development projects are large and complex and can include a large number of interconnected devices with a mix of various types of data flowing through both virtual and physical components. Computing devices, such as servers and routers, may have complex interactions, and behaviors in one area can affect the performance of the entire computing environment.

[0029] Furthermore, many software development projects require compliance to one or more sets of requirements such as a software development process. For example, one process is the ISO/IEC 15504, also referred to as SPICE (Software Process Improvement and Capability Determination), which is a set of technical standards for the software development process. Another set of requirements is Agile Scrum which is an iterative and incremental agile software development framework for managing software projects and product or application development. Software development projects may also require compliance to one or more industry standards such as ISO 26262 which is a functional safety standard. A software development project may require compliance to any number of such standards and processes. Such standards and processes can be complex and verification of compliance may be difficult and require significant effort. This effort can grow significantly when multiple standards and processes are involved.

[0030] Thus, there is a need for an improved software development system where verification of software development activities to one or more standards and processes can be provided in an efficient and secure manner. While the examples provided herein are described in the context of software development in an automotive context, the same concerns exist for any development in an industry where a software product is developed.

[0031] To address these issues, the present disclosure describes a development environment where the process of developing and deploying a software application may be provided by an integrated development and deployment environment that is configured to facilitate the verification of compliance to one or more standards and processes.

[0032] In at least some embodiments, software applications may be developed and deployed in a computing environment hosted by a multi-user computing and network services platform. Turning now to Figure 1, a high-level system diagram of a system 100 is illustrated where one or more aspects of the disclosure may be implemented. System 100 may comprise one or more user devices 110 for viewing, editing, or otherwise accessing content developed via components of system 100, such as, for example, source code files and documentation. The user devices 110 may include a computer. The user devices 110 may communicate via one or more communication links with an application development environment 150 over a network 130. The network 130 may include a computer. Each developer or user may create different types of content if desired. The user devices 110 may be configured to reproduce the content in the form of displayed images and text. As used herein, users of the system may be referred to as users, developers, or developer users.

[0033] Figure 2 is a functional block diagram depicting application development environment 150 in greater detail. In some aspects, one or more of the components depicted in

Figure 2 may be cloud-based. Application development environment 150 may include a user interface component 202. User interface component 202 may be configured to present one or more user interfaces enabling users, such as developers, to create content, view content, interact with content, and/or other actions.

[0034] User interface component 202 may also include a developer component configured to provide an editing interface control, which may facilitate the construction of application data, logic, or any other editing related task. In some aspects, the developer component of the user interface component 202 may provide a default editor that provides a core set of action objects that can be extended, modified, and used together to define a project. For example, the developer component of the user interface component may provide an interface for developer devices 120 to create content. For example, a developer component of the user interface component 202 may be configured to present options for uploading content to be edited and to add one or more components to the content.

[0035] An analysis component 206 may also be provided for analyzing change requests and work items for compliance to a standard or development process. A storage component 208 may also be provided for storing content both during the development process and after the development is complete.

[0036] In one example of an application development environment 150, application development environment 300 may include a server application cluster and set of development modules that may provide a centralized project environment where applications can be constructed/created/programmed/edited. Application development environment 300 may include server applications which may include, for example, a cluster of application images running as instances on a virtualized infrastructure. The instances may utilize default and developer specific configuration data which may be stored on digital media available to the applications.

[0037] Environment 300 may include development tools that, in some embodiments, may include a set of software libraries and tools that a software developer may use to construct additional configuration data that defines an application as well as additional tools that might be configured to allow editing of the application's data.

[0038] In one embodiment, application development environment 300 may include an editor, which may be, for example, an editing interface configured to facilitate the construction of application data, logic, and other programming or editing related tasks.

[0039] In one embodiment, application development environment 300 may include application input/output data and source code which may comprise developer unique data and source code constructed from libraries provided by the development tools.



**[0040]** In one embodiment, application development environment 300 may include a web based tool for tracking and controlling software changes to ensure compliance to one or more standards and processes. For example, software changes may be controlled via change requests initiated by the web based tool. The web based tool may be referred to herein as verification tool or verification function. Each change request may generally include a general description of the requested change and links to work items. The work items type may include a failure modes and effects analysis (FMEA), safety goal, requirement, test case, coding task, calibration task, defect task, supporting task, and quality audit task.

**[0041]** The verification tool may be a web based platform with a database set up on a server. The bundled applications in one embodiment may include a web interface or configuration management tool, Apache server repository, and Sub Version as the revision control tool. Suitable web interfaces or configuration management tools include Polarion® WebClient for SVN, or other commercially available and open-source tools. Users of the verification tool may access the tool with a user ID and a password to access the tool. A user may be provided a user interface to open a project by selecting a project or opening a new project.

**[0042]** In one example implementation, a wiki page may be provided with at least two user visualizations. A “Home” user assignments page may display work items assigned to the current logged user. A software development page may display the project links to the software process development such as “Planning”, ”Requirements”, “Implementation”, “Testing” and “Releases.” A “Software Process and Guidelines” link may be provided in case a consultation is needed.

**[0043]** Referring to Figure 3, in one embodiment the software development process may be a combination of the Automotive SPICE V process, ISO26262 frame work, and Agile Scrum development. Working items such as change requests, FMEA, safety goals, requirements, and test cases may belong to the Automotive SPICE V process and ISO 26262 frame work. Working items such as coding tasks, calibration tasks, defect tasks, and supporting tasks may belong to the Agile Scrum development. In various embodiments, linking between the working items may be used to facilitate development in compliance to the software development processes and standards. In one example embodiment described herein, Automotive SPICE and ISO 26262 are used as standards and Agile Scrum is used as the organizational and developmental methodology to prioritize what needs to be done in each of the software development phases. It should be understood that the aforementioned standards and processes

are used to illustrate embodiments of this disclosure, and other standards and processes or combinations thereof may be implemented.

**[0044]** Agile software development is a methodology that is followed to overcome issues associated with traditional waterfall development. In Agile development, an iterative approach is used where the software project is completed in iterative phases. Each iteration delivers an incremental working version of the application. Users continually evaluate each working version / iteration of the product, and provide feedback to the development team which the developers incorporate into subsequent versions of the product. This approach provides the opportunity to account for changing business realities and also minimize large scale project/product-failure risk that can sometimes happen when using the waterfall approach to product development. Agile still uses some of the keys steps associated with waterfall development in each iteration, such as analyze, build, and test.

**[0045]** The Agile methodology is typically used in scenarios where the requirements or details of outcomes are not clear at the outset, business needs are changing rapidly, and /or are continually evolving, where testing the feasibility of an available technology to solve a problem is important, where funds to develop the product may be made available incrementally based on the proven feasibility of the product, where bringing a version of the product to market as soon as possible is more critical than having all of the bells and whistles. Self-organizing teams are typically used.

**[0046]** Scrum is a series of “sprints.” Each sprint can last typically from 2 to 4 weeks. A sprint is a complete mini-software development cycle (analyze, build, and test phase). At the end of each sprint, the customer may receive a working version of the product with new/additional functionality as compared to the previous sprint. The customer / users may test the product and provide feedback to the team.

**[0047]** Steps in a “sprint” (i.e., the single basic unit/cycle of development in Scrum) may include:

STEP 1: Sprint Planning Session

- Product Owner informs the Team what is to be included in the Sprint, which may be selected from a Product Backlog.
- The Product Backlog may include a list of high level requirements.
- The Team may determine what can be committed.
- Committed items become the Sprint Backlog and does change during the Sprint.

STEP 2: Sprint

- The sprint is the period during which the team works on building the features identified in the Sprint Backlog for the sprint.
- Daily or weekly Scrum sessions may be held to review issues / problems / roadblocks / progress / commitments.
- A Sprint Burn Down chart may be updated each day to indicate progress / completion of items. The team may review the chart to determine where and how effort should to be expended.
- The Sprint ends per the schedule regardless of whether all items/tasks are completed.

#### STEP 3: Product Release (Or Incremental Release)

- A working version of the product is released with the features committed to as part of the Sprint.

#### STEP 4: Sprint Review Meeting

- Completed work is reviewed and not completed versus committed items for the Sprint.
- Work is presented to stakeholders as a demo.

#### STEP 5: Sprint Retrospective

- Team members may review the sprint.
- What worked well and what needs improvement are noted.
- A self-corrective session (lessons learned) is conducted to incorporate process improvements in preparation for the next Sprint.

**[0048]** Working items may have their own life cycle and the transition between steps of the life cycle may be performed by a user role in the verification tool. A change request is a working item that may initiate an individual software development and may require an approval with the role “approver” in the verification tool to initiate the life cycle Approved --> Development --> Testing. Typically a software supervisor or a software lead may be assigned this role, with authorization set up by a verification tool administrator.

**[0049]** A FMEA working item may be initiated, followed by requirements and related test cases. The coding tasks, calibration tasks, and defect tasks may be created from the change request or from a requirement. A traceability function provided by the verification tool may facilitate tracking of working items and tasks. When a change request is approved, the applicable FMEAs, requirements, test cases, coding tasks, calibration tasks, and defects can also be approved and assigned according to agreements between the team members in the “backlog

meeting.” Figure 4 illustrates an example of a scrum construction life cycle. Figure 5 illustrates an example of a software development workflow. During a backlog software team meeting, the team may review the work items and define the priority, severity, time point, and the assignee for resolution. The selection of the work items may follow Agile Scrum guidelines, where the highest priority work items are selected from the backlog. The selected items become “iteration work items” to be implemented at a time point (sprint). The subtasks of the change request may be created by the assignee once the work item has been assigned to an individual with role “developer.”

**[0050]** Referring to Figure 6, an initiator of the change request may create a new working item in the project. The user can create a new work item link in the main wiki page or in the user shortcuts in the left panel. A user with access to the verification tool and to the specific project can be authorized to create a change request work item. A user interface may be provided to display the work item to highlight the fields required in each status. The allowed next status may depend on the individual role setup in the verification tool.

**[0051]** The initiator of a FMEA work item can create a change request by selecting a new work item in the main wiki page or in the user shortcuts, thereby allowing for approvals and transitions with all the work items associated with the FMEA work item. A user with access to the verification tool and to the specific project can create a FMEA work item.

**[0052]** A user may also initiate a safety goal, requirement, or test case and create or link to a change request to allow approvals and planning through a software development process such as Agile Scrum. A safety goal, requirement, or test case work item may also be created.

**[0053]** A user may be designed with the role “approver” and is provided the ability to verify change requests in status “implemented” and change the approval field with the decision taken. Once the software implementation is complete, a user with role “build manager” may proceed to check-in the final software source code to a secured repository. A user with role “quality” may collect traceability data from the working items and, if applicable, from the product software drawings. Once collected and reviewed, a baseline may be created.

**[0054]** A user with role “quality” may create a “quality” work item and perform an analysis audit by collecting data pertaining to risks and process violations, a quality score chart, and a work items trend analysis to generate an assessment of the SPICE Level.

**[0055]** When a software delivery is to be provided to a customer, the individual with role “quality” may review the change requests, tasks, defects, and traceability data from the baseline and release the software with a “Software Delivery Letter” along with executable files at a specified time point to the customer.

[0056] FIG. 7 illustrates an example of a change request workflow for a developer user. FIG. 8 illustrates an example of a change request workflow for a supervisor or approver.

[0057] The fields of a working item that is in backlog status can be modified by a user with access to the project in the verification tool. The team members of the backlog meeting may review all the change requests that are in “backlog” status. A user with role “approver” may transition the work item to “approved” status and set the new assignee, time point, and priority. The work item may be placed back into backlog status if the software development for the item has been delayed to sprint iteration.

[0058] An assignee may change the status of an item from “approved” to “under development” when the software development begins for that specific work item. The assignee may create links associated with the change request to tasks such as requirements, coding, calibration, and supporting tasks, or link a defect found in a previous software iteration.

[0059] In one embodiment, a change request may continue to be in “under development” status until the approved sub working items for the current sprint iteration are completed. When the sub-work items are completed, the current assignee of the change request may transition the status to “Done” and change the assignee to the individual with “approver” role in the verification tool.

[0060] An assignee with role “approver” may review the comments of the change request and the sub-work items associated with the change request that are applicable to the current sprint iteration. If satisfied with the implementation the assignee may set the field “approvals” to “approved”. Otherwise the user may set the status to “unapproved,” send back the change request status to “under development,” and set the assignee back to the developer.

[0061] A user with role “quality” may review the change requests with status “done” with the approvals field set either to “approved” or “unapproved” to receive or collect information that will be baselined or released to the customer. If the change request indicates “unapproved,” then the reason may be provided in a comments field.

[0062] A user with role “quality” may collect all the sub-work items associated with the current sprint, set a baseline in the verification tool, and generate related software delivery documentation.

[0063] The change request can be rejected at any time when the status is in backlog, approved, or under development and a decision is made, for example at the backlog meeting. If the rejected work item should be reopened, the item may be placed directly in the backlog to reschedule during either a different sprint iteration than the current iteration or for the current

sprint iteration. The sub-work items may be placed back in the backlog until the change request is approved.

**[0064]** A user with role “approver” may send a change request to status “quote” and set the assignee to a user with access to the verification tool. When an item is in the status, software development estimates may be analyzed and the change request may indicate the overall estimate with sub-work items. Once the quote has been accepted either by an internal or external supplier or customer then, the quote can be approved.

**[0065]** Figure 9 illustrates an example user interface showing creation of a work item or a sub-work item. Figure 10 illustrates an example user interface showing manual creation of link associations.

**[0066]** A user with access to the verification tool and to the specific project can create a change request work item and fill the fields requested either in lite or full views. To make the tool easy to use the lite view will request only the required fields according to a status or when it is transitioned to another status.

**[0067]** The tables below provide examples for severity, priority, and resolution of work items.

**Severity – How big the impact in the system is?**

Name	Description	Sort Order	Default
Must Have	Absolutely required in the system	1	FALSE
Should Have	Needed in the system	3	TRUE
Nice to Have	Could be in the system	5	FALSE
Will not Have	Not required	6	FALSE

**Priority – How fast the work item needs to be implemented?**

Name	Description	Min Value	Sort Order	Default
Highest	1st priority	90	1	FALSE
High	2nd priority	70	2	FALSE
Medium	3rd priority	50	3	TRUE
Low	4th priority	30	4	FALSE
Lowest	5th priority	10	5	FALSE

**Resolutions – How the work item was implemented?**

Name	Description	Sort Order	Default
Done	The work item is resolved	1	FALSE
Temporary	Partial implementation	2	FALSE
Won't Do	Not required or the system can live with it	3	FALSE
Duplicate	Duplicated with another work item	4	FALSE

[0068] The table below provide examples showing actions, roles, fields, and transitions.

**Actions / Transitions**

Action System Variable	Required Roles	Required Fields	Transitions
Request init	Any		
Approve approve	project_approver	assignee timePoint	From: open to: approved From: rejected to: approved From: quote to: approved
Start development start-progress	project_developer	assignee timePoint initialEstimate	From: approved to: development
Implemented resolve		resolution	From: development to: resolved
Baseline baseline	project_approver	timePoint	From: resolved to: baselined
Rework rework	project_developer	assignee timePoint	From: resolved to: development
Delayed or need information delay	project_approver	assignee	From: approved to: open From: development to: open From: rejected to: open From: quote to: open
Approve and start development quick-rework	project_approver	assignee timePoint	From: open to: development
Reject reject	project_approver	resolution	From: approved to: rejected From: development to: rejected From: open to: rejected From: quote to: rejected
Start Quote start-quote	project_approver	assignee timePoint	From: open to: quote

**[0069]** The FMEA may be a child of the change request. If there is no current change request related to the FMEA, a change request initiator may create a new change request working item. A backlog software team meeting may be conducted to review the work items and define the priority, severity, time point and the assignee for resolution. The selection of the work items may follow the Agile Scrum guidelines (or other guidelines) and select the highest priority work items from the backlog. The selected work items may be indicated as “iteration work items” to be implemented at a time point (sprint). The subtasks of the change request may be created by the assignee once the work item has been assigned to an individual with role “developer.”



**[0070]** The workflow transitions may include role setup in the account of a user in the verification tool. A FMEA work item may be created under the change request. The FMEA work item may include fields to cover the regular columns or information of the FMEA such as causes, occurrence, severity, rate levels, and controls, among others. This work item may be used as an individual FMEA if desired.

**[0071]** Figure 11 illustrates an example flow for a work item or a change request.

**[0072]** In one embodiment, the FMEA working items may be set to status “draft” as a default. The “draft” status may be used to gather all the possible failures regardless if the failures are true failures. If not a true failure, then the working item can be rejected or deleted. The fields of the working item in draft status can be modified by a user with access to the project in the verification tool. The author of the FMEA working item may be designated as the assignee by default. The designation can be changed if desired to another user or developer.

**[0073]** In one embodiment, the status can be set to “analysis” by a user with the role “developer” or “approver” to initiate actions based on the causes of failure associated with the risk. The actions may include analysis and collaboration. For example, collaboration can occur via a meeting or electronic means such as email. The comments or the attachments fields in the working item may be used to provide additional information to aid in the analysis.

**[0074]** A new assignee may be changed to a user who will perform the activities defined in the “recommended actions” field and the status may be transitioned to “Pending actions.” The assignee may create links associated with the work item such as safety goals or test cases.

**[0075]** Figure 12 illustrates an example user interface showing creation of a sub work item. Associated links may be added, and the status may be maintained until associated actions are performed and requirements or safety goals have been completed.

**[0076]** When pending actions and associated sub-working items are completed, the FMEA working item may be transitioned to the status “done” and the assignee field may be changed to the individual with role “approver.” The approver may review the actions taken and associated implementations with sub-work items, and modify the “approvals” field with the decision taken. If the item is unapproved, then the assignee field may be changed to the developer and the status may be set to “analysis.”

**[0077]** A user with role “quality” may review change request with status “done” with the approval field set either to “approved” or “unapproved” to collect the information that will be baselined or released to the customer. If the work item shows “unapproved” then the reason for keep the work item in the current release may be addressed in the comments field.

[0078] A user with role “quality” may collect sub-work items associated with the current sprint, set a baseline in the verification tool, and generate the related software delivery documentation. A change request may be rejected when, for example, the status is in backlog, approved, or under development, and a decision to reject is made (e.g., at the backlog meeting). If the rejected work item needs to be reopened, the work item may be placed directly in the backlog to reschedule in a different sprint iteration than the current iteration or the work item may be approved for the current sprint iteration. Sub-work items may be placed back in the backlog until the change request is approved.

[0079] FMEA work items may include the following custom fields.

**FMEA work item custom fields**

Name System Variable	Description	Type
Severity Rating severityRating	Failure Severity Rating	integer
Occurrence Rating occurrenceRating	Cause Occurrence Rating	integer
Detection Rating detectionRating	Control Detection Rating	integer
Risk Priority rpn	Risk Priority Number	integer
Potential Cause(s) causes	Potential Cause(s) or Source of Malfunction	Text (Multi lines)
Current Controls controls	Current Controls	Text (Multi lines)
Characteristic characteristic	Characteristic	Text (Multi lines)
Recommended Actions recommendedActions	Recommended Actions	Text (Multi lines)
Taken Actions takenActions	Actions Taken	Text (Multi lines)
New Occurrence Rating occurrenceRatingNew	New Occurrence Rating after Actions	integer
New Detection Rating detectionRatingNew	New Detection Rating after Actions	integer
New Severity Rating severityRatingNew	New Severity Rating after Actions	integer
New Risk Priority rpnNew	New Risk Priority Number after Actions	integer

[0080] Actions and transactions may include the following:

**Actions / Transitions**

Action System Variable	Required Roles	Required Fields	Transitions
Initialization init	any		
Start Analysis start-analysis	any	assignee  safetyOperMode safetyFailureCauses	From: draft to: analysis  From: resolved to: analysis  From: baselined to: analysis
Determine Safety Level plan-actions	any	safetySeverityLvl safetySeverityRationale safetyExposureLvl safetyExposureRationale safetyControlLvl safetyControlRationale safetyASIL recommendedActions	From: analysis to: action-pending
Actions Taken actions-taken	any	resolution takenActions approvals	From: action-pending to: resolved
Baseline baseline	project_approver	resolution timePoint	From: resolved to: baselined
Return to Draft return-draft	project_approver	None	From: action-pending to: draft  From: analysis to: draft  From: rejected to: draft
Return to Analysis return-analysis	project_approver  project_admin	None	From: action-pending to: analysis
Reject reject	project_approver  project_admin	resolution	From: analysis to: rejected  From: action-pending to: rejected  From: draft to: rejected

[0081] A safety goal work item may be included under an FMEA work item. The safety goal work item may contain fields to cover ISO26262 items such ASIL level, exposure, controllability, severity, and rationale taken to determine the safety level. In one embodiment, safety goal working items may be set to the status “draft” as a default. This status may be used to gather the possible safety failures regardless if those are true failures or not. If it is not a true failure, then the working item may be rejected or deleted. The fields of the working item in draft status may be modified by a user with access to the project in the verification tool. The author of

the safety goal may be designated as the assignee by default. This designation may be changed if desired to another developer or user.

**[0082]** The work item status can be set to “analysis” by a user with role “developer” or “approver” to initiate analysis of the causes of failure associated with the risk. The analysis can be performed automatically or by collaboration between one or more users (for example by meeting or by email). The comments or the attachments fields in the working item may be used to add additional information to aid the analysis. A new assignee may be changed to the individual who will perform the activities defined in the “recommended actions” field and the status may be transitioned to “Pending actions.” The assignee may create links associated to the work item such as requirements or agile tasks. Links may be added, and the status may be maintained until all the actions are performed and all the requirements or safety goals have been completed. Once the pending actions and associated sub-working items are completed, the safety goal working item may be transitioned to status “done” and the assignee field may be changed to the user with role “approver.” The approver may review the actions taken, the associated implementation with sub-work items, and modify the “approvals” field with the decision taken. If the item is unapproved then the assignee field may be changed back to the developer and the status may be set to “analysis.” The user with role “quality” or “build manager” may reviews the change request in status “done” with the approvals field set either to “approved” or “unapproved” to collect the information that will be baselined or released to the customer. If the work item shows “unapproved” then the reason for keeping the item in the current release may be addressed in the comments field. A user with role “quality” or “build manager” may collect all the sub-work items associated with the current sprint, set a baseline in the verification tool and generate the related software delivery documentation. The change request can be rejected at when the status is in backlog, approved, or under development and a decision is made, for example at the backlog meeting. If the rejected work item needs to be reopened then the item may be placed directly in the backlog to reschedule in a different sprint iteration than the current iteration or the item may be approved for the current sprint iteration.

**[0083]** Safety goal fields may include the following:

Field ID	Description
type	Work item type
title	Short description in the title
priority	multiple selection: see table below
severity	multiple selection: see table below
status	multiple selection: see table below
created	Stamp date of when the work item was created
assignee	user assigned to the work item
author	user who created the work item
plannedStart	Date
timePoint	Milestone or Sprint when the baseline will take place.
description	Description in rich text format
categories	category or categories applicable to the workitem
resolution	multiple selection: see table below
previousStatus	multiple selection: same as status table
dueDate	date of when the work item is expected
initialEstimate	Estimate in hours or days
timeSpent	Hours or days spent
remainingEstimate	Hours or days still pending in the development
attachments	Files uploaded into the work item
updated	Stamp date of when the work item was updated
plannedEnd	Date
approvals	Users with approval allowance
planningConstraints	Date restrictions
linkedWorkItems	Link to parent or child work items
hyperlinks	Link to external http sites (outside of the tool)
linkedRevisions	Source code revision in subVersion tool.
comments	Peer review field

[0084] Custom fields may include the following:

**Custom Fields**

Name System Variable	Description	Type
Potential Cause(s) causes	Potential Cause(s) or Source of Malfunction	string
Recommended Actions recommendedActions	Recommended Actions	string
Taken Actions takenActions	Actions Taken	string
Vehicle Operating Mode safetyOperMode	Power mode, parking, driving, neutral, etc.	string
Failure Effect/Hazard safetyFailure	Hazard caused	string
Severity(S) Level safetySeverityLvl	S0 - S3 Levels of ISO 26262	enum:severity
Severity(S) Rationale safetySeverityRationale	Why the severity was determined	string
Exposure(E) Level safetyExposureLvl	E0 - E4 Levels of ISO 26262	enum:exposure
Exposure(E) Rationale safetyExposureRationale	Why the exposure was determined	string
Controllability(C) Level safetyControlLvl	C0 - C3 Levels of ISO 26262	enum:controllable
Controllability(C) Rationale safetyControlRationale	How the control level was determined	string
ASIL Rating safetyASIL	ASIL Level per ISO 26262	enum:ASIL

[0085] Actions and transitions may include the following:

Actions / Transitions

Action System Variable	Required Role	Required Fields	Transitions
Initialization init	Any	None	
Start Analysis start-analysis	Any	assignee safetyOpenMode safetyFailureCauses	From: draft to: analysis From: resolved to: analysis From: baselined to: analysis
Determine Safety Level plan-actions	Any	safetySeverityLvl safetySeverityRationale safetyExposureLvl safetyExposureRationale safetyControlLvl safetyControlRationale safetyASL recommendedActions	From: analysis to: action-pending
Actions Taken actions-taken	Any	resolution takenActions approvals	From: action-pending to: resolved
Baseline baseline	project_quality project_buildManager	resolution timePoint	From: resolved to: baselined
Return to Draft return-draft	project approver	None	From: action-pending to: draft From: analysis to: draft From: rejected to: draft
Return to Analysis return-analysis	project approver	None	From: action-pending to: analysis
Reject reject	project approver	resolution	From: analysis to: rejected From: action-pending to: rejected From: draft to: rejected

[0086] A vehicle requirement may be a child of the change request or a safety goal. The user may search for an existent change request or create a new change request working item in the current project using one of the user interfaces described herein. For example, a user may create a new work item link in the main wiki page or by making a selection under the change request or under safety goal work items. Figure 13 illustrates an example requirements workflow. In one embodiment the workflow transitions may be limited to the role set up in the account of the user in the verification tool. The requirement work item may be under the change request or the safety goal work items. The requirement work item may further contain fields to

cover the type of requirement such as drivability, safety, manufacturing service, or experimental. Requirement working items may be set to the status “draft” as a default. The assignee and time point fields may be inherited from the change request or safety goal. The draft status may be used to describe the requirement or wiki requirements. If it is not a true requirement, then the working item can be unmarked or deleted. The fields of the working item in draft status can be modified by a user with access to the project in the verification tool. The author of the safety goal may be the assignee by default, which can be changed if desired to another user or developer.






**[0087]** A requirement may be created in a Wiki page by writing directly in the desired wiki using the view edit, using for example “rich text.” Links can be created to other wiki pages. The wiki page may include attachments if desired. Additionally and optionally, the wiki page may include scripts. Text may be included in the wiki as reference or commentary. A requirement ID may be displayed in the wiki as an indication of the work item. The new requirement work item may be linked back to a change request to allow processing of approvals and coverage in the software iteration.

**[0088]** The requirement may be transitioned to status “approved” when the parent change request work item is approved by the user with role “approver.” The requirement may be transitioned to status “Done” when the associated coding tasks are completed by the user with role “developer.” The requirement may be placed back to status “draft” for further modifications in the text if required. Typically the time point of a requirement may be inherited from the change request, but the time point may be changed, even when its status is “approved” or “done.” The purpose is to reuse the same requirement in a newer sprint to address a defect or another agile task related to it.

**[0089]** A user with role “quality” or “build manager” may reviews the work items with status “done” to collect the information that will be baselined or released to the customer. If the work item indicates “unapproved” then the reason may be addressed in the comments field. The user with role “quality” or “build manager” may collect all the sub-work items associated with the current sprint, set a baseline in the verification tool, and generate the related software delivery documentation. The work item may be rejected when the status is in backlog, approved or under development, and a decision is reached, for example, at the backlog meeting. If the rejected work item should be reopened then the item may be placed directly in the backlog to reschedule in a different sprint iteration than the current iteration or may be approved for the current sprint iteration. The sub-work items may go back to the backlog until the change request is approved.



[0090] Custom fields may include the following:

Name	Description	Type	Allowed Values
Req. Type reqtype	Requirement	enum:reqtype	 System Driveability  Safety Software  Safety Hardware  ODD II  Experimental

[0091] Vehicle requirement fields may include the following:

Field ID	Description
type	Work item type
title	Short description in the title
priority	multiple selection; see table below
severity	multiple selection; see table below
status	multiple selection; see table below
created	Stamp date of when the work item was created
assignee	user assigned to the work item
author	user who created the work item
plannedStart	Date
timePoint	Milestone or Sprint when the baseline will take place.
description	Description in rich text format
categories	category or categories applicable to the workitem
resolution	multiple selection; see table below
previousStatus	multiple selection; same as status table
dueDate	date of when the work item is expected
initialEstimate	Estimate in hours or days
timeSpent	Hours or days spent
remainingEstimate	Hours or days still pending in the development.
attachments	Files uploaded into the work item
updated	Stamp date of when the work item was updated
plannedEnd	Date
approvals	Users with approval allowance
planningConstraints	Date restrictions
linkedWorkItems	Link to parent or child work items
hyperlinks	Link to external http sites (outside of the tool)
linkedRevisions	Source code revision in subVersion tool.
comments	Peer review field

[0092] Actions and transitions may include the following:

**Actions / Transitions**

Action	Required Roles	Required Fields	Transitions
Approve reviewed	any	linkedWorkitems	From: draft to: approved
Mark as Implemented implemented	any		From: approved to: resolved
Delayed, back to draft make-draft	any		From: approved to: draft From: resolved to: draft From: baselined to: draft From: rejected to: draft
Quick Approve and Implement quick-implement	project_approver		From: draft to: resolved
Baseline baseline	sw_quality	timePoint	From: resolved to: baselined
Reject reject	project_approver	resolution	From: draft to: rejected From: approved to: rejected

**[0093]** Test case work items based on software change requests and software baselines may be initiated, processed, and implemented. A test case work item may be created when an initiator of the safety goal creates a new working item in a project. A backlog software team meeting may review the work items and define the priority, severity, time point, and the assignee for resolution. In one embodiment the selection of the work items may follow the Agile Scrum guidelines where the highest priority work items are selected from the backlog. The selected items are “iteration work items” to be implemented in a time point (sprint). The subtasks below the change request may be created by the assignee once the work item has been assigned to a user role “developer.” It should be understood that development models other than Agile Scrum may be implemented.

**[0094]** Figure 14 illustrates an example test case work flow. The test case working items may be set to the status “draft” as a default. This status may be used to create the test case steps and any user assigned to it with role “approver” may transition to active or inactive statuses. The “active” status may be set when a test case has been selected to a specific software iteration (sprint). The table view can be selected to modify the status in bulk. A user or assignee

with role “developer” may execute the test and fill the fields “result” and “comments” to stamp the date when the test was executed. The time point field may be used to identify the software package used on that sprint iteration.

**[0095]** When testing is finished, the approver may peer review the final testing, coverage of requirements, coding, and test cases, and modify the field “approvals” with the decision. Inactive status may be selected when the test cases are not scheduled for current software iteration (sprint) and may be reused in future iterations.

**[0096]** A user with role “quality” or “build manager” may review the test cases with status “active” with the approvals field set either to “approved” or “unapproved” to collect the information that will be baselined or released to the customer. If the work item shows “unapproved” then the reason may be addressed in the comments field. A user with role “quality” may collect the sub-work items associated with the current sprint, set a baseline in the verification tool, and generate the related software delivery documentation. The change request may be rejected when the status is in backlog, approved, or under development and the decision, for example, is taken at the backlog meeting. If the rejected work item needs to be reopened then the item may go directly to the backlog to reschedule in a different sprint iteration than the current iteration or the iteration may be approved for the current sprint iteration. The sub-work items may go back to the backlog until the change request is approved.

**[0097]** A test run is a group of test cases used to identify what will be tested per category, release, or special request. A test run is created to query the test cases in active status to be executed in the sprint. In one embodiment this can be done by selecting a “Create Test Run” button in the Testing Wiki page or individually when the test case is executed and a “test run” group is selected. The test cases may be individual entities in the verification tool and thus can be moved to another document or exist in multiple test runs or documents. A test run by category template may be selected to execute test cases for a specific feature. The test run ID may be the release version followed by the feature (category). For example: v378\_torque may be used where “v” is the version, “378” is the release number and “torque” is the category. A test run by severity “regression” template may be selected to create a test run with all the individual test cases set with severity of “regression.” This type of test run may be executed when a major change in software is to be released and a number of regression test cases may be selected by setting the status to active or inactive. Templates for test runs by other severity levels for integration, unit, or integration and basic may be provided. A template for a test run for release checkout may be selected to create a test run with a single test case or group of test cases with severity set to “release checkout.” This test run may execute an overall test to confirm the final

build prior to release to the customer. The approval of this template may be restricted to a user with role “project\_approver.” The test cases may be executed directly in a test run wiki page that may be autogenerated from the templates or individually when the work item table view is used. If the test run is used then the overall results may be displayed along with the test run approvals.

[0098] Test run fields may include the following:

**Custom Fields**

Name	Description	Type
Test Run testRun	Group of test cases	string
Test Result testResult	pass / fail result	enum:testResult
Test Comment testComment	Brief description of the result	string
Retest retest	Execute the test again	boolean
Test Type testType	System driveability, Safety Software, Safety Hardware, OBD II, Experimental or User Acceptance	enum:testType
Test Case ID testCaseID	Unique identifier	string

[0099] Test run actions/transitions may include the following:

**Actions / Transitions**

Action System Variable	Required Roles	Required Fields	Transitions
Activate activate	project_approver	testType	From: inactive to: active From: draft to: active
Deactivate deactivate	project_approver	resolution	From: active to: inactive From: draft to: inactive
execute	any	testResult testComment	To it self

[0100] Test run field IDs may include the following:

Field ID	Description
type	Work item type
title	Short description in the title
priority	multiple selection; see table below
severity	multiple selection; see table below
status	multiple selection; see table below
created	Stamp date of when the work item was created
assignee	user assigned to the work item
author	user who created the work item
plannedStart	Date
timePoint	Milestone or Sprint when the baseline will take place.
description	Description in rich text format
categories	category or categories applicable to the workitem
resolution	multiple selection; see table below
previousStatus	multiple selection; same as status table
dueDate	date of when the work item is expected
initialEstimate	Estimate in hours or days
timeSpent	Hours or days spent
remainingEstimate	Hours or days still pending in the development
attachments	Files uploaded into the work item
updated	Stamp date of when the work item was updated
plannedEnd	Date
approvals	Users with approval allowance
planningConstraints	Date restrictions
linkedWorkItems	Link to parent or child work items
hyperlinks	Link to external http sites (outside of the tool)
linkedRevisions	Source code revision in subVersion tool.
comments	Peer review field

[0101] New tasks pertaining to a particular software development standard or process may also be supported. For example, a new agile task may be initiated, processed, and implemented based on a software change request and software baselines.

[0102] The initiator of an agile task may create a new working item in the project. The initial status may be set to “backlog” as a default. During the backlog software team meeting, the work items may be reviewed and the priority, severity, time point, and the assignee for resolution may be determined. The selection of the work items may follow Agile Scrum guidelines, where the highest priority work items from the backlog may be selected as “iteration work items” to be implemented at a time point (sprint). The subtasks of the change request may be created by the assignee once the work item has been assigned to an individual with role “developer.”

[0103] Figure 15 illustrates an example coding, calibration, defect and supporting task work flow. By default, the tasks working items may be set to the status “backlog.” This status

may be used to organize the priorities of the tasks to be reviewed in the backlog meeting. A user assigned to the item with role “developer” may create tasks for the change request work items. Fields of the working item in backlog status can be modified by a user with access to the project in the verification tool. The team members at the backlog meeting may review the change requests in “backlog” status. The user with role “approver” may transition the work item to the “approved” status and set the new assignee, time point, and priority. The work item can return to the backlog status if the software development on that specific item has been delayed to sprint iteration. The status of the agile task may be changed to approved when the parent change request work item is approved by the user with role “approver” or “developer.” In the case of a defect task, the user with role “developer” can authorize the implementation but the impact or timing is typically discussed with the user with role “approver.”

**[0104]** The agile task may remain in “under development” status until the implementation is completed. The coding change may be peer reviewed and the “comments” or “attachments” field in the work item may be used for this purpose. The agile task may be linked to a requirement. If the agile task is a safety related, the task may be linked to an individual test case to ensure testing has been completed for that specific functionality. If the agile task is drivability related then the task may be linked to a group of test cases or to the overall test case for that specific sprint. An assignee with role “approver” may review the comments of the agile task and the parent-work items associated with the task applicable to the current sprint iteration. If satisfied with the implementation, the field “approvals” in the parent change request may be set to “approved,” and otherwise the field may be set to “unapproved.” in which case the change request status may be changed to “under development” and the assignee may be sent to the developer.

**[0105]** A user with role “quality” may review agile tasks and their traceability to the source code. The change requests with status “done” and the approvals field may be set either to “approved” or “unapproved” to collect the information that will be baselined or released to the customer. If the change request indicates “unapproved” then the reason for maintaining in the current release may be addressed in the comments field. A user with role “quality” may collect the work items associated with the current sprint, set a baseline in the verification tool, and generate the related software delivery documentation. The agile task may be rejected when the status is in backlog, approved, or under development and such a decision is made at the backlog meeting. If the rejected work item should be reopened then the item may be returned to the backlog to reschedule at a different sprint iteration, the item may be approved for the current

sprint iteration. Agile tasks items may be returned to the backlog until the change request is approved.

[0106] Actions and transitions may include the following:

Actions / Transitions

Action	Required Roles	Required Fields	Transitions
Request init	Any	timePoint	
Approve approve	project_approver	assignee timePoint	From: open to: approved  From: rejected to: approved
Start development start-development	Any	assignee timePoint initialEstimate	From: approved to: development
Coding done resolve	Any	resolution timeSpent	From: development to: resolved
Baseline baseline	project_approver	timePoint assignee	From: resolved to: baselined
Rework rework	Any	assignee	From: resolved to: development
Delayed or need information delay	project_approver	assignee backlogResolution	From: approved to: open  From: development to: open  From: rejected to: open
Approve and start development quick-rework	project_approver	assignee timePoint	From: open to: development
Reject reject	project_approver	resolution	From: approved to: rejected  From: development to: rejected  From: open to: rejected

[0107] Field IDs may include the following:

Field ID	Description
type	Work item type
title	Short description in the title
priority	multiple selection; see table below
severity	multiple selection; see table below
status	multiple selection; see table below
created	Stamp date of when the work item was created
assignee	user assigned to the work item
author	user who created the work item
plannedStart	Date
timePoint	Milestone or Sprint when the baseline will take place.
description	Description in rich text format
categories	category or categories applicable to the workitem
resolution	multiple selection; see table below
previousStatus	multiple selection; same as status table
dueDate	date of when the work item is expected
initialEstimate	Estimate in hours or days
timeSpent	Hours or days spent
remainingEstimate	Hours or days still pending in the development
attachments	Files uploaded into the work item
updated	Stamp date of when the work item was updated
plannedEnd	Date
approvals	Users with approval allowance
planningConstraints	Date restrictions
linkedWorkItems	Link to parent or child work items
hyperlinks	Link to external http sites (outside of the tool)
linkedRevisions	Source code revision in subVersion tool.
comments	Peer review field

[0108] In some embodiments, a software quality tool may be provided that may be implemented as a web based platform with a database and set up on a server. The bundled applications with the tool may include a web interface or configuration management tool, Apache server repository and Sub Version as the revision control tool.

[0109] In one embodiment the roles may include:



Role Name	Description of Permissions
project_user	User is granted limited access to the project. User can read all objects but not modify or create them. User can download builds and view reports, but cannot initiate builds or refresh reports.
project_admin	User granted access to Administration for the project in which the role is assigned. User has full administrator permissions within the scope of the project. User cannot access Administration in other projects where this role is not assigned him/her. User cannot access Repository Administration unless granted the role <i>admin</i> in that scope. Note that a user assigned that permission does not need to have the <i>project_admin</i> role assigned in any project.
project_developer	User is granted read/write access (excluding Administration) to the project in which the role is assigned. User can read, create and modify all objects, run and browse reports and builds. User cannot be assigned Work Items unless role <i>project_assignable</i> is also assigned.
project_assignable	Project-scope permission to have Work Items assigned.
project_quality	Project-scope permission to generate all the documentation, scripts and build automated audits.
project_approver	Project-scope permission to approve/disapprove Work Items. Users assigned this role for a project appear in the list of approvers in the Approvals section of the project's Work Items (Work Items topic, Table view; Document, Table view). That list also includes users who are assigned the global approver role. (If no users are assigned this role for the project, then only those users assigned the global approver role appear in the Approvers list in project Work Items.)

[0110] The software quality tool may use project templates to create new projects. The templates may contain dedicated workflows, system variables, scripts, wiki pages, roles, work items, quality monitoring charts as part of a specific process. For example, a template that incorporates the Capability Maturity Model (CMM), Automotive SPICE, ISO 26262, and Agile Scrum may be generated.

[0111] Figure 16 is a flowchart depicting an example of a method for providing a software development environment. As seen at 1602, the method may begin when a user, such as a developer, logs into an application development environment, such as that illustrated in Figures 1 and 2. In response to validating the user's login credentials, a user may select an option to create a new project, as illustrated at 1604. The user may also be presented with options to edit or view an existing project.

[0112] Figure 17 illustrates an example of an operational procedure for developing and deploying software applications. In one embodiment, the operational procedure may be implemented in a computing environment hosted by a multi-user computing services platform.

**[0113]** In some embodiments, a system may be implemented. The system may be configured to develop and deploy software applications in a computing environment hosted by a multi-user web services platform. The system may comprise a memory storing computer instructions that, when executed by one or more processors of the system, cause the system to implement functions such as a developer editor and a development environment.

**[0114]** A "computer," as used in this disclosure, means any machine, device, circuit, component, or module, or any system of machines, devices, circuits, components, modules, or the like, which are capable of manipulating data according to one or more instructions, such as, for example, without limitation, a processor, a microprocessor, a central processing unit, a general purpose computer, a super computer, a personal computer, a laptop computer, a palmtop computer, a smart phone, a cellular telephone, a tablet, a web-book, a notebook computer, a desktop computer, a workstation computer, a server, a cloud, or the like, or an array of processors, microprocessors, central processing units, general purpose computers, super computers, personal computers, laptop computers, palmtop computers, notebook computers, desktop computers, workstation computers, servers, or the like.

**[0115]** A "network," as used in this disclosure, means any combination of software and/or hardware, including any machine, device, circuit, component, or module, or any system of machines, devices, circuits, components, modules, or the like, which are capable of transporting signals from one location to another location, where the signals may comprise information, instructions, data, and the like. A network may include, but is not limited to, for example, at least one of a local area network (LAN), a wide area network (WAN), a metropolitan area network (MAN), a personal area network (PAN), a campus area network, a corporate area network, a global area network (GAN), a broadband area network (BAN), or the like, any of which may be configured to communicate data via a wireless and/or a wired communication medium.

**[0116]** A "server," as used in this disclosure, means any combination of software and/or hardware, including at least one application and/or at least one computer to perform services for connected clients as part of a client-server architecture. The at least one server application may include, but is not limited to, for example, an application program that can accept connections to service requests from clients by sending back responses to the clients. The server may be configured to run the at least one application, often under heavy workloads, unattended, for extended periods of time with minimal human direction. The server may include a plurality of computers configured, with the at least one application being divided among the computers depending upon the workload. For example, under light loading, the at least one application can run on a single computer. However, under heavy loading, multiple computers may be required to

run the at least one application. The server, or any of its computers, may also be used as a workstation.

**[0117]** A "communication link," as used in this disclosure, means a wired and/or wireless medium that conveys data or information between at least two points. The wired or wireless medium may include, for example, a metallic conductor link, a radio frequency (RF) communication link, an Infrared (IR) communication link, an optical communication link, or the like, without limitation. The RF communication link may include, for example, Wi-Fi, Wi-MAX, IEEE 802.11, DECT, OG, 1G, 2G, 3G, or 4G cellular standards, Bluetooth®, and the like. One or more communication links may be used in an environment 100 (shown in Figure 1) to allow sufficient data throughput and interaction between end-users (such as, e.g., agents, consumers, insurance carriers, estate planners, financial providers, web host providers, and the like). Techniques for implementing such communications links are known to those of ordinary skilled in the art.

**[0118]** In at least some embodiments, a computer that implements a portion or all of one or more of the technologies described herein may include a general purpose computer system that includes or is configured to access one or more computer-accessible media. Figure 18 illustrates such a general purpose computing device 1800. In the illustrated embodiment, computing device 1800 includes one or more processors 1810a, 1810b, and/or 1810n (which may be referred herein singularly as "a processor 1810" or in the plural as "the processors 1810") coupled to a system memory 1820 via an input/output (I/O) interface 1830. Computing device 1800 further includes a network interface 1840 coupled to I/O interface 1830.

**[0119]** In various embodiments, computing device 1800 may be a uniprocessor system including one processor 1810 or a multiprocessor system including several processors 1810 (e.g., two, four, eight or another suitable number). Processors 1810 may be any suitable processors capable of executing instructions. For example, in various embodiments, processors 1810 may be general purpose or embedded processors implementing any of a variety of instruction set architectures (ISAs), such as the x86, PowerPC, SPARC, or MIPS ISAs or any other suitable ISA. In multiprocessor systems, each of processors 1810 may commonly, but not necessarily, implement the same ISA.

**[0120]** System memory 1820 may be configured to store instructions and data accessible by processor(s) 1810. In various embodiments, system memory 1820 may be implemented using any suitable memory technology, such as static random access memory (SRAM), synchronous dynamic RAM (SDRAM), nonvolatile/Flash-type memory or any other type of memory. In the illustrated embodiment, program instructions and data implementing one

or more desired functions, such as those methods, techniques and data described above, are shown stored within system memory 1820 as code 1825 and data 1826.

**[0121]** In one embodiment, I/O interface 1830 may be configured to coordinate I/O traffic between processor 1810, system memory 1820 and any peripheral devices in the device, including network interface 1840 or other peripheral interfaces. In some embodiments, I/O interface 1830 may perform any necessary protocol, timing or other data transformations to convert data signals from one component (e.g., system memory 1820) into a format suitable for use by another component (e.g., processor 1810). In some embodiments, I/O interface 1830 may include support for devices attached through various types of peripheral buses, such as a variant of the Peripheral Component Interconnect (PCI) bus standard or the Universal Serial Bus (USB) standard, for example. In some embodiments, the function of I/O interface 1830 may be split into two or more separate components, such as a north bridge and a south bridge, for example. Also, in some embodiments some or all of the functionality of I/O interface 1830, such as an interface to system memory 1820, may be incorporated directly into processor 1810.

**[0122]** Network interface 1840 may be configured to allow data to be exchanged between computing device 1800 and other device or devices 1860 attached to a network or networks 1850, such as other computer systems or devices as illustrated in FIGS. 1 through 18, for example. In various embodiments, network interface 1840 may support communication via any suitable wired or wireless general data networks, such as types of Ethernet networks, for example. Additionally, network interface 1840 may support communication via telecommunications/telephony networks such as analog voice networks or digital fiber communications networks, via storage area networks, such as Fibre Channel SANs, or via any other suitable type of network and/or protocol.

**[0123]** In some embodiments, system memory 1820 may be one embodiment of a computer-accessible medium configured to store program instructions and data as described above for FIGS. 1-10 for implementing embodiments of the corresponding methods and apparatus. However, in other embodiments, program instructions and/or data may be received, sent or stored upon different types of computer-accessible media. Generally speaking, a computer-accessible medium may include non-transitory storage media or memory media, such as magnetic or optical media, e.g., disk or DVD/CD coupled to computing device 1800 via I/O interface 1830. A non-transitory computer-accessible storage medium may also include any volatile or non-volatile media such as RAM (e.g., SDRAM, DDR SDRAM, RDRAM, SRAM, etc.), ROM, etc., that may be included in some embodiments of computing device 1800 as system memory 1820 or another type of memory. Further, a computer-accessible medium may

include transmission media or signals such as electrical, electromagnetic or digital signals, conveyed via a communication medium such as a network and/or a wireless link, such as may be implemented via network interface 1840. Portions or all of multiple computing devices, such as those illustrated in FIGURE 18, may be used to implement the described functionality in various embodiments; for example, software components running on a variety of different devices and servers may collaborate to provide the functionality. In some embodiments, portions of the described functionality may be implemented using storage devices, network devices or special purpose computer systems, in addition to or instead of being implemented using general purpose computer systems. The term “computing device,” as used herein, refers to at least all these types of devices and is not limited to these types of devices.

**[0124]** A network set up by an entity, such as a company or a public sector organization, to provide one or more services (such as various types of cloud-based computing or storage) accessible via the Internet and/or other networks to a distributed set of clients may be termed a provider network. Such a provider network may include numerous data centers hosting various resource pools, such as collections of physical and/or virtualized computer servers, storage devices, networking equipment and the like, needed to implement and distribute the infrastructure and services offered by the provider network. The resources may in some embodiments be offered to clients in units called instances, such as virtual or physical computing instances or storage instances. A virtual computing instance may, for example, comprise one or more servers with a specified computational capacity (which may be specified by indicating the type and number of CPUs, the main memory size, and so on) and a specified software stack (e.g., a particular version of an operating system, which may in turn run on top of a hypervisor).

**[0125]** A number of different types of computing devices may be used singly or in combination to implement the resources of the provider network in different embodiments, including general purpose or special purpose computer servers, storage devices, network devices and the like. In some embodiments a client or user may be provided direct access to a resource instance, e.g., by giving a user an administrator login and password. In other embodiments the provider network operator may allow clients to specify execution requirements for specified client applications and schedule execution of the applications on behalf of the client on execution platforms (such as application server instances, Java™ virtual machines (JVMs), general purpose or special purpose operating systems, platforms that support various interpreted or compiled programming languages such as Ruby, Perl, Python, C, C++, and the like, or high-performance computing platforms) suitable for the applications, without, for example, requiring the client to access an instance or an execution platform directly. A given execution platform may utilize one

or more resource instances in some implementations; in other implementations multiple execution platforms may be mapped to a single resource instance.

**[0126]** The terms "including," "comprising," "having," and variations thereof, as used in this disclosure, mean "including, but not limited to," unless expressly specified otherwise.

**[0127]** The terms "a," "an," and "the," as used in this disclosure, means "one or more," unless expressly specified otherwise.

**[0128]** Devices that are in communication with each other need not be in continuous communication with each other, unless expressly specified otherwise. In addition, devices that are in communication with each other may communicate directly or indirectly through one or more intermediaries.

**[0129]** Although process steps, method steps, algorithms, or the like may be described in a sequential order, such processes, methods and algorithms may be configured to work in alternate orders. In other words, any sequence or order of steps that may be described does not necessarily indicate a requirement that the steps be performed in that order. The steps of the processes, methods or algorithms described herein may be performed in any order practical. Further, some steps may be performed simultaneously.

**[0130]** When a single device or article is described herein, it will be readily apparent that more than one device or article may be used in place of a single device or article. Similarly, where more than one device or article is described herein, it will be readily apparent that a single device or article may be used in place of the more than one device or article. The functionality or the features of a device may be alternatively embodied by one or more other devices which are not explicitly described as having such functionality or features.

**[0131]** A "computer-readable medium," as used in this disclosure, means any medium that participates in providing data (for example, instructions) which may be read by a computer. Such a medium may take many forms, including non-volatile media, volatile media, and transmission media. Non-volatile media may include, for example, optical or magnetic disks and other persistent memory. Volatile media may include dynamic random access memory (DRAM). Transmission media may include coaxial cables, copper wire and fiber optics, including the wires that comprise a system bus coupled to the processor. Transmission media may include or convey acoustic waves, light waves and electromagnetic emissions, such as those generated during radio frequency (RF) and infrared (IR) data communications. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, any other magnetic medium, a CD-ROM, DVD, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, an EPROM, a

FLASH-EEPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

**[0132]** Various forms of computer-readable media may be involved in carrying sequences of instructions to a computer. For example, sequences of instruction (i) may be delivered from a RAM to a processor, (ii) may be carried over a wireless transmission medium, and/or (iii) may be formatted according to numerous formats, standards or protocols, including, for example, Wi-Fi, Wi-MAX, IEEE 802.11, DECT, OG, IG, 2G, 3G, or 4G cellular standards, Bluetooth®, or the like.

**[0133]** The present disclosure is not to be limited in terms of the particular embodiments described in this application, which are intended as illustrations of various aspects. Many modifications and variations can be made without departing from its spirit and scope, as will be apparent to those skilled in the art. Functionally equivalent methods and apparatuses within the scope of the disclosure, in addition to those enumerated herein, will be apparent to those skilled in the art from the foregoing descriptions. Such modifications and variations are intended to fall within the scope of the appended claims. The present disclosure is to be limited only by the terms of the appended claims, along with the full scope of equivalents to which such claims are entitled. It is to be understood that this disclosure is not limited to particular methods, reagents, compounds, compositions or biological systems, which can, of course, vary. It is also to be understood that the terminology used herein is for the purpose of describing particular embodiments only, and is not intended to be limiting.

**[0134]** There is little distinction left between hardware and software implementations of aspects of systems; the use of hardware or software is generally (but not always, in that in certain contexts the choice between hardware and software can become significant) a design choice representing cost vs. efficiency tradeoffs. There are various vehicles by which processes and/or systems and/or other technologies described herein can be effected (e.g., hardware, software, and/or firmware), and that the preferred vehicle will vary with the context in which the processes and/or systems and/or other technologies are deployed. For example, if an implementer determines that speed and accuracy are paramount, the implementer may opt for a mainly hardware and/or firmware vehicle; if flexibility is paramount, the implementer may opt for a mainly software implementation; or, yet again alternatively, the implementer may opt for some combination of hardware, software, and/or firmware.

**[0135]** One skilled in the art will appreciate that, for this and other processes and methods disclosed herein, the functions performed in the processes and methods may be implemented in differing order. Furthermore, the outlined steps and operations are only

provided as examples, and some of the steps and operations may be optional, combined into fewer steps and operations, or expanded into additional steps and operations without detracting from the essence of the disclosed embodiments.

**[0136]** While the disclosure has been described in terms of exemplary embodiments, those skilled in the art will recognize that the disclosure can be practiced with modifications in the spirit and scope of the appended claims. These examples given above are merely illustrative and are not meant to be an exhaustive list of all possible designs, embodiments, applications or modifications of the disclosure.

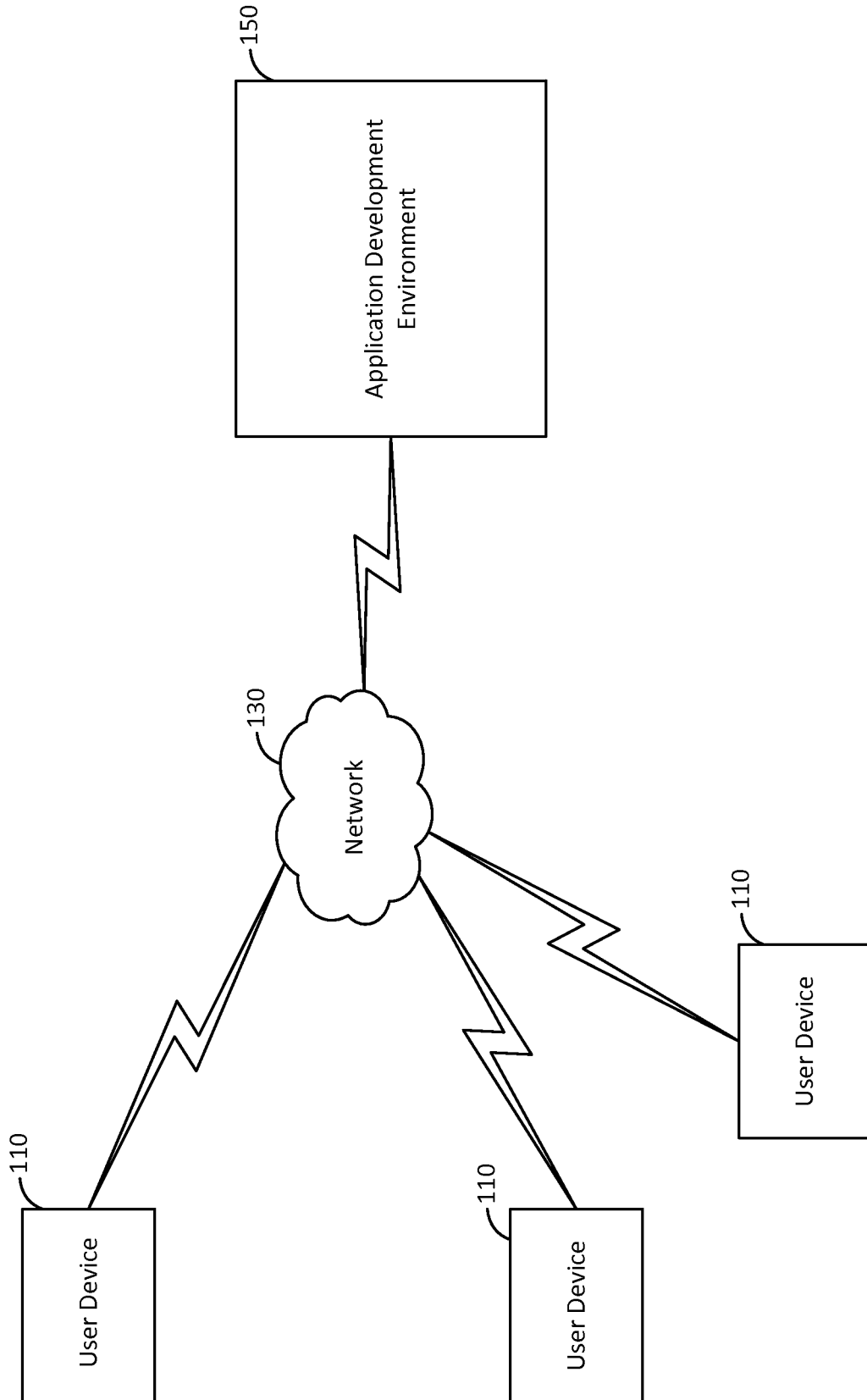


**Claims:**

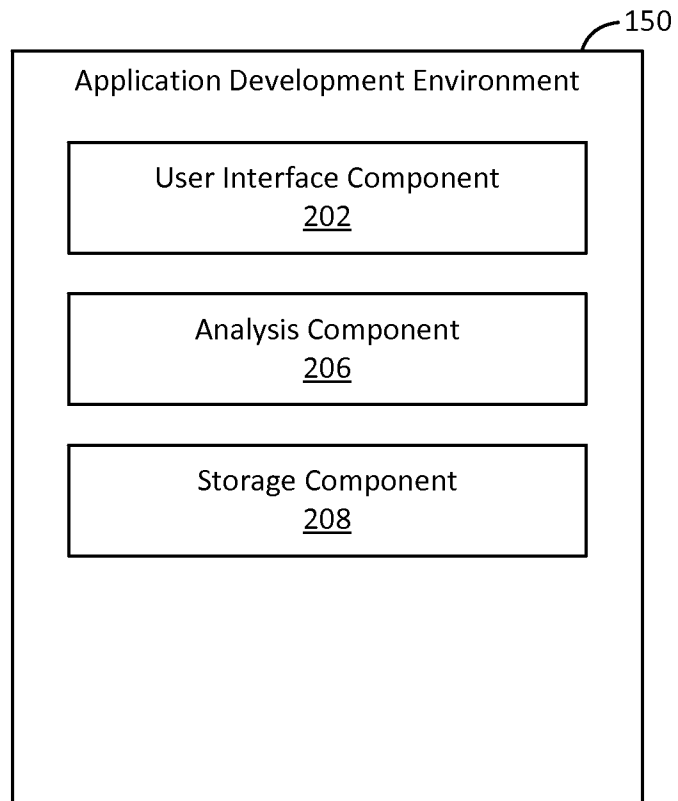
1. A computer-implemented method for developing and deploying software applications comprising:  
presenting, to a developer user of a multi-user computing platform, a user interface providing one or more options for accessing a software development project hosted by the multi-user computing services platform;  
receiving, by the multi-user computing services platform via the user interface, inputs to the software development project, wherein the inputs comprise at least one of a change request and a work item;  
in response to receiving the inputs, accessing data associated with at least one industry standard and at least one software development process;  
automatically generating, based on the data, one or more user actions consistent with conformance to the at least one industry standard and at least one software development process; and  
providing, by the multi-user computing services platform, a user interface indicative of the one or more user actions.
2. The method of claim 1, wherein the at least one industry standard comprises one or more of ISO 26262, ISO/IEC 15504, and Automotive SPICE.
3. The method of claim 1, wherein the at least one software development process comprises one or more of Agile Scrum and the Capability Maturity Model.
4. The method of claim 1, wherein the one or more user actions comprises one or more of request, approve, baseline, start development, rework, and reject.
5. A system configured to develop and deploy software applications hosted by a multi-user computing services platform comprising  
at least one memory having stored therein computer instructions that, upon execution by one or more processors of the system, cause the system to:  
present, to a developer user of a multi-user computing platform, a user interface providing one or more options for accessing a software development project hosted by the multi-user computing services platform;  
receive, by the multi-user computing services platform via the user interface, inputs to the software development project, wherein the inputs comprise at least one of a change request and a work item;  
in response to receiving the inputs, access data associated with at least one industry standard and at least one software development process;

automatically generate, based on the data, one or more user actions consistent with conformance to the at least one industry standard and at least one software development process; and provide, by the multi-user computing services platform, a user interface indicative of the one or more user actions.

6. The system of claim 5, wherein the at least one memory further comprises computer instructions that, upon execution by one or more processors of the system, cause the system to implement a developer editor configured to present a web-based user interface and a development environment.
7. The system of claim 5, wherein the at least one industry standard comprises one or more of ISO 26262, ISO/IEC 15504, and Automotive SPICE.
8. The system of claim 5, wherein the at least one software development process comprises one or more of Agile Scrum and the Capability Maturity Model.
9. The system of claim 5, wherein the one or more user actions comprises one or more of request, approve, baseline, start development, rework, and reject.



**FIG. 1**



**FIG. 2**

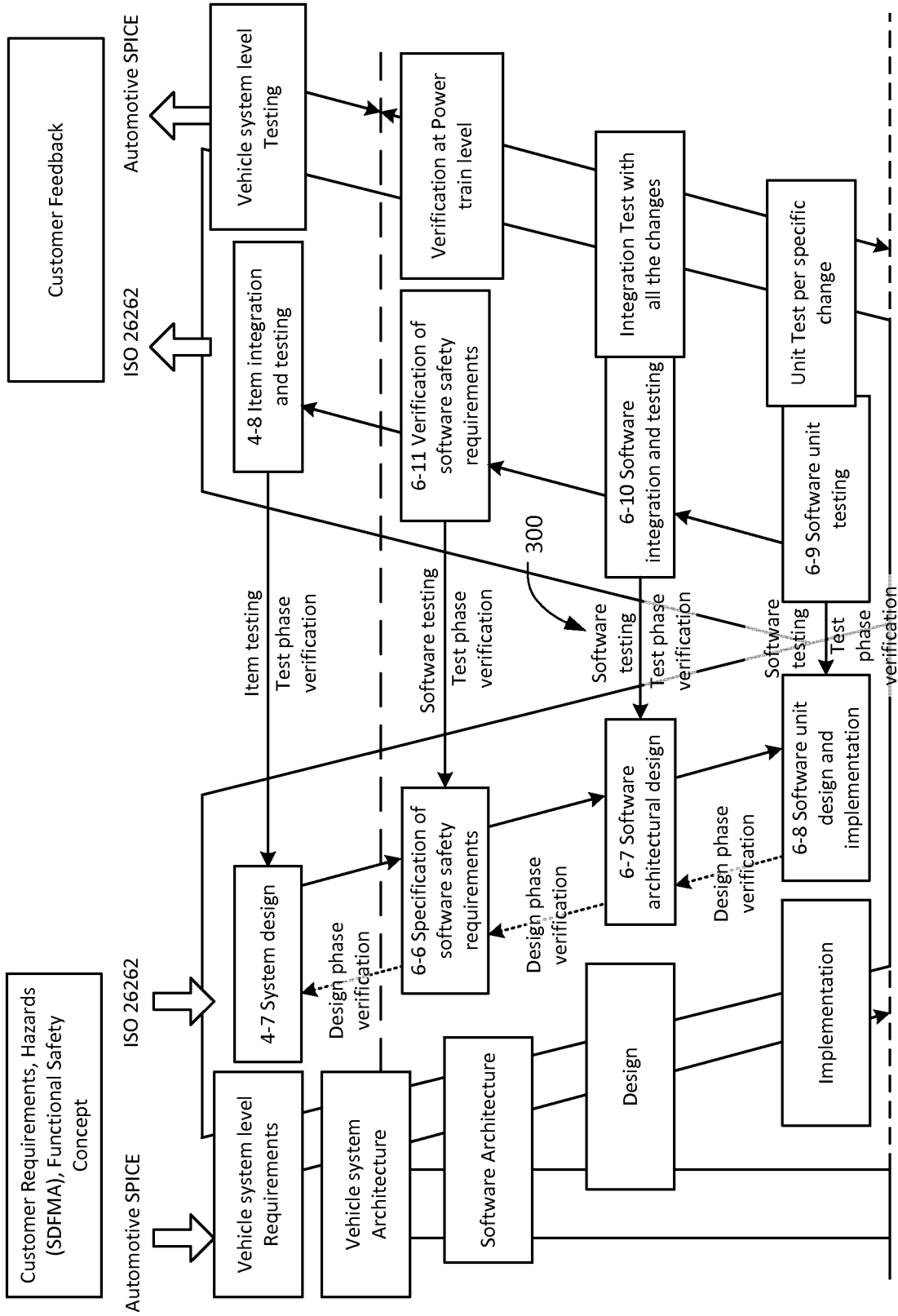


FIG. 3

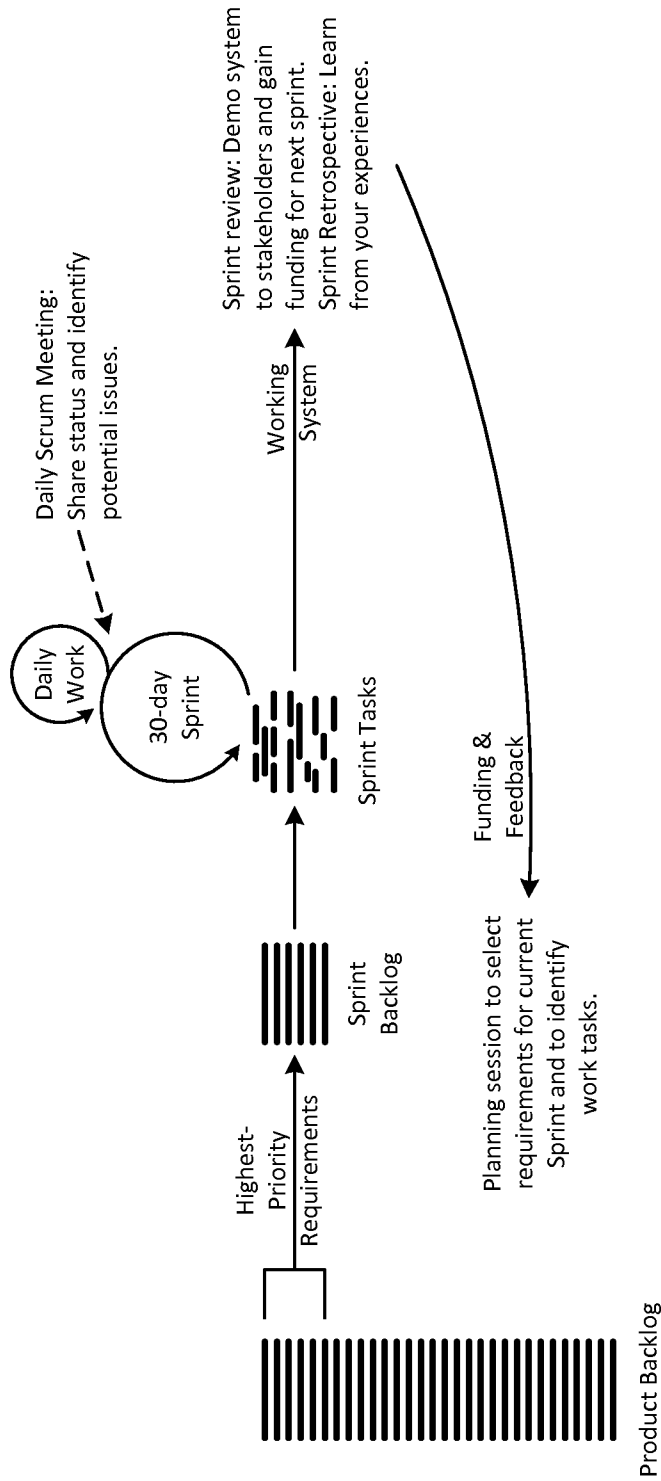


FIG. 4

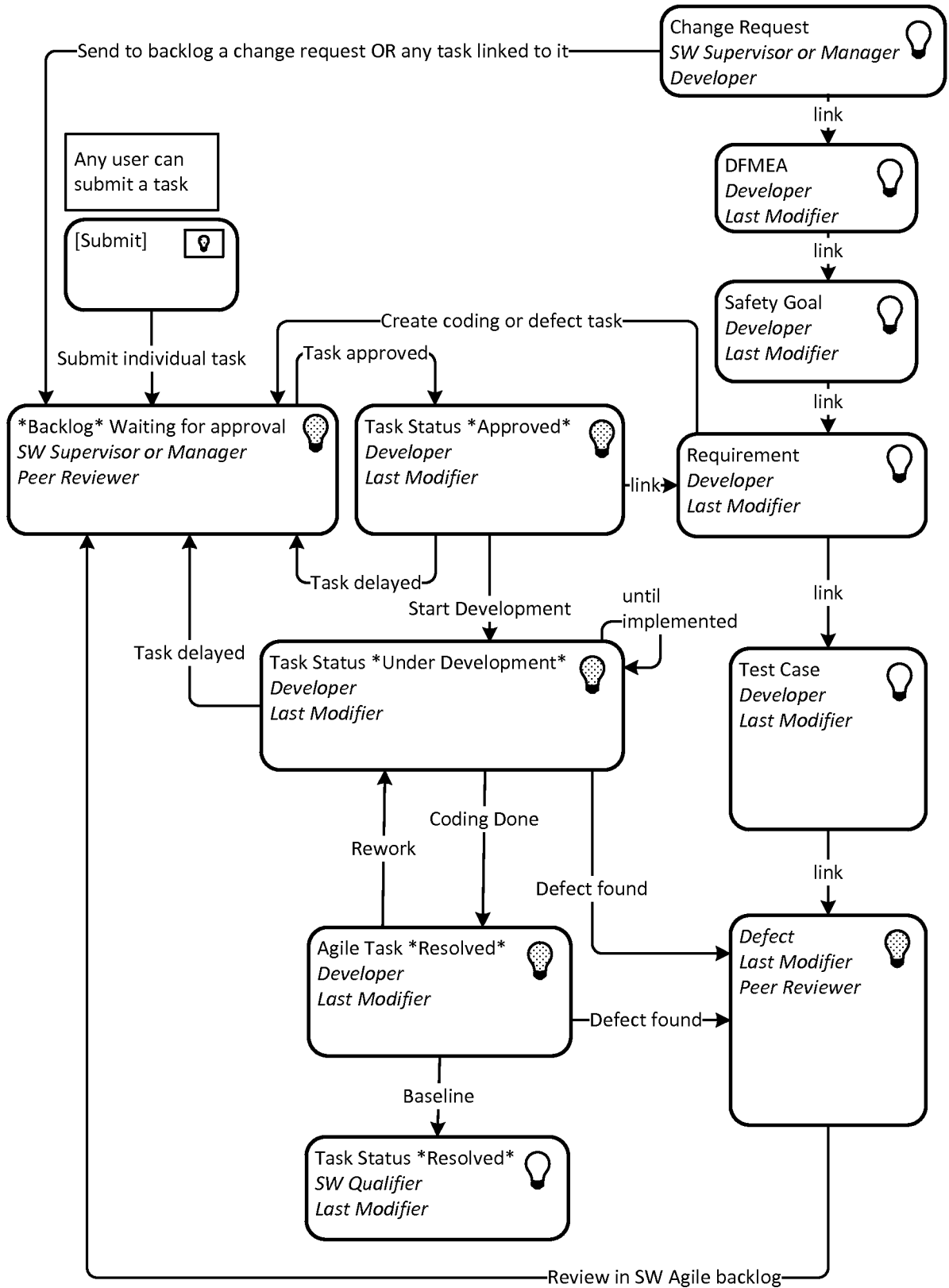


FIG. 5

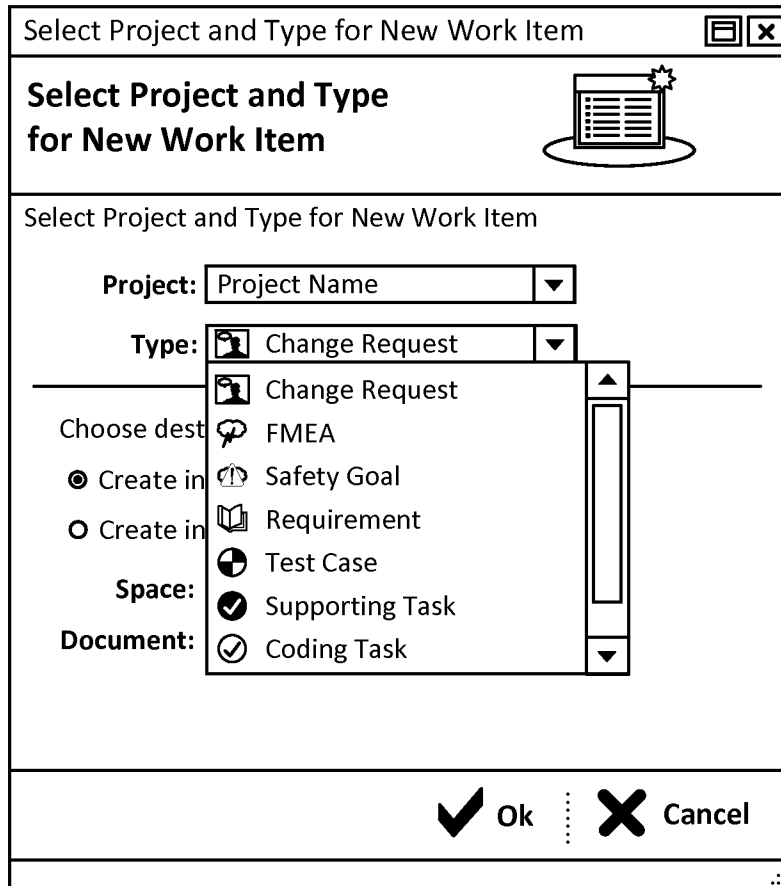


FIG. 6



7/18

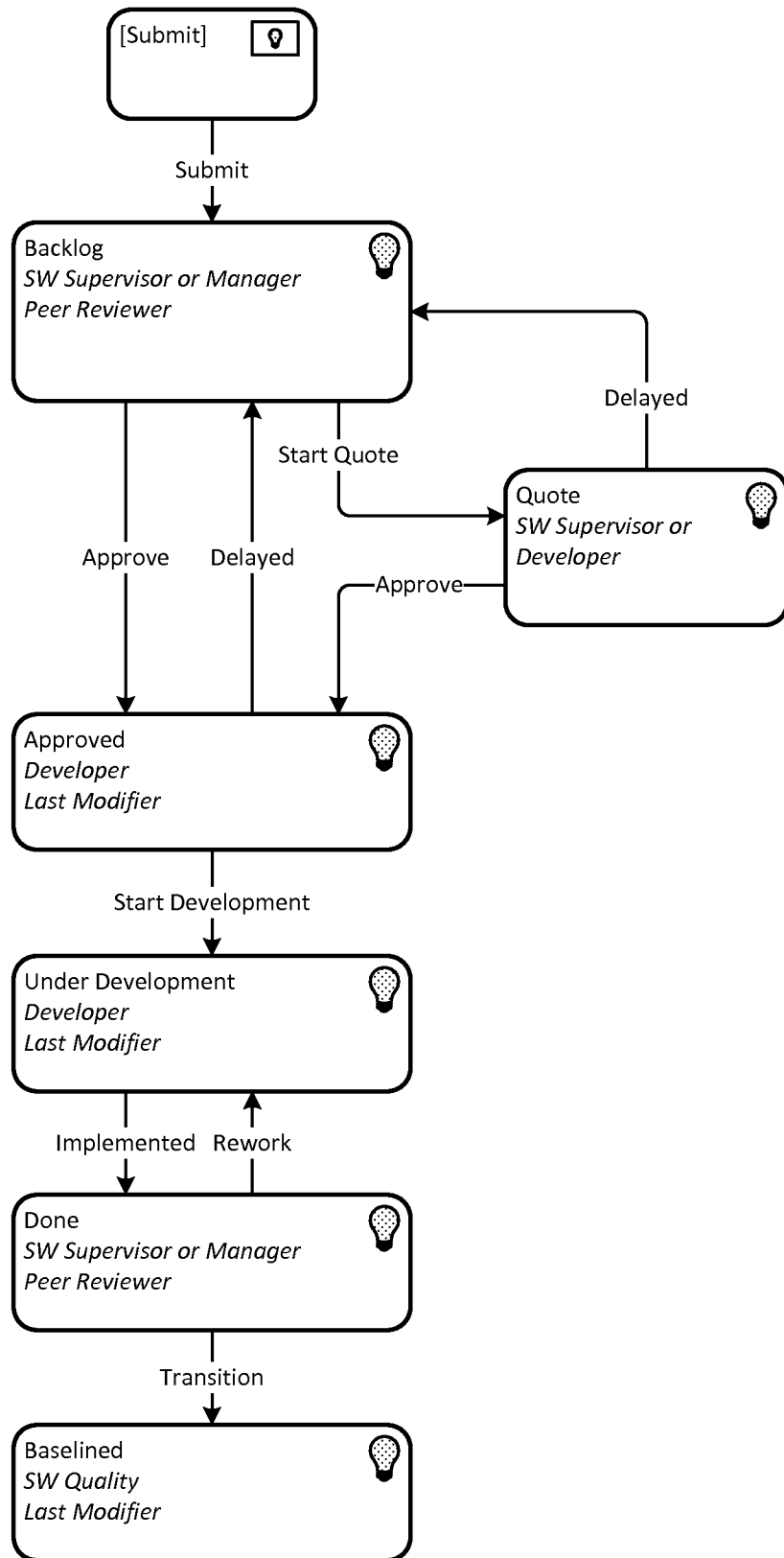


FIG. 7

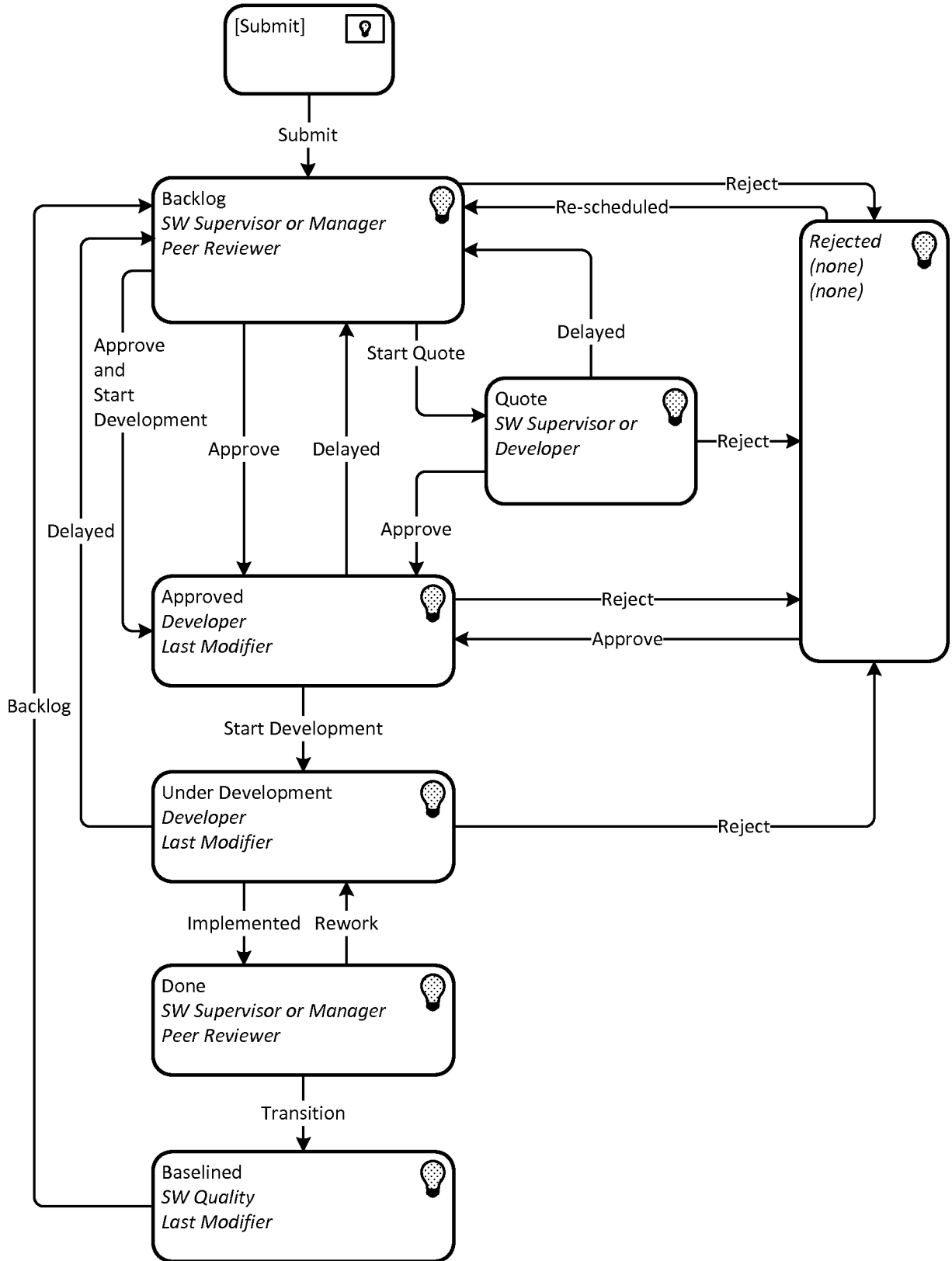







FIG. 8


↑ .....  **Fxxxx-261 – Sample – changerequest scripts and est**


↑ .....  Fxxxx-330

↑ .....  Fxxxx-262 

.....









**Type:**  Change Request

**Status:**  Backlog

**Backlog Resolution:**  Need information

**Assignee:** Jane Doe

**Description**

-  Implement with SW Coding Task
-  Implement with Calibration
-  Analyze/Document prior to implement
-  Create Defect (bug found)
-  Create FMEA
-  Create Safety Goal
-  Create a Requirement
-  Create a Derived Change Request

**FIG. 9**







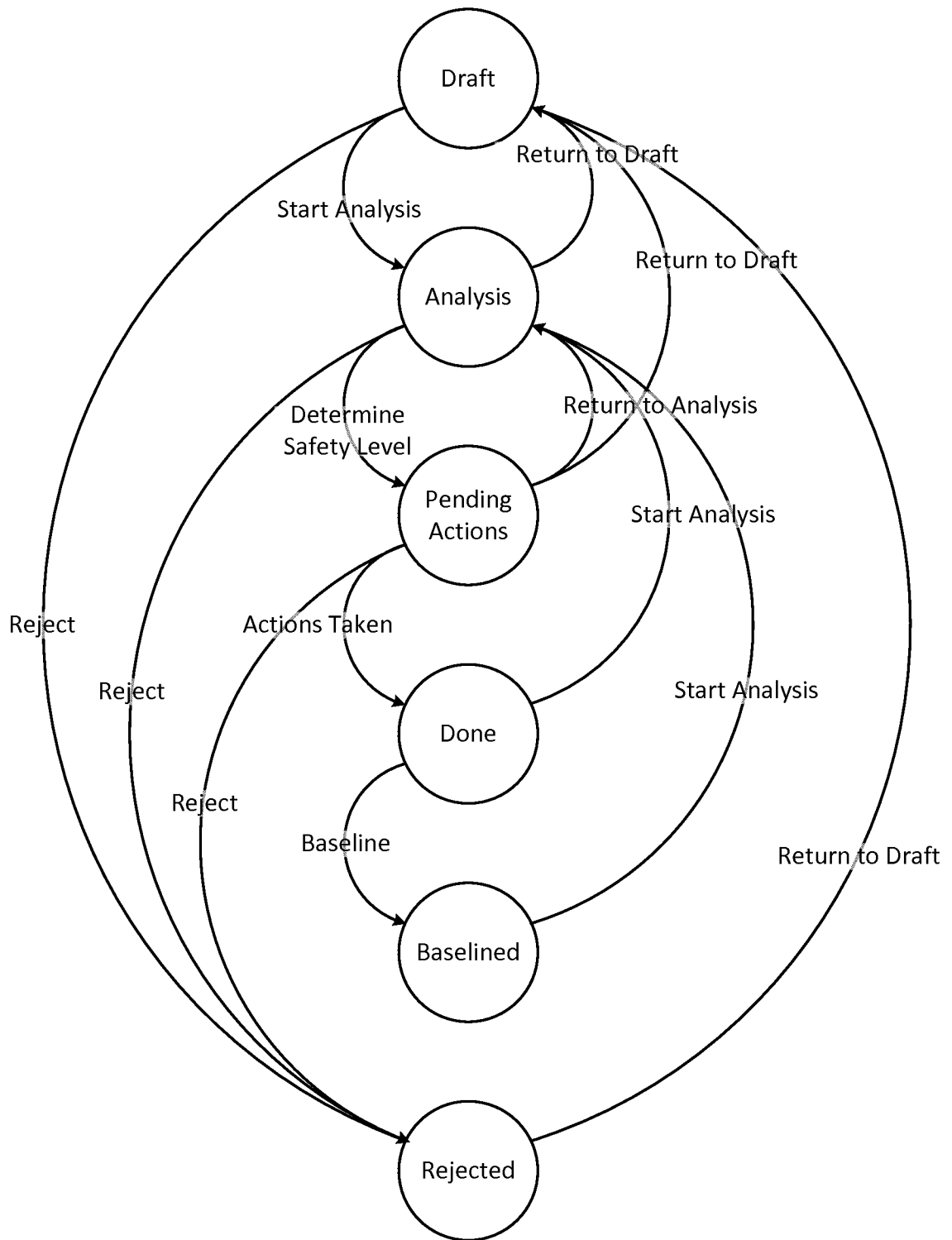
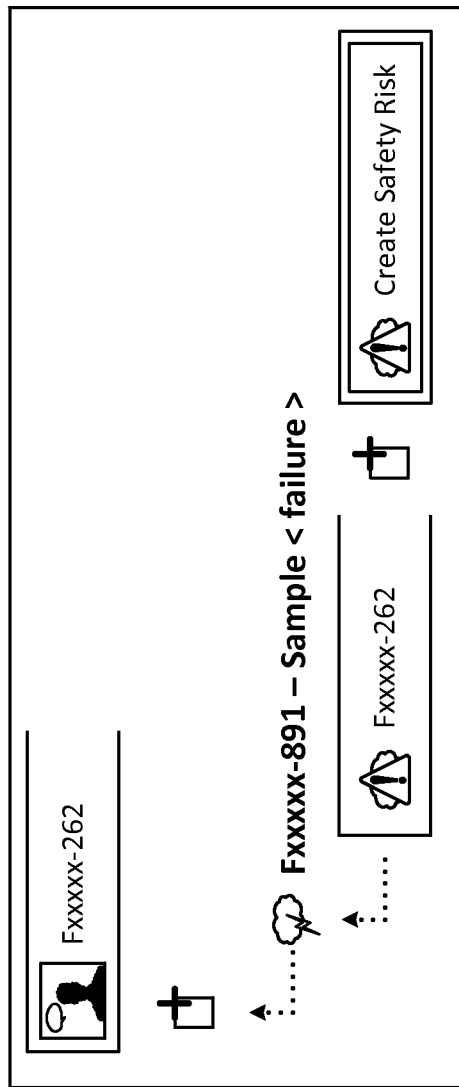
Linked Work Items							
Suspect	Role	Title	Project	Revision	Status	Assignee	Actions
<input type="checkbox"/>	implements	 Fxxx-2025 – Process DEMO	MY11 Smith	Head			
<input type="checkbox"/>	implements	SAMPLE-2  - This is a sample of a non-functional requirement.	MY11 Smith	Head	 Draft		

FIG. 10

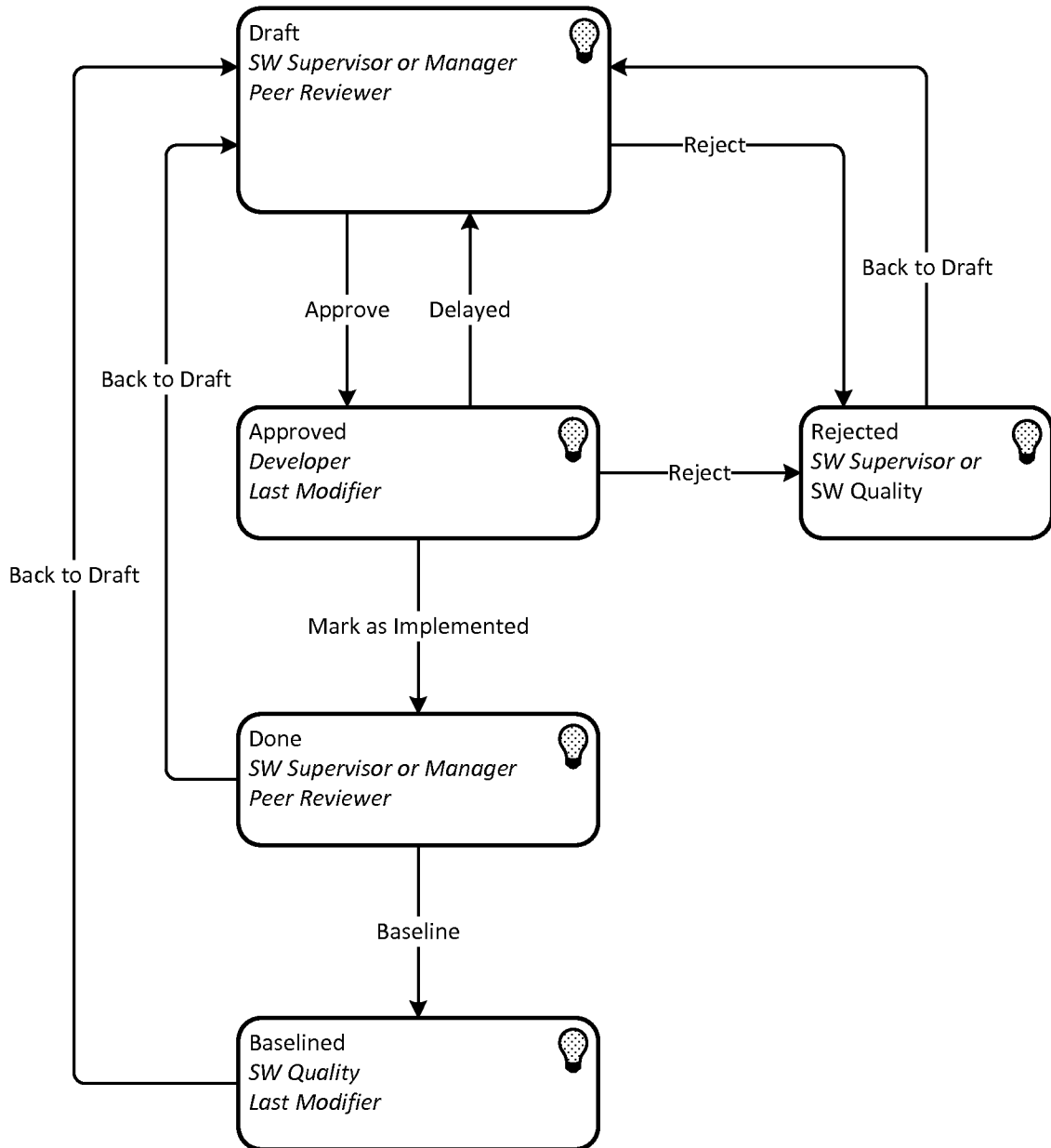
11/18



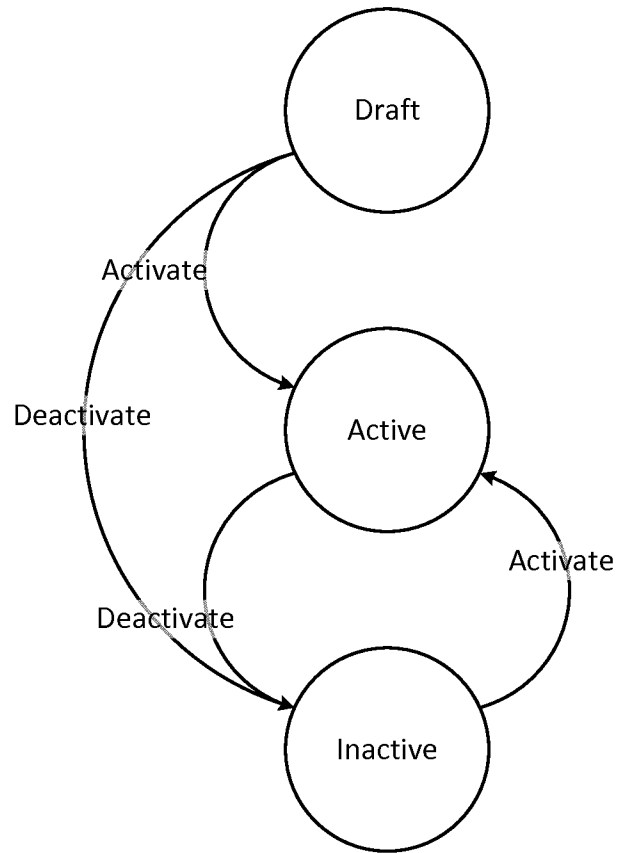
**FIG. 11**



**FIG. 12**



**FIG. 13**



**FIG. 14**



15/18

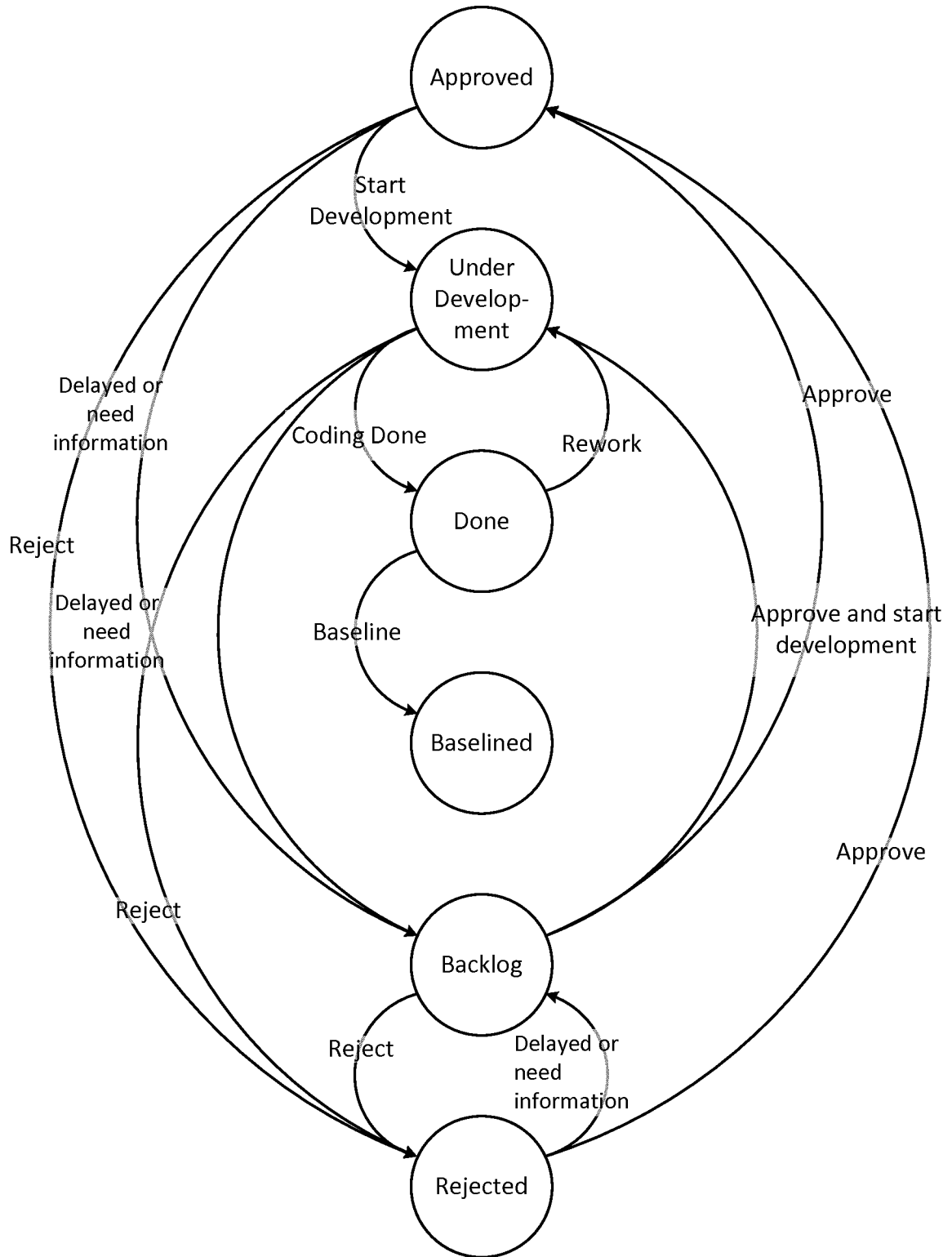
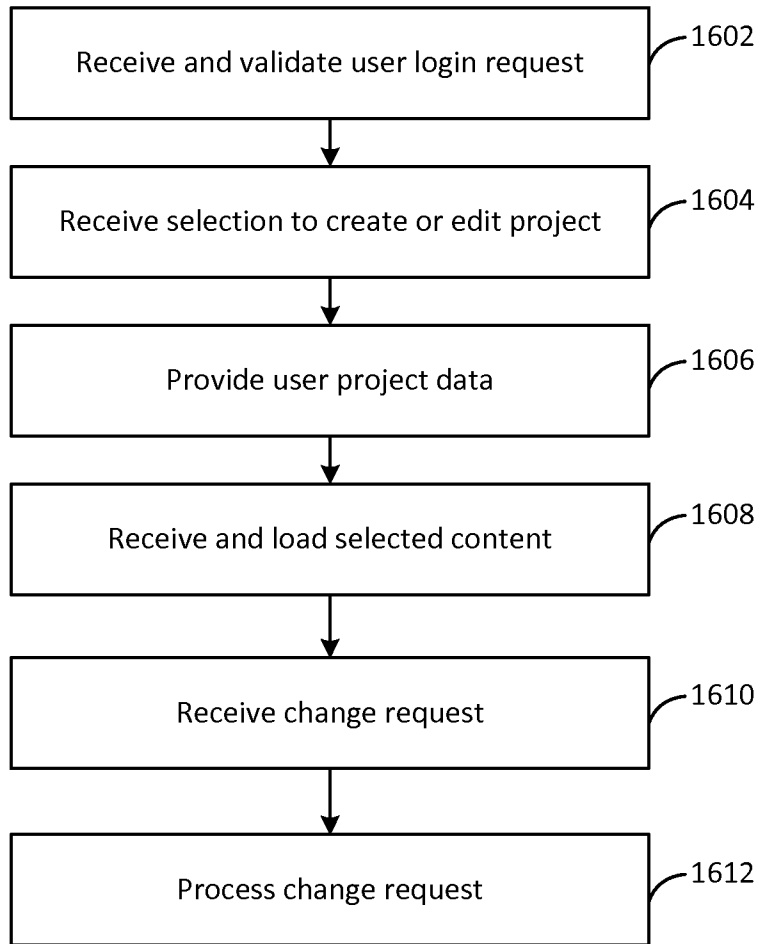


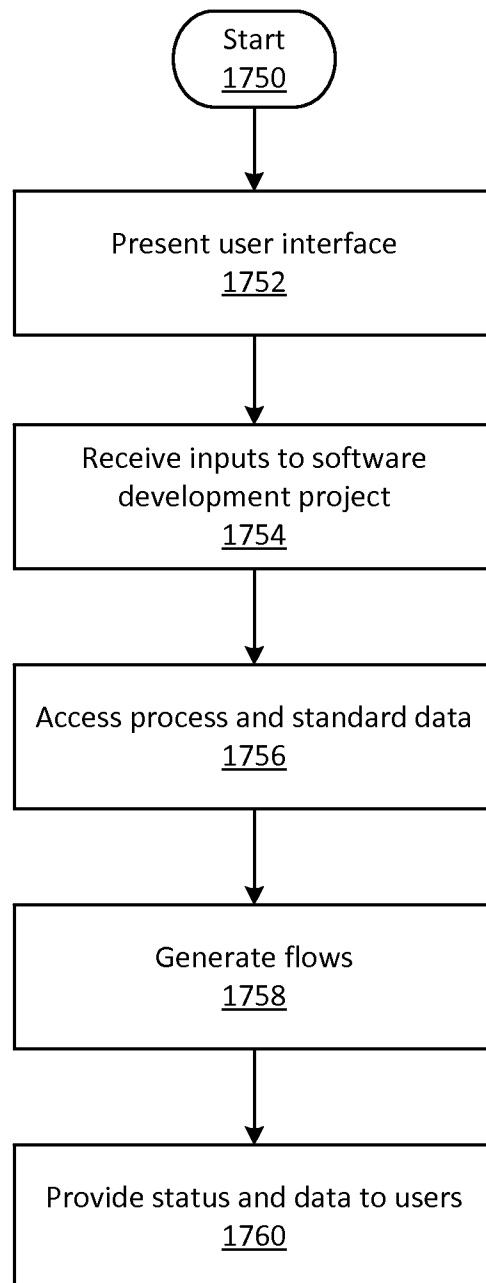
FIG. 15

16/18



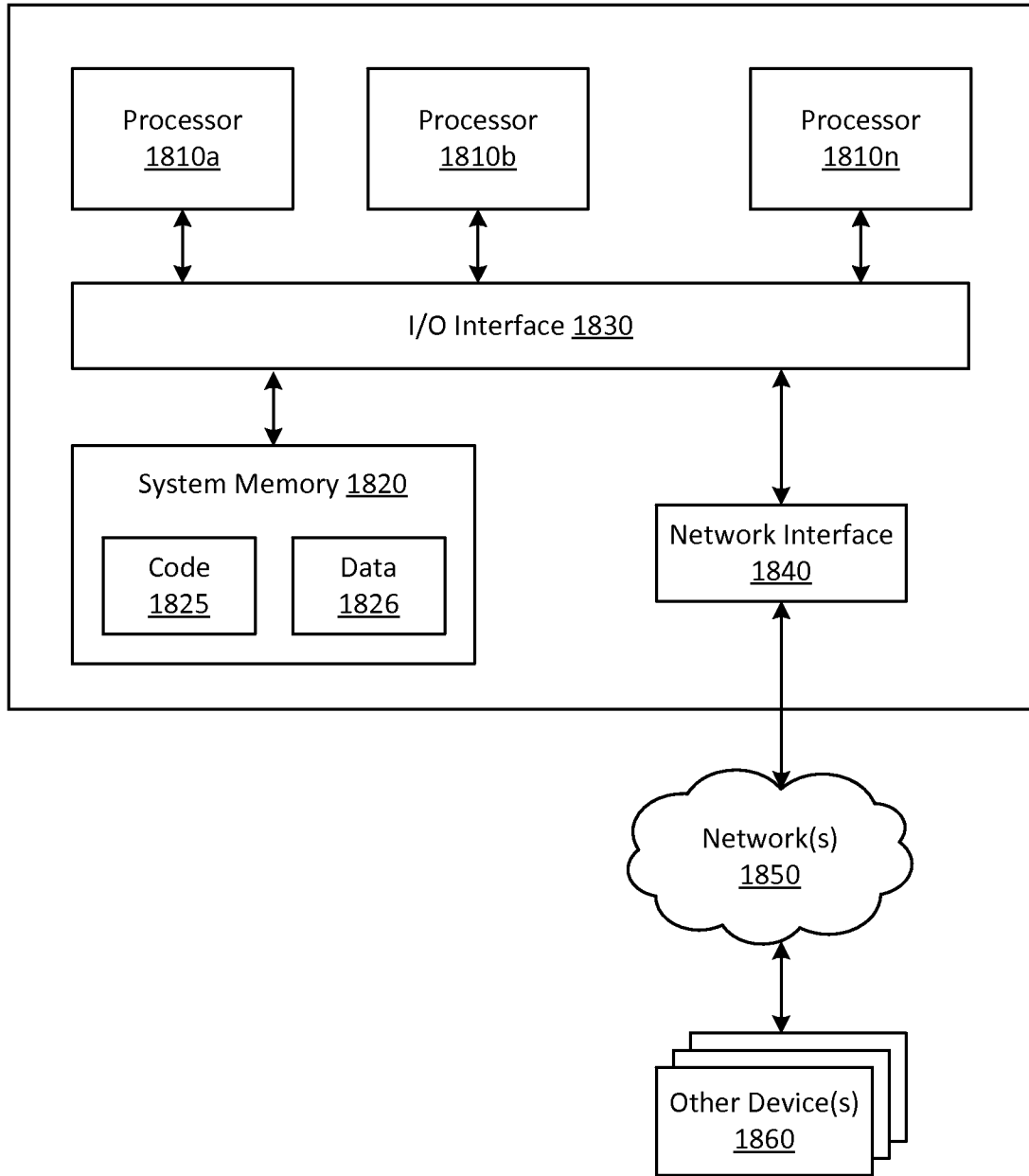
**FIG. 16**

17/18

**FIG. 17**

18/18

1800



**FIG. 18**

**INTERNATIONAL SEARCH REPORT**

International application No.  
**PCT/US2015/032065**

**A. CLASSIFICATION OF SUBJECT MATTER**

**G06F 9/44(2006.01)i, G06F 15/16(2006.01)i**

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

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)  
G06F 9/44; G06F 15/16; G06F 3/048; G06F 17/00; G06F 3/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched  
Korean utility models and applications for utility models  
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)  
eKOMPASS(KIPO internal) & Keywords: software, development, project, industry, standard

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2003-0055659 A1 (ERIC R. ALLING) 20 March 2003 See abstract; paragraphs [0014]-[0022]; and figures 1-2.	1-9
A	WO 2011-031328 A2 (LDRA TECHNOLOGY, INC. et al.) 17 March 2011 See abstract; paragraphs [0010]-[0055], [0185]; and figures 1A-2.	1-9
A	US 05671415 A (HOSSAIN; K. OMAR) 23 September 1997 See abstract; column 5, line 42 - column 9, line 57; and figures 1-2.	1-9
A	US 2008-0263505 A1 (STCLAIR WILLIAM GRVFFYTH et al.) 23 October 2008 See abstract; paragraphs [0022]-[0069]; and figures 1A-2.	1-9

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  
20 July 2015 (20.07.2015)

Date of mailing of the international search report  
**21 July 2015 (21.07.2015)**

Name and mailing address of the ISA/KR  
International Application Division  
Korean Intellectual Property Office  
189 Cheongsu-ro, Seo-gu, Daejeon Metropolitan City, 302-701,  
Republic of Korea  
Facsimile No. +82-42-472-7140

Authorized officer  
CHOI, Jeong Kwon  
Telephone No. +82-42-481-8507



**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International application No.

**PCT/US2015/032065**

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2003-0055659 A1	20/03/2003	None	
WO 2011-031328 A2	17/03/2011	WO 2011-031328 A3	07/07/2011
US 05671415 A	23/09/1997	US 05671415 A	23/09/1997
US 2008-0263505 A1	23/10/2008	CN 101689111 A	31/03/2010
		EP 2145252 A1	20/01/2010
		EP 2145252 A4	21/12/2011
		IL 201368 D0	31/05/2010
		US 2015-095890 A1	02/04/2015
		US 8949770 B2	03/02/2015
		WO 2008-124038 A1	16/10/2008
		WO 2008-124038 A8	31/12/2008