

SYSTEM, METHOD AND COMPUTER READABLE MEDIUM FOR CONFIGURING LIGHTWEIGHT WORKFLOW BASED ON MICRO-SERVICES

TECHNICAL FIELD

[0001] The present disclosure relates generally to a system, method and computer readable medium for configuring workflow based on micro-services. Embodiments of the disclosure are related to a configurable lightweight workflow engine under the micro-services architecture, comprising various individually and dynamically controllable workflow processes.

BACKGROUND

[0002] A workflow engine is a software application that manages business processes. It is a key component in workflow technology and typically makes use of a database server. Adoption of a workflow engine enables all the stakeholders including management, employees (and sometimes customers as well) to view all activities performed in a workflow simultaneously. Workflow engines also help managers take better control over process execution. For instance, the managers can quickly check the status of any document and see whether or not a document has been signed. Managers can do it at any time and at any stage of the process. Given that, the integration of a workflow engine contributes to increased transparency for all the stakeholders.

[0003] The value of workflow engines has been acknowledged in various fields and industries to enable effective coordination and structured work, for example, banking and financial services, automobile, manufacturing industry, oil and gas, real estate, logistics, information technology, healthcare, government and non-profit organizations, insurance and so on.

[0004] Micro-services are a variant of the Service-Oriented Architecture (SOA) structural style that arranges an application as a collection of loosely coupled services. In a micro-services architecture, services are fine-grained and the protocols are lightweight. The micro-services style of architecture develops

complex application software from small, individual applications that communicate with each other using language-independent interfaces (APIs).

[0005] In the existing workflow engine solutions, developers often need to reconfigure the entire process manually in Extensible Markup Language (XML) file, in order to cater to different business requirements. The reason behind this lengthy configuration process is the tight coupling development mode where business logic is written directly as a part of the code. Accordingly, if any addition or alteration is to be made to the workflow engine, the existing solutions are inevitably highly time-consuming and require repetitive development and testing work load, especially in case of frequent business logic change.

[0006] In addition, another problem persists in the existing workflow engine solutions. If the permission(s) and role(s) of the responsible person are changed, the existing solutions are unable to recover the task in time during the execution of the process, thereby resulting in potential information leakage.

[0007] The existing workflow engines also pose an issue in terms of inefficient execution of the process. In the existing art, when a task execution is blocked or enters a manual processing step, the entire workflow has to wait for the current task to complete before the next task can be processed, thereby resulting in lags and hindrances.

[0008] Further, in the existing workflow engine solutions, addition of more types of nodes requires adjusting existing execution logic in back-end code, rendering the process inconvenient to the user, time consuming and ineffective.

[0009] Moreover, in case of occurrence of an error during workflow execution, the existing workflow engine solutions require the user to restart the workflow from the beginning which often is accompanied by large amounts of user input, thus causing waste of time to unnecessarily re-process the already executed steps.

[0010] Additionally, existing workflow engines are very heavy to be adopted to manage business processes. Said engines inevitably lead to a highly time-consuming and repetitive development and testing workload when facing change of requirement or business change of mode.

SUMMARY

[0011] The specification describes a system, method and computer readable medium for configuring workflow based on micro-services. Embodiments of the disclosure are related to a configurable lightweight workflow engine under a micro-services architecture, comprising various individually and dynamically controllable workflow processes.

[0012] The methods, systems and computer readable media disclosed in the present application are advantageous in several ways, including but not limited to, providing a workflow which supports execution independent of the fixed configuration, automatic recycling or notification to responsible person(s) to reclaim the task, enabling information security, dynamic visualization of the workflow configuration mode thus making it convenient for business users to configure a desired workflow independently and quickly, easy and simple development of new node type without re-coding overall execution logic, prompt and accurate reprocessing of a workflow via graphic interface and smooth execution of workflow tasks without any delay attributed to blocked tasks.

[0013] It is, therefore, one aim of the disclosed embodiments to provide for system, method and computer readable medium for configuring and executing workflow process. A Graphical User Interface (GUI) displays one or more nodes related to at least one business processing requirement. Business users can drag and drop required nodes and can link nodes to configure at least one workflow process. The storage medium stores the workflow process along with a unique workflow identity. The execution unit retrieves the configuration node details based on the unique workflow identity and processes the workflow to

produce as result. The results are stored in the storage medium. Based on the business requirements, nodes are well defined and stored in the storage medium, for example a database. Tasks of all on-going work processes are put to the specified workflow engine queue, for example, RabbitMQ and is waited in the queue until be picked by a listener to perform as workflow engine queue.

[0014] It is, therefore, one aim of the disclosed embodiments to provide a configurable lightweight workflow engine under the micro-services architecture, comprising various individually and dynamically controllable workflow processes.

[0015] It is, therefore, one aim of the disclosed embodiments to provide a system, method and computer readable medium for configuring and executing workflow process that address the inflexibility of using Extensible Markup Language (XML) for process configuration, the commonality of services are extracted according to their characteristics and a process node model is constructed. With construction of different nodes according to their servicing purpose, a various yet extendable collection of nodes such as data service, review, approval, and supplement can be dynamically configured to a desired process according to business requirements. This allows the workflow to be executed independently of the fixed configuration.

[0016] It is, therefore, one aim of the disclosed embodiments to provide a system, method and computer readable medium for configuring and executing workflow process that prevents information leakage. The system uses configurable settings to trigger direct auto-recycling of distributed tasks and notifies users when their roles and permissions change. In the meanwhile, the tasks being recycled are redistributed in a timely manner. The system will automatically re-distribute the tasks in the event of a new setting of roles and permission being published.

[0017] It is, therefore, one aim of the disclosed embodiments to provide a system, method and computer readable medium for configuring and executing

workflow process in which, when the business requirement changes rapidly in current market, development of new node type is simple by just adding new node executor without re-coding overall execution logic.

[0018] It is, therefore, one aim of the disclosed embodiments to provide a system, method and computer readable medium for configuring and executing workflow process which supports re-processing of a workflow via a graphic interface. Users with the returning to node permission can return the workflow to any executed node. This way of going back to the executed nodes allows the process to be adjusted promptly and accurately in response to market changes, updating the system with the latest data.

[0019] It is, therefore, one aim of the disclosed embodiments to provide a system, method and computer readable medium for configuring and executing a workflow process in which during the process management, an asynchronous message mechanism is adopted, for example, based on RabbitMQ. Under this mechanism, if the processing of the process node changes, the node executor is notified asynchronously to obtain the latest process information. Based on the latest processing status, the specific execution logic of the process is determined, whether to continue to wait or return to the message queue and wait for the next processing. The asynchronous message processing mechanism makes processing of the process more intelligent. When the user role or role permissions change, the task execution permissions can be recovered in time to ensure information security.

[0020] It is, therefore, one aim of the disclosed embodiments to provide a configurable lightweight workflow engine based on micro-services framework. Dynamic visualization of the workflow configuration mode which makes it easy for business people to configure a desired workflow independently and quickly. Each task execution weight is equal. Blocked task does not affect the other workflow tasks. The system can automatically recycle or notify the relevant responsible person to reclaim the task. User can directly return to the workflow node that has already been executed, and re-execute the subsequent nodes

without requiring the customer or business personnel to re-initiate the process. It is easy to add new type of nodes to fulfil rapidly changed business requirements.

[0021] In a first aspect of the present disclosure, a system for configuring and executing a workflow based on micro-services is disclosed. The system comprises at least one Graphical User Interface (GUI) for displaying one or more nodes, at least one storage medium for storing at least one workflow process configured using the GUI along with a unique workflow identity and at least one execution unit for retrieving the workflow process based on the unique workflow identity from storage medium and processing the workflow to produce a result, wherein, the GUI is configured to allow a user to drag and drop the nodes and link the nodes to configure the workflow process.

[0022] According to an embodiment in conjunction to the first aspect of the present disclosure, the nodes are connected in a desired relationship to form the workflow process.

[0023] According to an embodiment in conjunction to the first aspect of the present disclosure, the relationship between two nodes is stored as a from-to pair.

[0024] According to an embodiment in conjunction to the first aspect of the present disclosure, the workflow process comprises a first node as a start node and a last node as an end node.

[0025] According to an embodiment in conjunction to the first aspect of the present disclosure, each node comprises a unique process node identity (ID).

[0026] According to an embodiment in conjunction to the first aspect of the present disclosure, the unique process node identity consists of a basic node code and a current timestamp.

- [0027]** According to an embodiment in conjunction to the first aspect of the present disclosure, each node is designed to have a different status change and a decision change.
- [0028]** According to an embodiment in conjunction to the first aspect of the present disclosure, each workflow process has at least one workflow ID.
- [0029]** According to an embodiment in conjunction to the first aspect of the present disclosure, each node in the workflow process has one workflow node ID.
- [0030]** According to an embodiment in conjunction to the first aspect of the present disclosure, each node configured in the workflow process has a corresponding node executor.
- [0031]** According to an embodiment in conjunction to the first aspect of the present disclosure, an execution weight of each node in the execution engine is equal.
- [0032]** According to an embodiment in conjunction to the first aspect of the present disclosure, each node in the workflow process waits in a queue for execution in the execution engine.
- [0033]** According to an embodiment in conjunction to the first aspect of the present disclosure, each node is queued up again after completion of execution.
- [0034]** According to an embodiment in conjunction to the first aspect of the present disclosure, the execution unit applies a validation to the workflow process based on a pre-defined validation rule.
- [0035]** According to an embodiment in conjunction to the first aspect of the present disclosure, the queue uses an asynchronous message mechanism based on RabbitMQ.

[0036] According to an embodiment in conjunction to the first aspect of the present disclosure, the execution unit receives an assignment, and processes the assignment to produce a result.

[0037] In a second aspect of the present disclosure, a method of configuring and executing a workflow based on micro-services is disclosed. The method comprises displaying, on at least one Graphical User Interface (GUI), one or more nodes and allowing a user to drag and drop the nodes and link the nodes to configure at least one workflow process; storing, in at least one storage medium, the workflow process with a unique workflow identity; retrieving, by at least one execution unit, the workflow process based on the unique workflow identity from the storage medium; and processing, by the execution unit, the workflow process to produce a result.

[0038] According to an embodiment in conjunction to the second aspect of the present disclosure, the method may further comprise receiving, by the execution unit, assignment and processing, by the execution unit, the assignment to produce a result.

[0039] According to an embodiment in conjunction to the second aspect of the present disclosure, the method may further comprise allowing the user to return to any executed returnable node.

[0040] According to an embodiment in conjunction to the second aspect of the present disclosure, the method may further comprise checking, by the execution engine, for a return-able node by finding, previous executed nodes of a current node using node relations in process nodes details and executed node details; sorting, previous executed nodes based on execution time and finding the latest executed node based on a pre-defined validation rule; applying, validation for the latest executed node; and marking, the latest executed node as return-able, if it passes the validation.

[0041] According to an embodiment in conjunction to the second aspect of the present disclosure, the method may further comprise sorting, the previous

executed nodes based on execution time and finding a latest executed node based on the pre-defined validation rule; applying, validation for the latest executed node; and marking, the latest executed node as return-able, if it passes the validation.

[0042] According to an embodiment in conjunction to the second aspect of the present disclosure, each node configured in the workflow process has a corresponding node executor.

[0043] According to an embodiment in conjunction to the second aspect of the present disclosure, the node executor is developed by defining, an abstract class implementing an execute method; defining, the node executor for each type of node by extending the abstract class; and defining, an execute factory with a register to put all the node code and executor pairs in a map.

[0044] According to an embodiment in conjunction to the second aspect of the present disclosure, the execute method of the abstract class comprises checking a status and initializing work, executing work, and changing the status and handling an error.

[0045] According to an embodiment in conjunction to the second aspect of the present disclosure, each node is designed to have a different status change and a decision change during execution of the workflow.

[0046] According to an embodiment in conjunction to the second aspect of the present disclosure, the method further comprises assigning an equal weight to each node and execution of each node in the order of sequence entered in an execution queue.

[0047] According to an embodiment in conjunction to the second aspect of the present disclosure, the method further comprises queuing up the nodes again after completing execution of the node.

[0048] According to an embodiment in conjunction to the second aspect of the present disclosure, the method further comprises removing automatically, by the execution unit, the node from the execution queue; and moving the node to a task pool, when the execution of the corresponding node is blocked.

[0049] According to an embodiment in conjunction to the second aspect of the present disclosure, the method further comprises removing automatically, by the execution unit, the node from the execution queue; and moving the node to a task pool, when permission of corresponding node is revoked.

[0050] In a third aspect of the present disclosure, a computer readable medium storing a program comprising instructions that, when executed by a system, cause the system to perform the method for configuring and executing a workflow based on micro-services is disclosed. The instructions, when executed by the system comprise, displaying on at least one Graphical User Interface (GUI), one or more nodes and allowing a user to drag and drop the nodes and link the nodes to configure at least one workflow process; storing, in at least one storage medium, the workflow process with a unique workflow identity; retrieving, by at least one execution unit, the workflow process based on the unique workflow identity from the storage medium; and processing, by the execution unit, the workflow process to produce a result.

[0051] According to an embodiment in conjunction to the third aspect of the present disclosure, the instructions stored on the computer storable medium comprise receiving, by the execution unit, assignment and processing, by the execution unit, the assignment to produce a result.

[0052] According to an embodiment in conjunction to the third aspect of the present disclosure, the instructions stored on the computer storable medium further comprise allowing the user to return to any executed returnable node.

[0053] According to an embodiment in conjunction to the third aspect of the present disclosure, the instructions when executed by the system further comprise, checking by the execution engine for a return-able node by: finding,

previous executed nodes of a current node using node relations in process nodes details and executed node details; sorting previous executed nodes based on execution time and finding the latest executed node; applying validation for the latest executed node based on a pre-defined validation rule; and marking the latest executed node as return-able, if it passes the validation.

[0054] According to an embodiment in conjunction to the third aspect of the present disclosure, the instructions when executed by the system further comprise, sorting the previous executed nodes based on execution time and finding a latest executed node based on the pre-defined validation rule; applying validation for the latest executed node; and marking the latest executed node as return-able, if it passes the validation.

[0055] According to an embodiment in conjunction to the third aspect of the present disclosure, each node configured in the workflow process has a corresponding node executor.

[0056] According to an embodiment in conjunction to the third aspect of the present disclosure, the node executor is developed by: defining an abstract class implementing an execute method; defining the node executor for each type of node by extending the abstract class; and defining an execute factory with a register to put all the node code and executor pairs in a map.

[0057] According to an embodiment in conjunction to the third aspect of the present disclosure, the execute method of the abstract class comprises a status checking and an initialization work part, an actual execution work part, and a status change and an error handling part.

[0058] According to an embodiment in conjunction to the third aspect of the present disclosure, each node is designed to have a different status change and a decision change during execution of the workflow.

[0059] According to an embodiment in conjunction to the third aspect of the present disclosure, the instructions when executed by the system further

comprise, assigning an equal weight to each node and execution of each node in the order of sequence entered in an execution queue.

[0060] According to an embodiment in conjunction to the third aspect of the present disclosure, the instructions when executed by the system further comprise, queuing up the nodes again after completing execution of the node.

[0061] According to an embodiment in conjunction to the third aspect of the present disclosure, the instructions when executed by the system further comprise, automatically removing, by the execution unit, the node from the execution queue and moving the node to a task pool, when the execution of the corresponding node is blocked.

[0062] According to an embodiment in conjunction to the third aspect of the present disclosure, the instructions when executed by the system further comprise, automatically removing, by the execution unit, the node from the execution queue and moving the node to a task pool, when permission of the corresponding node is revoked.

[0063] The following disclosure is provided to facilitate an understanding of some of the innovative features unique to the disclosed embodiment and is not intended to be a full description. A full appreciation of the various aspects of the embodiments disclosed herein can be gained by taking into consideration the entire specification, claims, drawings, and abstract as a whole.

BRIEF DESCRIPTION OF THE DRAWINGS

[0064] **FIG. 1** is a block diagram depicting an example system **10** for configuring and executing a workflow process, in accordance with the disclosed embodiments;

[0065] FIG. 2 is a diagrammatic representation of an example list **100** of basic nodes with different node types for business processing requirement, in accordance with the disclosed embodiments;

[0066] FIG. 3 is a diagrammatic representation of an example of configure details **200** of workflow nodes, in accordance with the disclosed embodiments;

[0067] FIG. 4 is a flow diagram that illustrates an example application **250** of a logic of checking return-able nodes, to each node, in accordance with the disclosed embodiments;

[0068] FIG. 5 is a diagrammatic representation of an example “return to node” modal **300** used in system **10**, in accordance with the disclosed embodiments;

[0069] FIG. 6 is a flow diagram that illustrates an example process **350** of developing a node executor, in accordance with the disclosed embodiments;

[0070] FIG. 7 is a schematic diagram that illustrates an exemplary workflow configuration **500**, in accordance with the disclosed embodiments;

[0071] FIG. 8 is a flow diagram that illustrates an example method **550** of configuring and executing a workflow engine based on a micro-services architecture using system **10** of FIG. 1, in accordance with the disclosed embodiments; and

[0072] FIG. 9 is a schematic diagram that illustrates various functions **600** performed by an example execution engine, in accordance with the disclosed embodiments.

[0073] The summary above, as well as the following detailed description of illustrative embodiments, are better understood when read in conjunction with the appended figures. For the purpose of illustrating the present disclosure, exemplary constructions of the disclosure are shown in the figures. However, the disclosure is not limited to specific methods and instrumentalities disclosed

herein. Moreover, those in the art will understand that the drawings are not to scale. Wherever possible, like elements have been indicated by identical numbers.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0074] The particular configurations discussed in the following description are non-limiting examples that can be varied and are cited merely to illustrate at least one embodiment and are not intended to limit the scope thereof.

[0075] Workflow engines are used to manage processes, commonly in businesses and are a key component in the workflow technology. Workflow engines typically makes use of a database server.

[0076] The present invention generally relates to a configurable lightweight workflow engine based on micro-services frameworks, including frontend configuration and back-end workflow engine.

[0077] **FIG. 1** is a block diagram depicting an example system **10** for configuring and executing a workflow process, in accordance with the disclosed embodiments. The system **10** comprises at least one Graphical User Interface (GUI) of a computing device **12**, at least one storage medium **14** and at least one execution unit or workflow engine **16**. The GUI **12** visually represents or displays the components of the workflow process. The GUI **12** allows the user to perform actions such as configure a process, return to any executed nodes for re-processing and so on. In the present specification, process refers to a drag-able combination of nodes and relations. The GUI **12** displays one or more nodes related to at least one business processing requirement. A business user can drag and drop required nodes and can link nodes to configure at least one workflow process **13**. jsplumb is an open source JS library used for building connectivity applications and it is imported in front-end coding to build relationships between different nodes. After the business user drags and drops each node into the configure panel of the GUI, a unique process node Identity (processNodeId) is generated. The processNodeId

consists of basic node code and current timestamp. For example, when a contract node is placed into the panel, an Identity (ID) for example, contract_1581385010000 is assigned as processNodeId. And when the business user links each node with lines, the relationship between two nodes are stored as a from-to pair.

[0078] After a process is created by the business user, the business user can use such a process to create a workflow so that detailed execution of each node can be configured. When a workflow is initially created, nodes and node relations of the process used by the workflow are saved to the storage medium **14** with another unique workflow node ID associated with workflow ID. The storage medium **14**, may be a database and store, for example, nodes, nodeRelations of the process used by the workflow, current node details, workflow process, execution logic, workflow node ID or decision node ID, workflow ID. Clicking on each node will open a configure modal to set up details. Each node type will have a different configuration modal which the developers can define. Once the configuration of a node is saved, the configuration associated with that decision node ID is immediately saved. When the back-end engine executes the workflow, it will fetch corresponding configuration details based on the decision node ID or the workflow ID.

[0079] The execution unit or the workflow engine or the back-end engine **16** retrieves the configuration node details **17** based on the unique workflow identity from storage medium **14** and processes the workflow to produce a result **15**. The results **15** are stored in the storage medium **14**. Based on the business requirements, nodes are well defined and stored in the storage medium **14**. Tasks of all on-going work processes are put to the specified workflow engine queue **11**, for example, RabbitMQ, and said tasks wait in the queue until picked by a listener to perform as a workflow engine queue. The storage medium **14** can be a database or database server. The execution unit or the workflow engine or the back-end engine **16** can also receive an assignment **19** (work received from outside the system) and process the same to produce the result **15**.

[0080] A data structure is formed and sent to the back-end when a process is created jsplumb is an open source JS library used for building connectivity applications and it is imported in front-end coding to build relationships between different nodes. The data structure discloses various nodes and relationships between each node. Each node is designed to have different status change and decision change. After the business user drags and drops each node into the configure panel, a unique process node Identity (processNodeId) is generated. The processNodeId consists of basic node code and current timestamp. For example, when a contract node is placed into the panel, an Identity (ID) for example, contract_1581385010000 is assigned as processNodeId. And when the business user links each node with lines, the relationship between two nodes are stored as a from-to pair. Before the user saves the process, in some implementations of the present invention, a validation may be applied to the overall process based on predefined validation rules, which can be defined by the developer. For example, one predefined validation rule may be that a process must contain one and only one end node.

[0081] **FIG. 2** is a diagrammatic representation of an example list **100** of basic nodes with different node types for business processing requirement, in accordance with the disclosed embodiments. The example list **100** is a non-limiting list disclosing different nodes that can be created depending on the business processing requirements, including but not limited to assist nodes, manual nodes and auto nodes.

[0082] The assist nodes, for example, as represented in **FIG. 2** are the “start node” **102**, “end node” **134**, “decide node” **106**. The start node **102** and the end node **134** are used to indicate the start and end point of a process respectively, while the decide node **106** is used to handle branching situations.

[0083] The manual nodes refer to nodes that involve manual intervention. The manual nodes, for example, as represented in **FIG. 2** are the “approval (user)” **104**, “review (role)” **108**, “disbursement” **110**, “collateral” **112**, “supplement (role)” **118**, “account” **120**, “interview” **122**, “in-principal” **124**, “review (user)” **126**, “approve (role)” **128**, “supplement (user)” **130** and “contract” **132**. The

approval nodes **104** are the nodes used by the business users to approve the application. Multiple users or a group of users with selective permission can be defined to process or approve applications. For example, in the banking or financial services, an operator may be permitted to also change the final approved amount or interest rate on this node. The review nodes **126** are the nodes used by business users to review the application. In some implementations, a voting mechanism may be configured to review the application. There may be two modes during a review, for example, single review and multi review, which can support voting mechanisms for the application review. For example, if three different reviewers are required to review the application, it may be required that at least two people give a pass vote for the application to process further. The supplement nodes **118** are the nodes used by business users for requesting or providing additional or supplement information for the application. For example, bank officers may need to fill in some credit information or upload specific documents before the application can be approved. The interview node **122** is used by business users to perform online interviews with applicants. The contract node **132** is used by business users to define and generate contracts for applications. To provide in-principle approval for the application, the node **124** in-principle approval may be used. The collateral node **112** is used to check the collateral information used in the application. For example, checking the invoice validity, checking the vehicle or house information and so on. The account node **120** is used to open a customer bank account. This normally happens before the loan disbursement.

[0084] Auto nodes, for example, as represented in **FIG. 2** are “risk rule” **114** and “data service” **116** and may be used to evaluate rules or fetch third party data for rule evaluation. The “risk rule” node **114** is used to define anti-fraud or risk rules to evaluate the application. An example of the rules can be, “applicants must be twenty years old or above”. The “data service” node **116** is used to connect to third party data sources to retrieve relevant information, for example, credit bureau or bank credit history information.

[0085] It should be noted that all those basic nodes in the list **100** are stored in storage medium **14**, for example, a database. Developers can define additional basic nodes as per the business requirements of business users.

[0086] After a process is created, a business user can create a workflow so that details of the execution of each node can be configured. When a workflow is initially created, nodes and node relations of the process used by the workflow are fetched from Database (DB) and saved to workflow DB with another unique workflow node ID associated with workflow ID. Clicking on each node will open a configure modal to set up details. Each node type will have a different configuration modal which the developers can define. Once the configuration of a node is saved, the configuration associated with that decision node ID is immediately saved. When the back-end engine executes the workflow, it will fetch corresponding configuration detail based on decision node ID. It should be noted that the terms decision node ID and workflow node ID are interchangeably used in this disclosure.

[0087] **FIG. 3** is a diagrammatic representation of an example of configure details **200** of workflow nodes, in accordance with the disclosed embodiments. In the diagrammatic representation in **FIG. 3**, the “start” node **202** is connected to the “data service” node **204** and then to the “risk rule” node **206** and the “decide” node **208**. Based on the decision of “decide” node **208**, the process either proceeds directly to “approval” node **212** or first to “review” **210** and then to “approval” **212**. Then, the process proceeds towards the “contract” node **214**, the “data service” node **216** and eventually to the “risk rule” node **218**. The configure details **200** of workflow nodes, as shown in **FIG. 3** is an example based on the business requirement of the user or business. The user can perform manual intervention when the workflow is currently at manual nodes like review, approve, and supplement. Besides normal intervention, users can choose to return to the previous executed node when necessary. In some implementations, the “return to node” modal may be implemented wherein the process of the flow is fetched and displayed. The back-end will return executed nodes details, all process nodes details, and current node details at the same

time. When rendering the nodes of the process, a logic of checking return-able is applied to each node, as shown in **FIG. 4**.

[0088] **FIG. 4** is a flow diagram that illustrates an example application of a logic of checking return-able nodes, to each node, in accordance with the disclosed embodiments. As disclosed in **FIG. 4**, the logic can be applied, for example, at step **252**, by finding the previously executed nodes of a current node using node relations in process nodes details, and executed nodes details. Then, at step **254** the previously executed nodes may be sorted based on execution time and the latest executed node is identified or found. Thereafter, as disclosed in **FIG. 4**, the logic can be applied at step **256**, by applying validation for the latest node identified or found in the last step and the validation check is performed. Developers can define validation rules based on business requirements. For example, a validation rule may be that the node should be active and the node type should be manual. If the node passes the validation, the latest executed node is marked as return-able at step **258**. Thereafter, the previously executed nodes identified in step **252** are sorted based on execution time and the now latest executed node is found or identified as at step **254**, validation is applied for said latest executed node and if the node passes the validation as at step **256**, said node is marked as return-able as at step **258**. As disclosed in **FIG. 4**, the logic can be applied, for example, by performing the above steps for all or as many nodes as required. In the example application as disclosed in **FIG. 4** for a logic of checking return-able nodes, after return to a specific node, for example A, the workflow will start from node A again and continue the process.

[0089] In a “return to node” modal, all return-able nodes are clickable. Business users can select the wanted node(s) to re-execute from in the “return to node” modal. Back-end workflow engine handles the flow of execution after the business user submits a return request.

[0090] **FIG. 5** is a diagrammatic representation of an example “return to node” modal **300** used in system **10**, in accordance with the disclosed embodiments. In the example “return to node” modal of **FIG. 5**, it is disclosed that each return-

able node is clickable. Business users can select the specific node to re-execute from. Back-end workflow engine handles the flow of execution after the business user submits a return request. In **FIG. 5**, for example, nodes such as “start” node **302**, “test” node **304**, “supplement” node **306**, “review” node **308** and “approve” node **310** are shown. As shown in the example modal, a “test” node **304** is return-able and therefore, clickable. If the current node in the workflow process is the “supplement” node **306**, the business user can click on the returnable node that is, the “test” node **304** to re-execute it.

[0091] **FIG. 6** is a flow diagram that illustrates an example process **350** of developing a node executor, in accordance with the disclosed embodiments. Each type of node configured in a workflow process, except the start and the end nodes, has a corresponding node executor. Node executor is the basic unit of workflow execution. As disclosed in **FIG. 6**, a node executor may be developed **350**, for example, by defining an abstract class “AbstractNodeExecutor” implementing “execute” method of interface “Executable” at step **352**. The “execute” method of the abstract node executor contains three parts, that is, the initial part which includes status checking and initialization work, the second part which does the actual execution work, and the last part which performs status change and error handling. Then, the node executor for each type of node is defined by extending the abstract node executor at step **354**. The node executor implements all the abstract methods and has its own implementations. Thereafter, as disclosed in **FIG. 6**, the process **350** can be implemented, for example at step **356**, by defining execute factory with a static method “register” to put all the node code and executor pairs in a map, and then providing a method to get executor by node code.

[0092] Execution of each node can be referred to as a task, as done in the present specification. Tasks of all on-going approval processes are put to the specified workflow engine queue and said tasks wait in the queue until picked by a listener. In some implementations, Rabbit MQ may be used to perform as the workflow engine queue. Each task has an equal weight and is executed based on the rule of FIFO (First In, First Out) in the message queue. When the listener picks a task from the queue, based on the current node detail stored

in the database, the listener will execute the task using the respective node executor. Once this task is finished, the current node of the corresponding process is updated to the next node based on the configuration. A new task for next node execution is put to the end of the same queue. The above pick-and-execute steps will repeat until reaching the “End” node of the workflow.

[0093] Execution status is designed for node executors to mark the status of the node execution which is used to make sure that the task to be executed is represented by the correct status. Each task to be executed should have the correct execution status – “waiting”. Below Table 1 provides a detailed description of each execution status.

Table 1: Execution status

Execution Status	Description
waiting	When a task is put in the queue, the execution status is set as “waiting”. After this task is picked by the listener, the executor will check if the status is “waiting” at the initial verification stage, else error is thrown.
running	After the initial verification is passed, the execution status is changed to “running” which means the task is at running stage.
blocked	If a task is waiting for an external decision or depending on a response from another system, the execution status is updated to “blocked” until it gets a response. Once the response is returned, the task is put back to the queue with “waiting” status again.
end	Status of “end” is only set at the cases when error is encountered or the workflow reaches its “End” node.

[0094] Workflow status is designed to distinguish the entire execution status of workflow. The below Table 2 provides a detailed description of each workflow status:

Table 2: Workflow status

Workflow Status	Description
start	When a workflow is about to start execution, the workflow is initiated with “start” status.
inProgress	The workflow status is changed from “start” to “inProgress” until the entire workflow is finished unless the workflow is blocked or has an error.
blocked	When the current task of workflow is at “blocked” execution status, the workflow status is “blocked” as well.
error	If there is an error encountered during any task of this workflow, the workflow status becomes “error”.
retry	If an error is encountered, a retry function is provided. Once the retry is triggered, the status is converted from “error” to “retry”.
end	When the workflow reaches its “End” node, the workflow status becomes “end”.

[0095] The workflow engine solution disclosed in the present invention can be applied in various fields and industries to enable effective coordination and structured work, for example, banking and financial services, automobiles, manufacturing industry, oil and gas, real estate, logistics, IT, healthcare, government and non-profit organizations, insurance and so on.

[0096] **FIG. 7** is a schematic diagram that illustrates an exemplary workflow configuration **500**, in accordance with the disclosed embodiments. The workflow configuration **500** as shown in **FIG. 7**, is used for example for an order approval process, or a loan application approval process and comprises a “start” node **502**, a “decide” node **504**, a “data service” node **506**, a “risk rule” node **508**, an “interview” node **510**, an “approval” node **512**, a “contract” node **514** and an “end” node **516**.

[0097] As disclosed in **FIG. 7**, an exemplary workflow configuration comprises a “start” node **502** which indicates the starting point for the workflow execution. Further, a “decide” node **504** is disclosed. Each “decide” node **504** may have multiple downstream nodes. For example, in a loan application approval process, a “decide” node may be configured such that an application is sent to the “data service” node **506** if a customer works for a government institute, otherwise, the application goes to the “risk rule” node **508**. “Data service” nodes **506** usually call or refer to some external data providers to check for more information. For instance, in the case of the above example where a customer works for a government institute, bank officials may connect to some government provided API to check if the customer is eligible to apply for the loan. After the check, the application flows to “risk rule” node **508** as disclosed in **FIG. 7**.

[0098] The “risk rule” node **508** performs some checks based on rule configuration. For instance, in the case of the above example, a loan may only be provided to customers whose age is less than sixty five years. A rule can be configured at this node, and can automatically reject those applications applied by those customers aged more than sixty five. If the customer is less than sixty five years old, the flow goes to the next node, that is the “interview” node **510** in **FIG. 7**. In the “interview” node **510**, the respective bank officer can schedule an online interview with the applicant to conduct some questionnaire and clarify any doubts about the application. If the application is processed following the interview, the flow goes to the next node disclosed as the “approval” node **512** in **FIG. 7**. A bank officer or the approver can give the final

approval for the application in the “approval” node **512**. Once the approval is given, the flow goes to the next node, disclosed as the “contract” node **514**, to generate the contract document based on pre-configured templates. If the business user accepts the contract **514**, and the agreement is signed, the flow will move to the final node called the “end” node **516**, which marks the end of the workflow.

[0099] Table 3 shows the status change during the overall workflow execution in the condition that “data service” node **506** is determined to be executed after Decide node **504**, and assuming the approval process successfully reached End node **514**.

Table 3: Status change during an order approval process

Order Status	Workflow Status	Execution Status	Last Finished Node	Current Node	Next Node	Remarks
init	start	waiting	Start		Decide	During order submission, status will be initialized and next node will be set to be “Decide” node.
approvalInProgress	inProgress	running	Start	Decide		The executor is running.
						Assuming “Decide” node goes to “Data Service” branch. In fact, it will depends on the rules configured at “Decide” node to decide which branch to go.
approvalInProgress	inProgress	waiting	Decide		Data Service	
approvalInProgress	inProgress	running	Decide	Data Service		Data service node is running.
approvalInProgress	blocked	blocked	Decide	Data Service		Data service is waiting for result for data server call.
approvalInProgress	inProgress	waiting	Decide	Data Service		Result returned and is waiting for execution.
approvalInProgress	inProgress	running	Decide	Data Service		Data service node is running again.
approvalInProgress	inProgress	waiting	Data Service		Risk Rule	“Risk Rule” node is waiting in the queue to be executed.
approvalInProgress	inProgress	running	Data Service	Risk Rule		“Risk Rule” node is running.
approvalInProgress	inProgress	waiting	Risk Rule		Interview	“Interview” node is waiting to be executed.
approvalInProgress	inProgress	running	Risk Rule	Interview		“Interview” node is running.
approvalInProgress	blocked	blocked	Risk Rule	Interview		Waiting for interview result.
						Interview result returned. “Interview” node is waiting to be executed again.
approvalInProgress	inProgress	waiting	Risk Rule	Interview		
approvalInProgress	inProgress	running	Risk Rule	Interview		“Interview” node is running again.
approvalInProgress	inProgress	waiting	Interview		Approve	“Approve” node is waiting to be executed.
approvalInProgress	inProgress	running	Interview	Approve		“Approve” node is running.
approvalInProgress	blocked	blocked	Interview	Approve		“Approve” node is pending for approval by user.
						Approval is given by user. Now is waiting for to be executed again.
approvalInProgress	inProgress	waiting	Interview	Approve		
approvalInProgress	inProgress	running	Interview	Approve		“Approve” node is running again.
approved	inProgress	waiting	Approve		Contract	“Contract” node is waiting to be executed.
approved	inProgress	running	Approve	Contract		“Contract” node is running.
approved	blocked	blocked	Approve	Contract		Waiting for contract signing result.
approved	inProgress	waiting	Approve	Contract		Contract signing is finished and result returned.
approved	inProgress	running	Approve	Contract		“Contract” node is running again
approved	End	End	End			Workflow is executed successfully reaching “End” node.

[00100] FIG. 8 is a flow diagram that illustrates an example method **550** of configuring and executing a workflow engine based on a micro-services architecture using system **10** of FIG. 1, in accordance with the disclosed embodiments. As disclosed in FIG. 8, the process **550** can be implemented, for example, at step **552** by displaying one or more nodes related to at least one business processing requirement using at least one Graphical User

Interface (GUI) and then allowing a user to drag and drop the nodes and link nodes to configure at least one workflow process at step **554**. Then, as disclosed in **FIG. 8**, the method **550** can be implemented, for example, at step **556** by storing the workflow process configured using the GUI along with a unique workflow identity in at least one storage medium. Further, the process includes retrieving the workflow process based on the unique workflow identity from the storage medium at step **558** and then processing said workflow process using at least one execution unit to produce a result at step **560**. Thereafter, the example process disclosed in **FIG. 8** can be implemented by storing the result in the storage medium at step **562**. When a workflow is initially created, nodes and nodeRelations of the process used by the workflow are fetched or retrieved from the database and saved to the workflow database with another unique workflow node ID associated with workflow ID. Clicking on each node will open a configure modal to set up details. Each node type has a different configure modal which can be defined by the developers.

[00101] FIG. 9 is a schematic diagram that illustrates various functions **600** performed by an example execution engine, in accordance with the disclosed embodiments. When an error **608** is encountered during the workflow execution, the order status is updated to “error” and the entire workflow execution is stopped. To continue the execution, a “retrigger” function is provided to retrigger the execution from the stopping node. When a task is blocked due to waiting for external data service results or manual intervention, such as manual review or approval, the task is removed from the execution queue and does not affect the processing of the entire service and hence, improves processing efficiency significantly. Each task execution weight is equal, as at block **602**, and is executed in order of sequence entering the message queue. When a task execution is blocked, it does not affect the processing of the entire service and hence improves processing efficiency significantly. Every time the execution unit or workflow unit or engine **16** fetches a task waiting in queue, it executes the next node, as at block **604**. Upon finishing the execution of the node, the workflow engine re-queues the task to the end of the queue and processes the next task. This ensures every task is allocated an equal chance of execution. Meanwhile, if a task is blocked, the

task is removed, as at block **606** from the execution queue and the workflow engine moves it to the task pool for manual intervention. In the event the permission to a respective user or position to handle a task is revoked, the system can automatically put the task back to the pool so it can be assigned to someone else, as at block **610**. The respective users are notified about the permission change and information security is preserved. For example, if a user pulls a task pending for review to make an amendment and the user is deprived of the right to review the task before any action is taken, said user will not be able to view and take any action on the task. In the event of a change of business view and market data, the process node that has already been executed can be resumed, and the entire process can be re-executed without requiring the customer or business personnel to re-initiate the process, greatly enhancing the user experience, as at block **612**. A workflow consists of several independent steps or nodes and the result of a node execution depends on its context, which affects the subsequent flow. In an example where a workflow consists of seven independent nodes, if a task at node five needs to be re-evaluated from node five, the user can return the task to node five using a button and selecting the node. In the condition a “return to node” is performed by manual intervention, the current node is updated to the selected return node and the task is put in the workflow execution queue. All the subsequent nodes are then re-processed based on the new node execution result. Under the micro-services architecture, different services concentrate on different functionalities and won't affect the work of each other. As the present invention uses a message queue to dispatch tasks to be executed by different micro-services, developers may write a new node executor if a new node type is needed as per business requirement, thus increasing the flexibility and scalability of the flow.

[00102] The present invention makes it easy for business users to configure a desired workflow process independently and quickly because of the dynamic visualization of the process configuration mode. The workflow process can be put into production immediately without requiring any intervention by the developer and thus, the present invention saves a lot of time. Business users can also change the process in graphical interface by rearranging and

connecting the required nodes in desired order to form a new workflow. The new workflow takes immediate effect for any incoming tasks.

[00103] Thus, by way of the above disclosure, the present invention is able to provide a configurable lightweight workflow engine where business users can quickly configure a required business process with flexible and fast execution and can be applied in various fields and industries requiring, for example, banking and financial services, automobiles, manufacturing industry, oil and gas, real estate, logistics, IT, healthcare, government and non-profit organizations, insurance etc.

[00104] The present invention also overcomes the inflexibility of using XML for process configuration as the process node model of the present invention is constructed based on the commonality of services extracted according to their characteristics. Thus, the workflow attained with the solution of the present invention is configurable using a graphical user interface, thus making said workflow easy to use unlike the existing solutions wherein each workflow is required to be developed using XML. Further, in the present invention, developing a graphical interface is simple and straightforward.

[00105] The workflow engine of the present invention is extremely light-weight and is based on a micro-services architecture, including front-end configuration and back-end workflow engine.

[00106] The workflow engine of the present invention processes tasks in an absolutely fair and non-blocking manner, so every task gets an equal chance of execution. Tasks are returned to the pool in case the authorization of privileges of a specific user are revoked, thereby ensuring data privacy.

[00107] The workflow engine of the present invention also permits the returning of task execution to any previous node, so tasks can be re-executed given new context. Exceptions are handled in the workflow automatically without affecting other tasks should task execution fail, and the concerned task can be triggered at a later point of time thereby avoiding lags.

[00108] It will be appreciated that variations of the above disclosed and other features and functions, or alternatives thereof, may be desirably combined into many other different systems or applications. Also, various presently unforeseen or unanticipated alternatives, modifications, variations or improvements therein may be subsequently made by those skilled in the art which are also intended to be encompassed by the following claims.

[00109] Although embodiments of the current disclosure have been described comprehensively in considerable detail to cover the possible aspects, those skilled in the art would recognize that other versions of the disclosure are also possible.

CLAIMS

What is claimed is:

1. A system for configuring and executing a workflow based on micro-services, comprising:
 - at least one Graphical User Interface (GUI) for displaying one or more nodes;
 - at least one storage medium for storing at least one workflow process configured using the GUI along with a unique workflow identity; and
 - at least one execution unit for retrieving the workflow process based on the unique workflow identity from storage medium and processing the workflow to produce a result;wherein, the GUI is configured to allow a user to drag and drop the nodes and link the nodes to configure the workflow process.
2. The system of claim 1, wherein the nodes are connected in a desired relationship to form the workflow process.
3. The system of claim 2, wherein the relationship between two nodes is stored as a from-to pair.
4. The system of claim 1, wherein each node comprises a unique process node identity (ID).
5. The system of claim 4, comprising one or more of:
 - a basic node code and a current timestamp; and
 - a workflow ID and a workflow node ID.
6. The system of claim 1, wherein each node is designed to have a different status change and a decision change.

7. The system of claim 1, wherein each node configured in the workflow process has a corresponding node executor.
8. The system of claim 1, wherein an execution weight of each node in the execution engine is equal.
9. The system of claim 1, wherein each node in the workflow process waits in a queue for execution in the execution engine and uses an asynchronous message mechanism.
10. The system of claim 9, wherein each node is queued up again after completion of execution.
11. The system of claim 1, wherein the execution unit applies a validation to the workflow process based on a pre-defined validation rule.
12. The system of claim 1, wherein the execution unit receives an assignment, and processes the assignment to produce a result.
13. A method of configuring and executing a workflow based on micro-services, comprising:
 - displaying, on at least one Graphical User Interface (GUI), one or more nodes and allowing a user to drag and drop the nodes and link the nodes to configure at least one workflow process;
 - storing, in at least one storage medium, the workflow process with a unique workflow identity;
 - retrieving, by at least one execution unit, the workflow process based on the unique workflow identity from the storage medium; and
 - processing, by the execution unit, the workflow process to produce a result.
14. The method of claim 13, comprises:
 - receiving, by the execution unit, an assignment; and
 - processing, by the execution unit, the assignment to produce a result.

15. The method of claim 13, further comprising:
 - allowing the user to return to any executed returnable node;
 - checking, by the execution engine, for a return-able node by:
 - finding, previous executed nodes of a current node using node relations in process nodes details and executed node details;
 - sorting, previous executed nodes based on execution time and finding the latest executed node;
 - applying, validation for the latest executed node based on a pre-defined validation rule; and
 - marking, the latest executed node as return-able, if it passes the validation;
 - sorting, the previous executed nodes based on execution time and finding a latest executed node;
 - applying, validation for the latest executed node based on the pre-defined validation rule; and
 - marking, the latest executed node as return-able, if it passes the validation.
16. The method of claim 13, wherein each node configured in the workflow process has a corresponding node executor and the node executor is developed by:
 - defining, an abstract class implementing an execute method;
 - defining, the node executor for each type of node by extending the abstract class; and
 - defining, an execute factory with a register to put all the node code and executor pairs in a map,wherein the execute method of the abstract class comprises checking a status and initializing work, executing work, and changing the status and handling an error.
17. The method of claim 13, wherein each node is designed to have a different status change and a decision change during execution of the workflow.
18. The method of claim 13, further comprising:
 - assigning an equal weight to each node and execution of each node in the order of sequence entered in an execution queue.
19. The method of claim 13, further comprising:
 - queuing up the nodes again after completing execution of the node;

removing automatically, by the execution unit, the node from the execution queue;
and

moving the node to a task pool, when the execution of the corresponding node is blocked or when permission of corresponding node is revoked.

20. A computer readable medium storing a program comprising instructions that, when executed by a system, cause the system to perform the method for configuring and executing a workflow based on micro-services of any of claims 13 to 19.