

(54) **ARTIFICIAL INTELLIGENCE  
LAYER-BASED PROCESS EXTRACTION  
FOR ROBOTIC PROCESS AUTOMATION**

(60) Provisional application No. 62/915,351, filed on Oct. 15, 2019.

(71) Applicant: **UiPath, Inc.**, New York, NY (US)  
(72) Inventors: **Prabhdeep SINGH**, Bellevue, WA (US); **Christian BERG**, Seattle, WA (US)  
(73) Assignee: **UiPath, Inc.**, New York, NY (US)

(21) Appl. No.: **18/528,023**  
(22) Filed: **Dec. 4, 2023**

**Related U.S. Application Data**

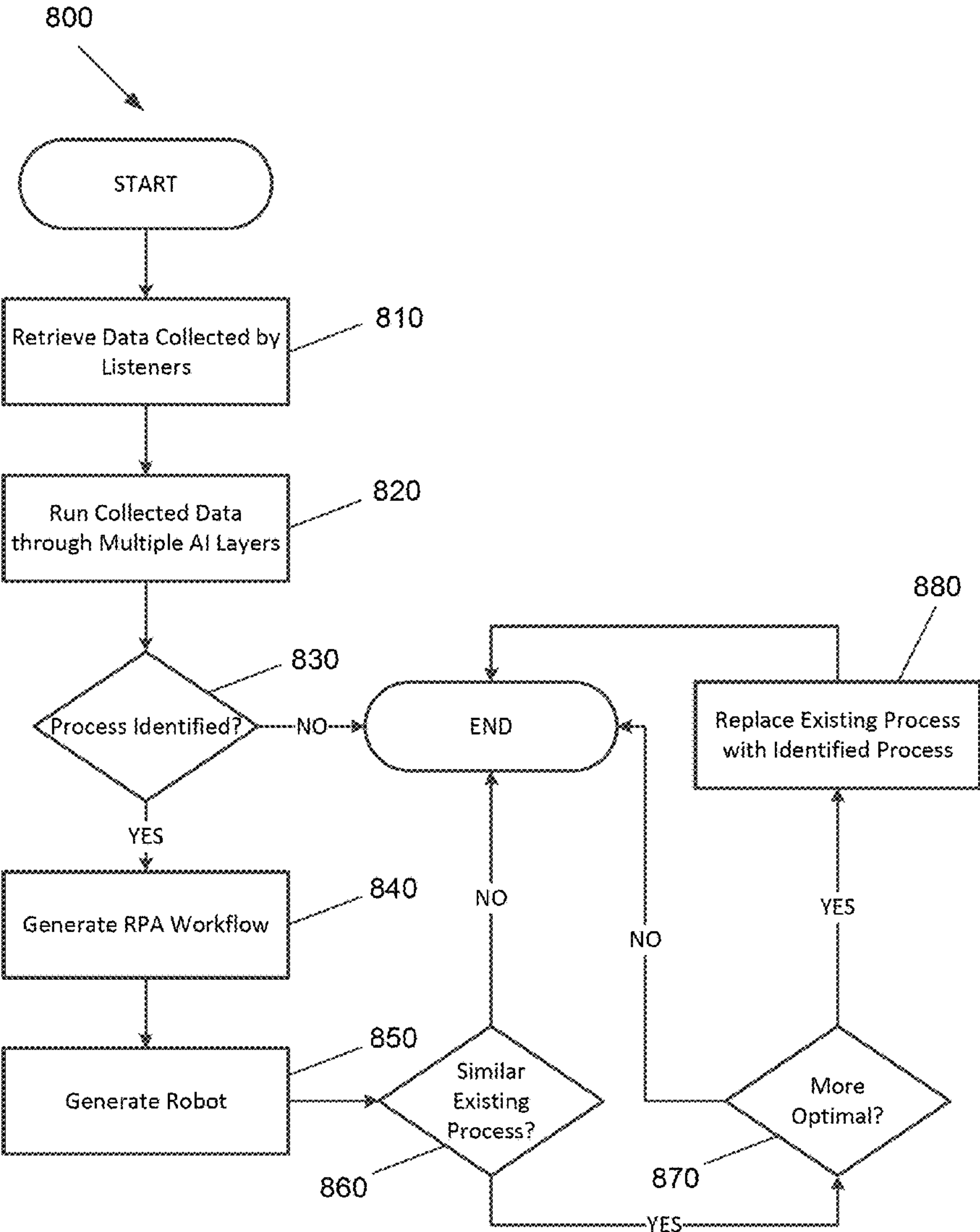
(63) Continuation of application No. 18/051,822, filed on Nov. 1, 2022, now Pat. No. 11,836,626, which is a continuation of application No. 16/707,705, filed on Dec. 9, 2019, now Pat. No. 11,488,015.

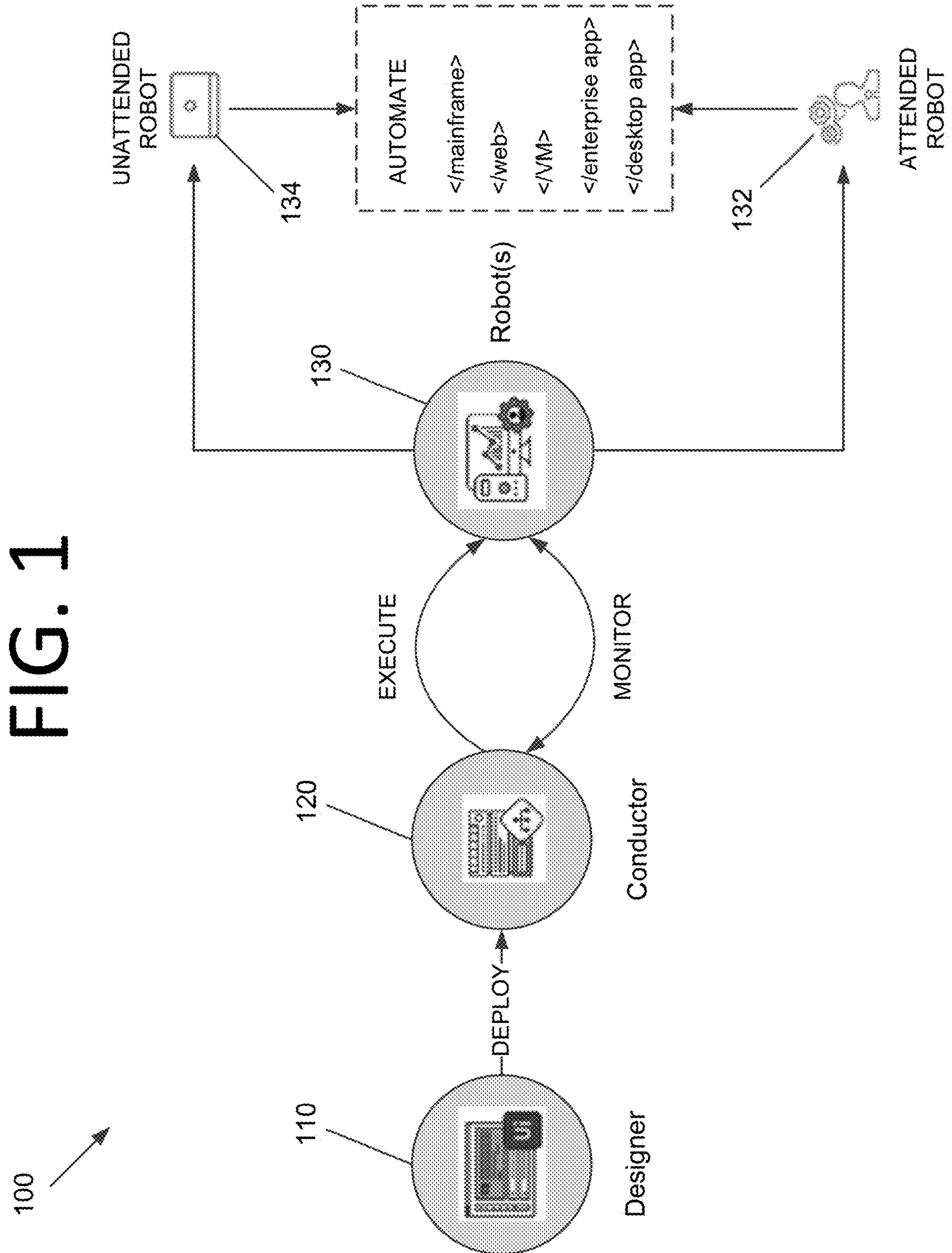
**Publication Classification**

(51) **Int. Cl.**  
**G06N 3/08** (2006.01)  
(52) **U.S. Cl.**  
CPC ..... **G06N 3/08** (2013.01)

(57) **ABSTRACT**

Artificial intelligence (AI) layer-based process extraction for robotic process automation (RPA) is disclosed. Data collected by RPA robots and/or other sources may be analyzed to identify patterns that can be used to suggest or automatically generate RPA workflows. These AI layers may be used to recognize patterns of user or business system processes contained therein. Each AI layer may “sense” different characteristics in the data and be used individually or in concert with other AI layers to suggest RPA workflows.







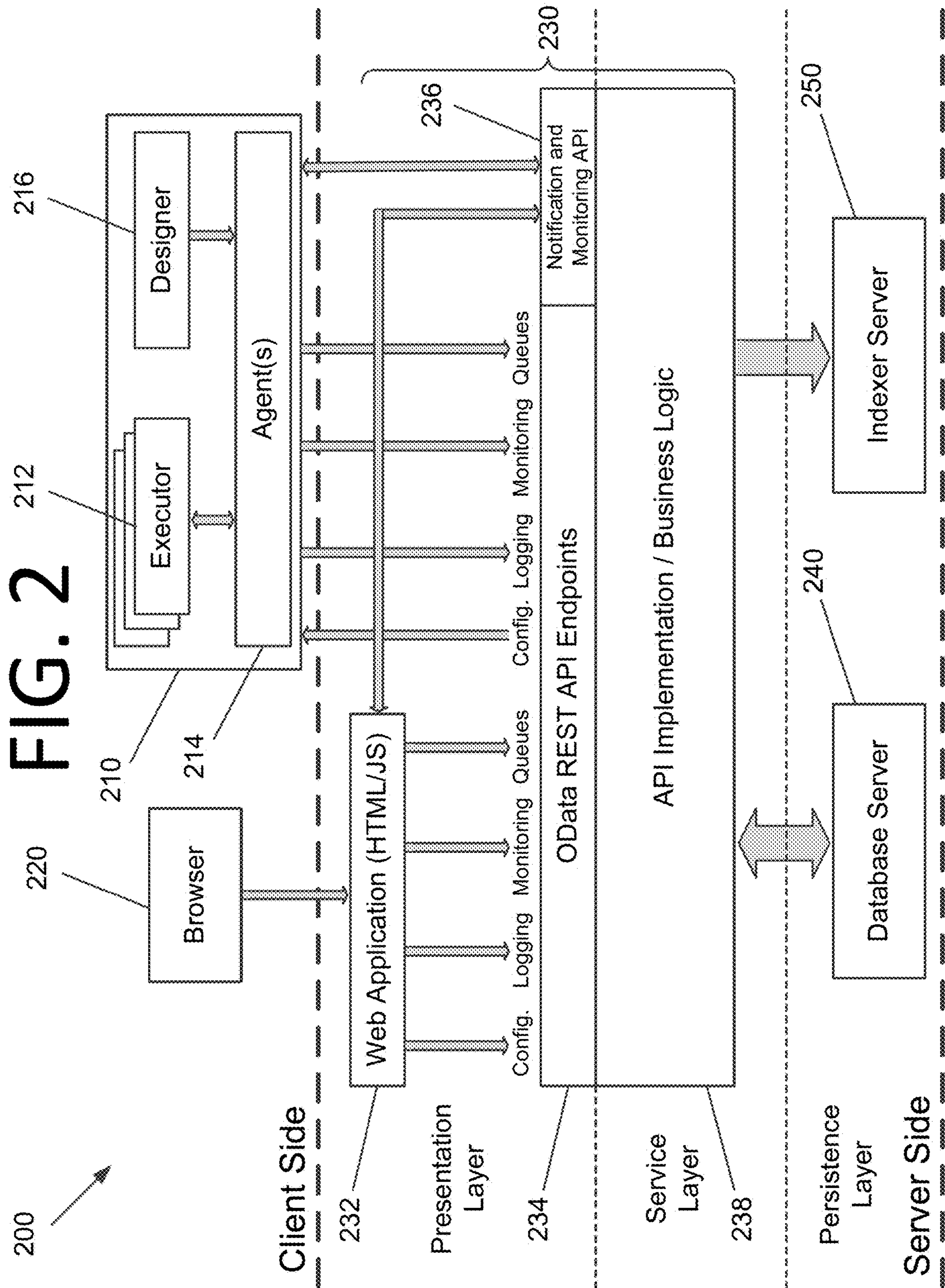
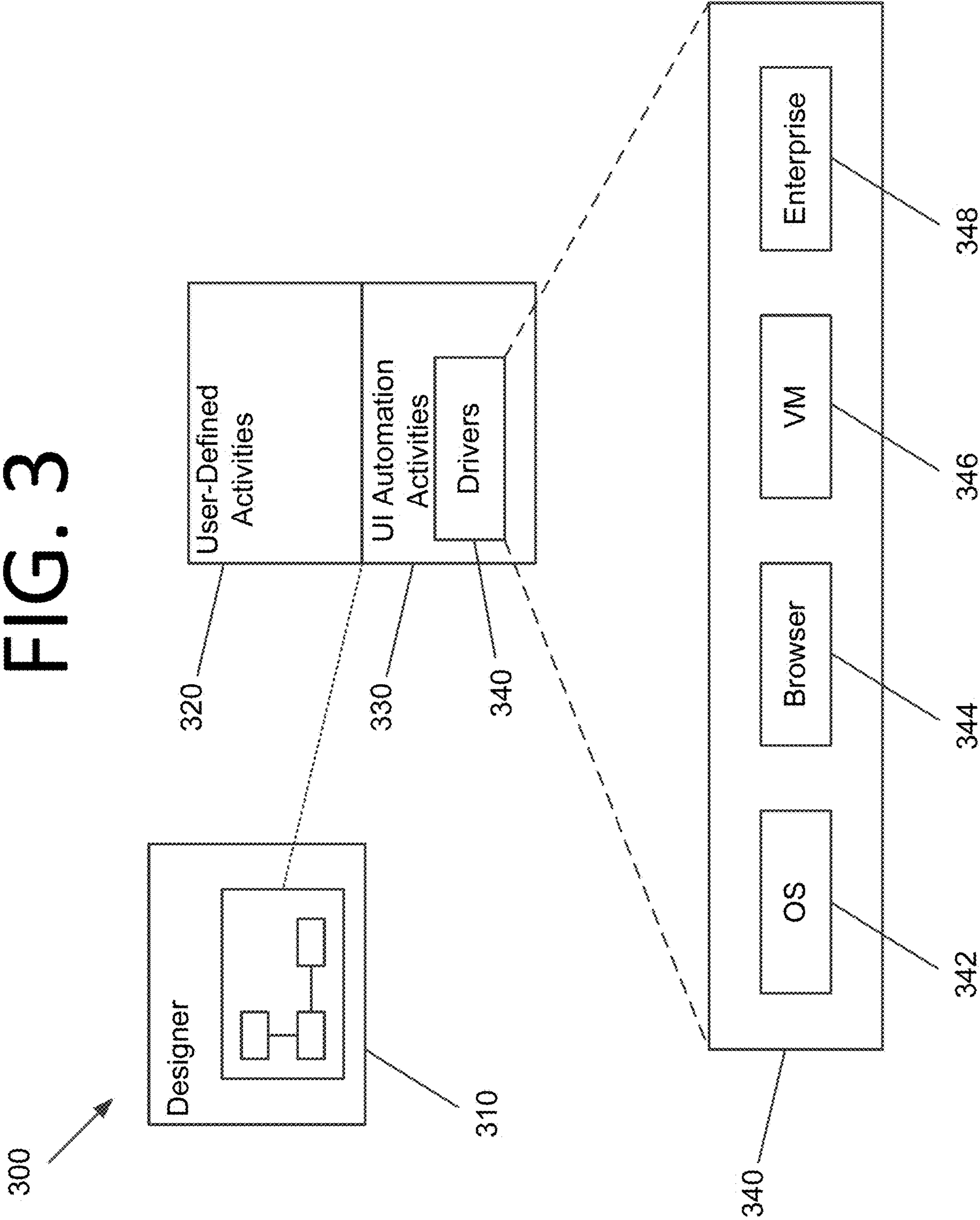


FIG. 3



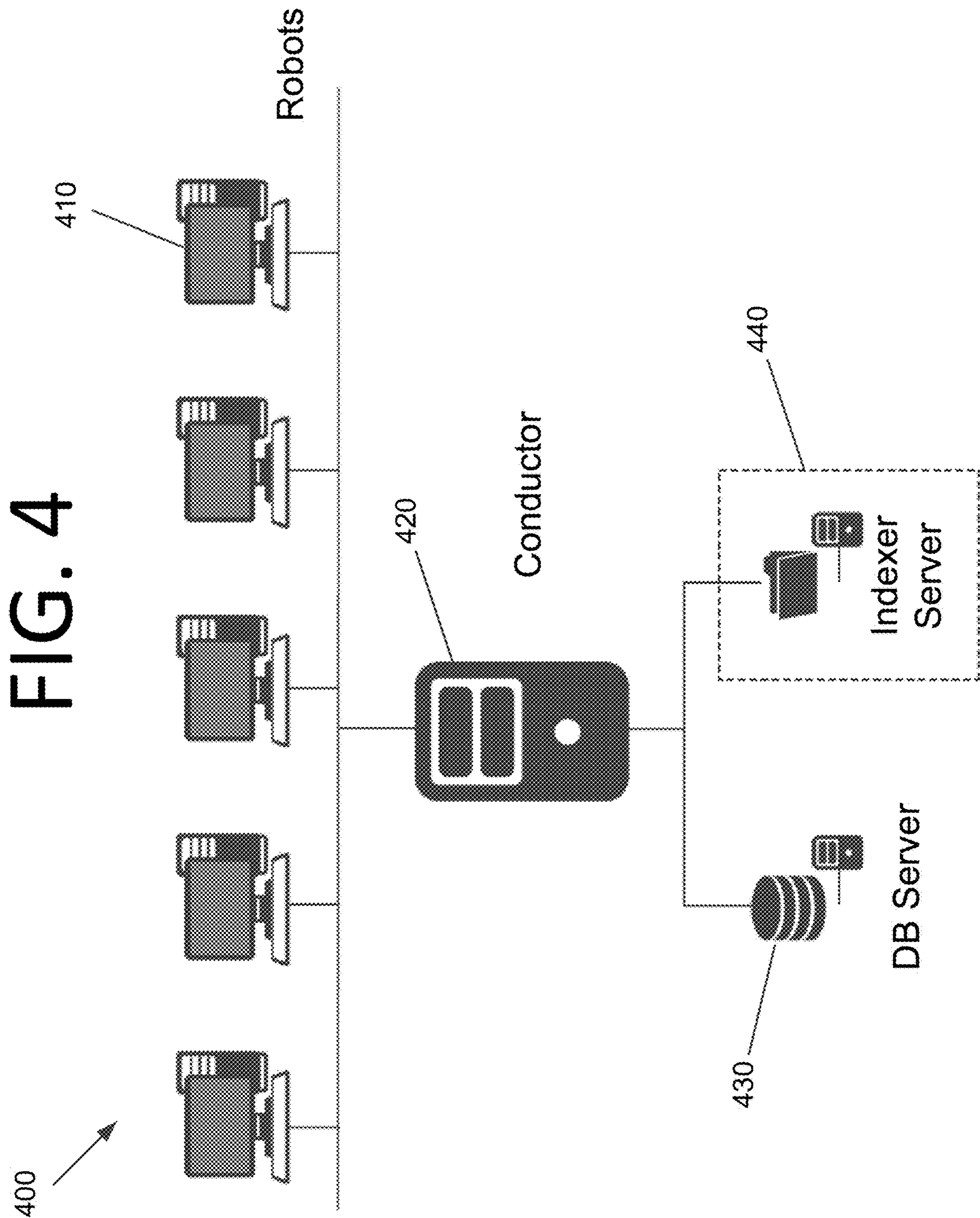
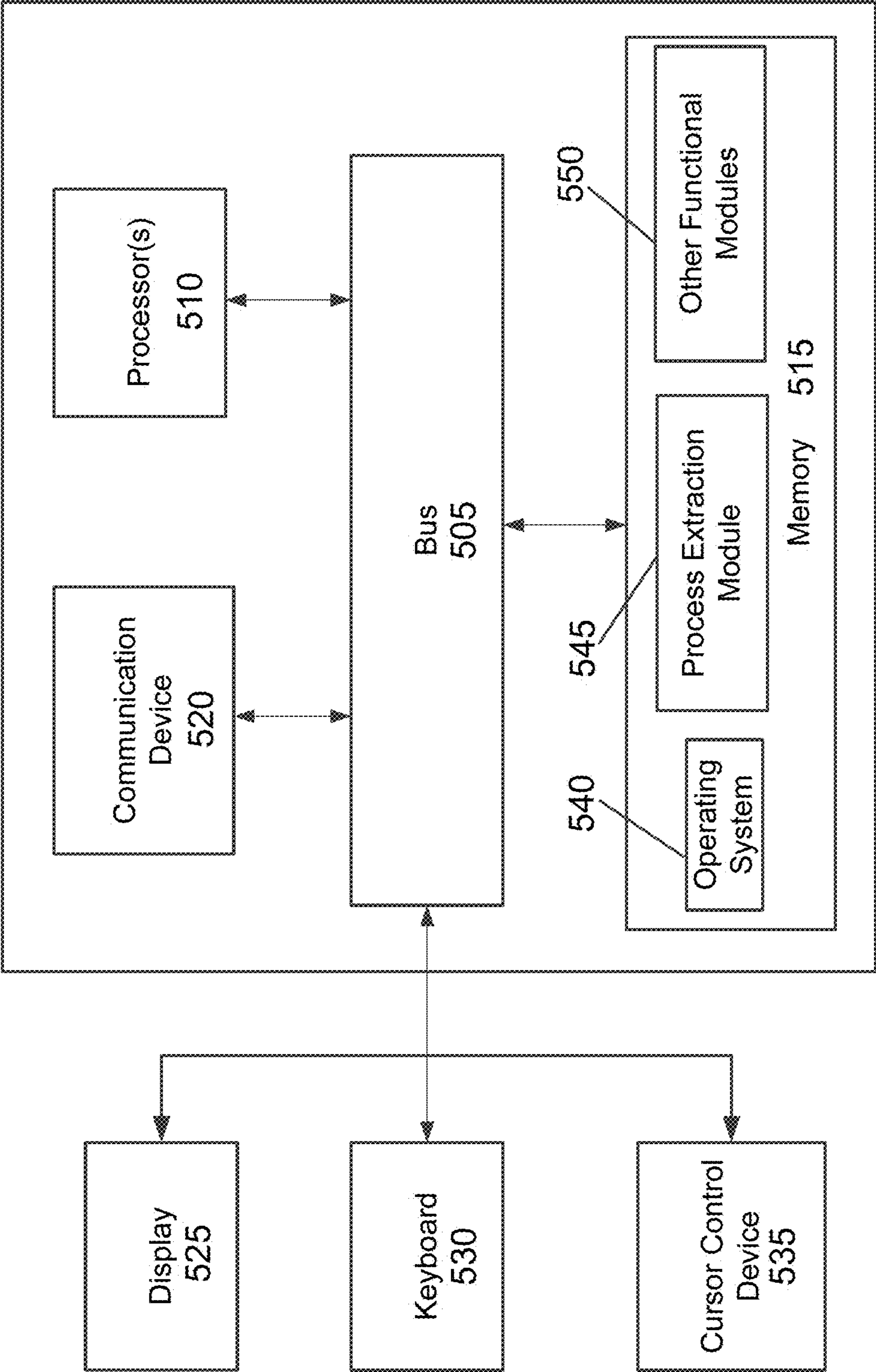
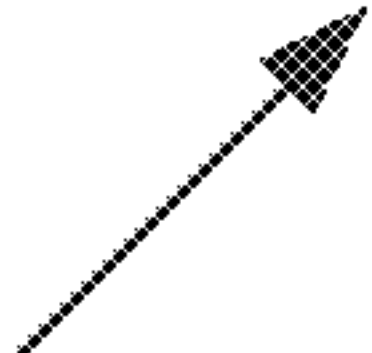




FIG. 5

500



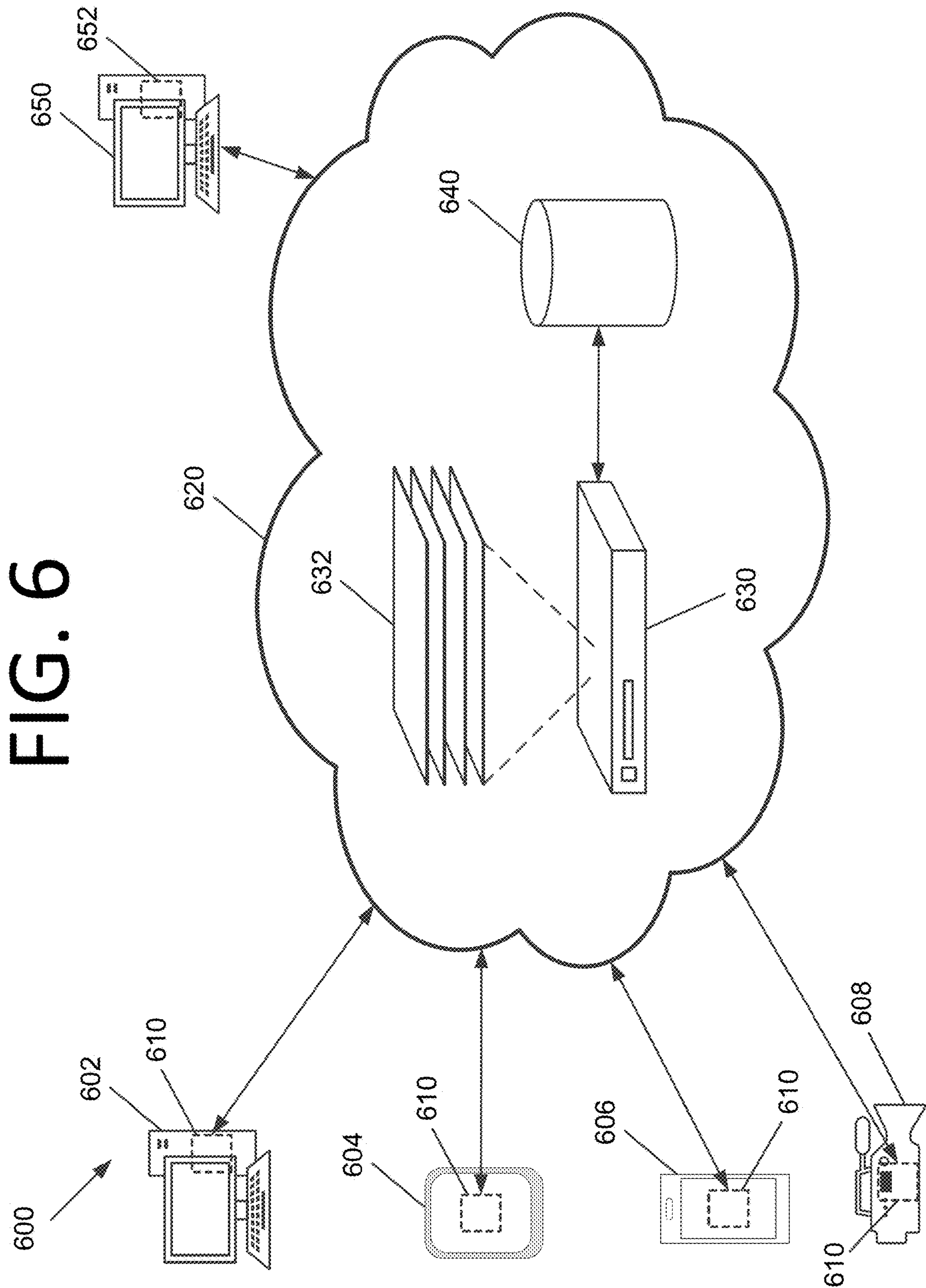


FIG. 7

700

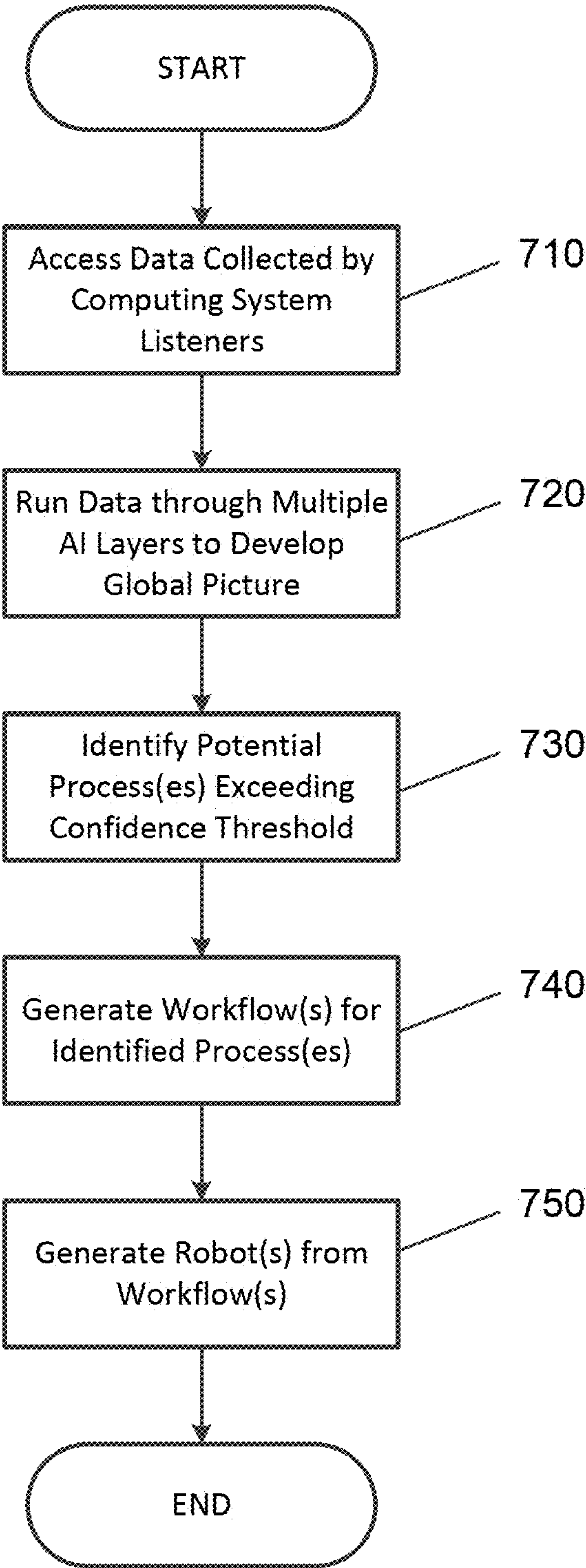
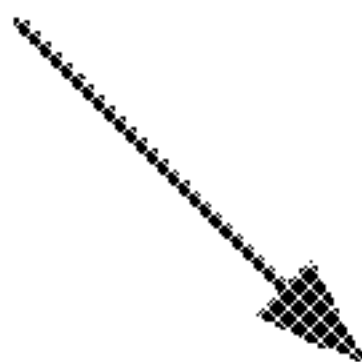
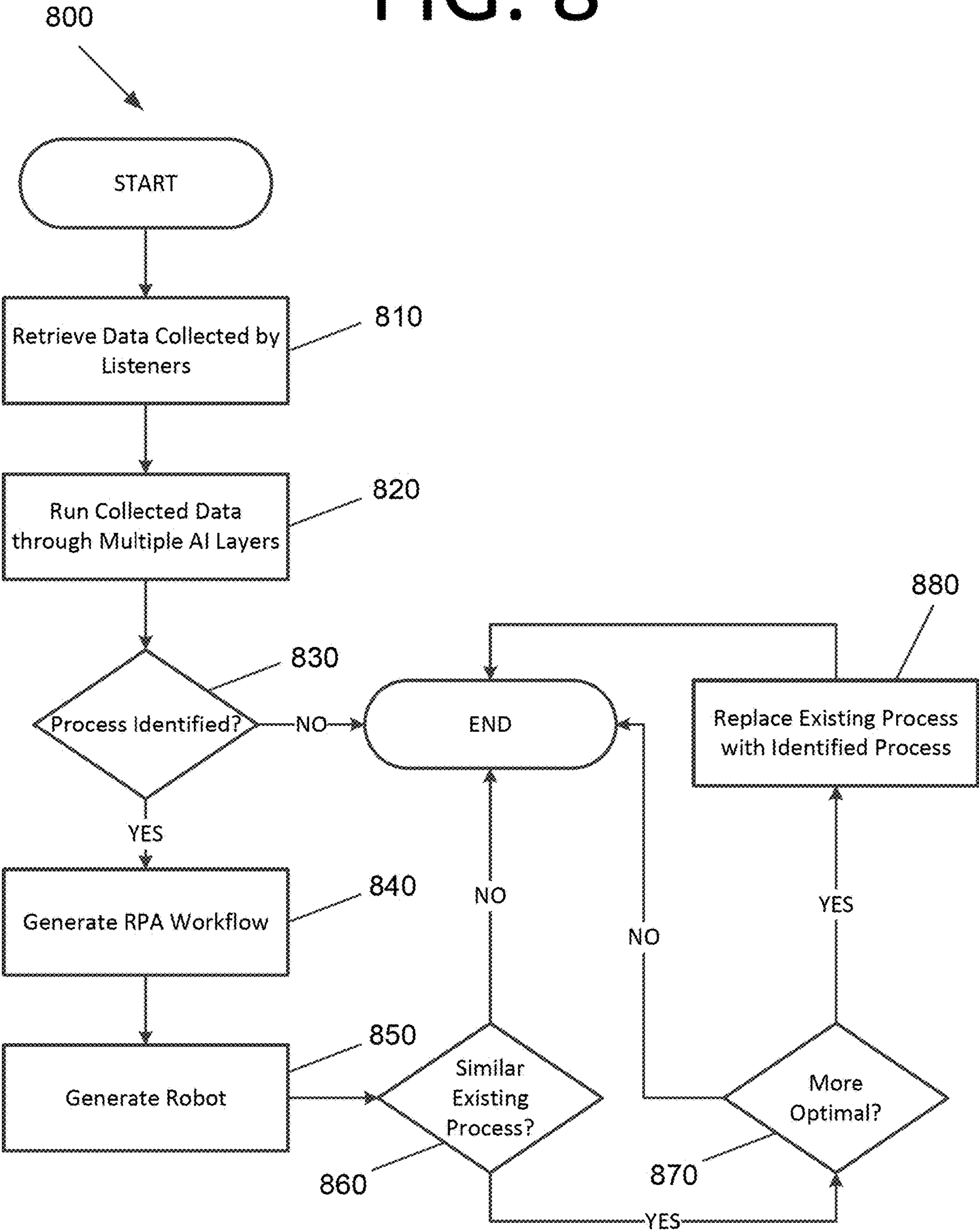




FIG. 8



# ARTIFICIAL INTELLIGENCE LAYER-BASED PROCESS EXTRACTION FOR ROBOTIC PROCESS AUTOMATION

## CROSS-REFERENCE TO RELATED APPLICATION

**[0001]** This application is a continuation of U.S. Nonprovisional patent application Ser. No. 18/051,822 filed Nov. 1, 2022, and issued as U.S. Pat. No. 11,836,626, which is a continuation of U.S. Nonprovisional patent application Ser. No. 16/707,705 filed Dec. 9, 2019, and issued as U.S. Pat. No. 11,488,015 on Nov. 1, 2022, which claims the benefit of, and priority to, U.S. Provisional Patent Application No. 62/915,351 filed Oct. 15, 2019. The subject matter of these earlier filed applications is hereby incorporated by reference in its entirety.

## FIELD

**[0002]** The present invention generally relates to robotic process automation (RPA), and more specifically, to artificial intelligence (AI) layer-based process extraction for RPA.

## BACKGROUND

**[0003]** Large amounts of data may be collected from business computing systems using RPA robots. However, this data does not facilitate RPA or improvements thereto in and of itself. Indeed, businesses may or may not be aware of processes that could benefit from RPA. Accordingly, an improved mechanism for analyzing data for RPA and extracting processes therefrom may be beneficial.

## SUMMARY

**[0004]** Certain embodiments of the present invention may provide solutions to the problems and needs in the art that have not yet been fully identified, appreciated, or solved by current RPA techniques. For example, some embodiments of the present invention pertain to AI layer-based process extraction for RPA.

**[0005]** In an embodiment, a computer-implemented method for performing AI-based process extraction for RPA using multiple AI layers includes retrieving data collected by listeners, by a computing system. The computer-implemented method also includes running the data collected by the listeners through the multiple AI layers, by the computing system, to process the collected data. When a potential RPA process is identified that exceeds the confidence threshold, the computer-implemented method further includes generating an RPA workflow for the identified RPA process, by the computing system and generating a robot that is configured to execute the generated RPA workflow for the identified RPA process, by the computing system.

**[0006]** In another embodiment, a computer program is embodied on a nontransitory computer-readable medium. The program is configured to cause at least one processor to run the data collected by a plurality of listeners through multiple AI layers to process the collected data and identify one or more potential RPA processes in the collected data that exceed a confidence threshold. The program is also configured to cause the at least one processor to generate an RPA workflow for the identified RPA process when the identified RPA process exceeds the confidence threshold.

**[0007]** In yet another embodiment, an apparatus includes memory storing computer program instructions for performing AI-based process extraction for RPA using multiple AI layers and at least one processor communicably coupled to the memory and configured to execute the computer program instructions. The instructions are configured to cause the at least one processor to run the data collected by a plurality of listeners through multiple AI layers to process the collected data and identify one or more RPA processes in the collected data that exceed a confidence threshold. The instructions are also configured to cause the at least one processor to generate an RPA workflow for the identified RPA process when the identified RPA process exceeds the confidence threshold. The collected data is run through the multiple AI layers in series, in parallel, or a combination of both series and in parallel.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0008]** In order that the advantages of certain embodiments of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. While it should be understood that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings, in which:

**[0009]** FIG. 1 is an architectural diagram illustrating an RPA system, according to an embodiment of the present invention.

**[0010]** FIG. 2 is an architectural diagram illustrating a deployed RPA system, according to an embodiment of the present invention.

**[0011]** FIG. 3 is an architectural diagram illustrating the relationship between a designer, activities, and drivers, according to an embodiment of the present invention.

**[0012]** FIG. 4 is an architectural diagram illustrating an RPA system, according to an embodiment of the present invention.

**[0013]** FIG. 5 is an architectural diagram illustrating a computing system configured to perform multiple AI layer-based process extraction for RPA, according to an embodiment of the present invention.

**[0014]** FIG. 6 is an architectural diagram illustrating a system configured to perform AI-based process extraction for RPA using multiple AI layers, according to an embodiment of the present invention.

**[0015]** FIG. 7 is a flowchart illustrating a process for AI-based process extraction for RPA using AI layers, according to an embodiment of the present invention.

**[0016]** FIG. 8 is a flowchart illustrating a process for AI-based process extraction for RPA using AI layers, according to an embodiment of the present invention.

## DETAILED DESCRIPTION OF THE EMBODIMENTS

**[0017]** Some embodiments pertain to AI layer-based process extraction for RPA. Such embodiments may analyze data collected by RPA robots and/or other sources to identify patterns that can be used to suggest or automatically generate RPA workflows. These AI layers may be used to



recognize patterns of user or business system processes contained therein. Each AI layer may “sense” different characteristics in the data and be used individually or in concert with other AI layers to suggest RPA workflows.

[0018] FIG. 1 is an architectural diagram illustrating an RPA system 100, according to an embodiment of the present invention. RPA system 100 includes a designer 110 that allows a developer to design and implement workflows. Designer 110 may provide a solution for application integration, as well as automating third-party applications, administrative Information Technology (IT) tasks, and business IT processes. Designer 110 may facilitate development of an automation project, which is a graphical representation of a business process. Simply put, designer 110 facilitates the development and deployment of workflows and robots.

[0019] The automation project enables automation of rule-based processes by giving the developer control of the execution order and the relationship between a custom set of steps developed in a workflow, defined herein as “activities.” One commercial example of an embodiment of designer 110 is UiPath Studio™. Each activity may include an action, such as clicking a button, reading a file, writing to a log panel, etc. In some embodiments, workflows may be nested or embedded.

[0020] Some types of workflows may include, but are not limited to, sequences, flowcharts, Finite State Machines (FSMs), and/or global exception handlers. Sequences may be particularly suitable for linear processes, enabling flow from one activity to another without cluttering a workflow. Flowcharts may be particularly suitable to more complex business logic, enabling integration of decisions and connection of activities in a more diverse manner through multiple branching logic operators. FSMs may be particularly suitable for large workflows. FSMs may use a finite number of states in their execution, which are triggered by a condition (i.e., transition) or an activity. Global exception handlers may be particularly suitable for determining workflow behavior when encountering an execution error and for debugging processes.

[0021] Once a workflow is developed in designer 110, execution of business processes is orchestrated by conductor 120, which orchestrates one or more robots 130 that execute the workflows developed in designer 110. One commercial example of an embodiment of conductor 120 is UiPath Orchestrator™. Conductor 120 facilitates management of the creation, monitoring, and deployment of resources in an environment. Conductor 120 may act as an integration point with third-party solutions and applications.

[0022] Conductor 120 may manage a fleet of robots 130, connecting and executing robots 130 from a centralized point. Types of robots 130 that may be managed include, but are not limited to, attended robots 132, unattended robots 134, development robots (similar to unattended robots 134, but used for development and testing purposes), and non-production robots (similar to attended robots 132, but used for development and testing purposes). Attended robots 132 are triggered by user events and operate alongside a human on the same computing system. Attended robots 132 may be used with conductor 120 for a centralized process deployment and logging medium. Attended robots 132 may help the human user accomplish various tasks, and may be triggered by user events. In some embodiments, processes cannot be started from conductor 120 on this type of robot and/or they cannot run under a locked screen. In certain

embodiments, attended robots 132 can only be started from a robot tray or from a command prompt. Attended robots 132 should run under human supervision in some embodiments.

[0023] Unattended robots 134 run unattended in virtual environments and can automate many processes. Unattended robots 134 may be responsible for remote execution, monitoring, scheduling, and providing support for work queues. Debugging for all robot types may be run in designer 110 in some embodiments. Both attended and unattended robots may automate various systems and applications including, but not limited to, mainframes, web applications, VMs, enterprise applications (e.g., those produced by SAP®, Salesforce®, Oracle®, etc.), and computing system applications (e.g., desktop and laptop applications, mobile device applications, wearable computer applications, etc.).

[0024] Conductor 120 may have various capabilities including, but not limited to, provisioning, deployment, configuration, queueing, monitoring, logging, and/or providing interconnectivity. Provisioning may include creating and maintenance of connections between robots 130 and conductor 120 (e.g., a web application). Deployment may include assuring the correct delivery of package versions to assigned robots 130 for execution. Configuration may include maintenance and delivery of robot environments and process configurations. Queueing may include providing management of queues and queue items. Monitoring may include keeping track of robot identification data and maintaining user permissions. Logging may include storing and indexing logs to a database (e.g., an SQL database) and/or another storage mechanism (e.g., Elasticsearch®, which provides the ability to store and quickly query large data-sets). Conductor 120 may provide interconnectivity by acting as the centralized point of communication for third-party solutions and/or applications.

[0025] Robots 130 are execution agents that run workflows built in designer 110. One commercial example of some embodiments of robot(s) 130 is UiPath Robots™. In some embodiments, robots 130 install the Microsoft Windows® Service Control Manager (SCM)-managed service by default. As a result, such robots 130 can open interactive Windows® sessions under the local system account, and have the rights of a Windows® service.

[0026] In some embodiments, robots 130 can be installed in a user mode. For such robots 130, this means they have the same rights as the user under which a given robot 130 has been installed. This feature may also be available for High Density (HD) robots, which ensure full utilization of each machine at its maximum potential. In some embodiments, any type of robot 130 may be configured in an HD environment.

[0027] Robots 130 in some embodiments are split into several components, each being dedicated to a particular automation task. The robot components in some embodiments include, but are not limited to, SCM-managed robot services, user mode robot services, executors, agents, and command line. SCM-managed robot services manage and monitor Windows® sessions and act as a proxy between conductor 120 and the execution hosts (i.e., the computing systems on which robots 130 are executed). These services are trusted with and manage the credentials for robots 130. A console application is launched by the SCM under the local system.



**[0028]** User mode robot services in some embodiments manage and monitor Windows® sessions and act as a proxy between conductor **120** and the execution hosts. User mode robot services may be trusted with and manage the credentials for robots **130**. A Windows® application may automatically be launched if the SCM-managed robot service is not installed.

**[0029]** Executors may run given jobs under a Windows® session (i.e., they may execute workflows. Executors may be aware of per-monitor dots per inch (DPI) settings. Agents may be Windows® Presentation Foundation (WPF) applications that display the available jobs in the system tray window. Agents may be a client of the service. Agents may request to start or stop jobs and change settings. The command line is a client of the service. The command line is a console application that can request to start jobs and waits for their output.

**[0030]** Having components of robots **130** split as explained above helps developers, support users, and computing systems more easily run, identify, and track what each component is executing. Special behaviors may be configured per component this way, such as setting up different firewall rules for the executor and the service. The executor may always be aware of DPI settings per monitor in some embodiments. As a result, workflows may be executed at any DPI, regardless of the configuration of the computing system on which they were created. Projects from designer **110** may also be independent of browser zoom level in some embodiments. For applications that are DPI-unaware or intentionally marked as unaware, DPI may be disabled in some embodiments.

**[0031]** FIG. 2 is an architectural diagram illustrating a deployed RPA system **200**, according to an embodiment of the present invention. In some embodiments, RPA system **200** may be, or may be a part of, RPA system **100** of FIG. 1. It should be noted that the client side, the server side, or both, may include any desired number of computing systems without deviating from the scope of the invention. On the client side, a robot application **210** includes executors **212**, an agent **214**, and a designer **216**. However, in some embodiments, designer **216** may not be running on computing system **210**. Executors **212** are running processes. Several business projects may run simultaneously, as shown in FIG. 2. Agent **214** (e.g., a Windows® service) is the single point of contact for all executors **212** in this embodiment. All messages in this embodiment are logged into conductor **230**, which processes them further via database server **240**, indexer server **250**, or both. As discussed above with respect to FIG. 1, executors **212** may be robot components.

**[0032]** In some embodiments, a robot represents an association between a machine name and a username. The robot may manage multiple executors at the same time. On computing systems that support multiple interactive sessions running simultaneously (e.g., Windows® Server 2012), there may be multiple robots running at the same time, each in a separate Windows® session using a unique username. This is referred to as HD robots above.

**[0033]** Agent **214** is also responsible for sending the status of the robot (e.g., periodically sending a “heartbeat” message indicating that the robot is still functioning) and downloading the required version of the package to be executed. The communication between agent **214** and conductor **230** is always initiated by agent **214** in some embodiments. In the notification scenario, agent **214** may open a WebSocket

channel that is later used by conductor **230** to send commands to the robot (e.g., start, stop, etc.).

**[0034]** On the server side, a presentation layer (web application **232**, Open Data Protocol (OData) Representative State Transfer (REST) Application Programming Interface (API) endpoints **234**, and notification and monitoring **236**), a service layer (API implementation/business logic **238**), and a persistence layer (database server **240** and indexer server **250**) are included. Conductor **230** includes web application **232**, OData REST API endpoints **234**, notification and monitoring **236**, and API implementation/business logic **238**. In some embodiments, most actions that a user performs in the interface of conductor **220** (e.g., via browser **220**) are performed by calling various APIs. Such actions may include, but are not limited to, starting jobs on robots, adding/removing data in queues, scheduling jobs to run unattended, etc. without deviating from the scope of the invention. Web application **232** is the visual layer of the server platform. In this embodiment, web application **232** uses Hypertext Markup Language (HTML) and JavaScript (JS). However, any desired markup languages, script languages, or any other formats may be used without deviating from the scope of the invention. The user interacts with web pages from web application **232** via browser **220** in this embodiment in order to perform various actions to control conductor **230**. For instance, the user may create robot groups, assign packages to the robots, analyze logs per robot and/or per process, start and stop robots, etc.

**[0035]** In addition to web application **232**, conductor **230** also includes service layer that exposes OData REST API endpoints **234**. However, other endpoints may be included without deviating from the scope of the invention. The REST API is consumed by both web application **232** and agent **214**. Agent **214** is the supervisor of one or more robots on the client computer in this embodiment.

**[0036]** The REST API in this embodiment covers configuration, logging, monitoring, and queueing functionality. The configuration endpoints may be used to define and configure application users, permissions, robots, assets, releases, and environments in some embodiments. Logging REST endpoints may be used to log different information, such as errors, explicit messages sent by the robots, and other environment-specific information, for instance. Deployment REST endpoints may be used by the robots to query the package version that should be executed if the start job command is used in conductor **230**. Queueing REST endpoints may be responsible for queues and queue item management, such as adding data to a queue, obtaining a transaction from the queue, setting the status of a transaction, etc.

**[0037]** Monitoring REST endpoints may monitor web application **232** and agent **214**. Notification and monitoring API **236** may be REST endpoints that are used for registering agent **214**, delivering configuration settings to agent **214**, and for sending/receiving notifications from the server and agent **214**. Notification and monitoring API **236** may also use WebSocket communication in some embodiments.

**[0038]** The persistence layer includes a pair of servers in this embodiment—database server **240** (e.g., a SQL server) and indexer server **250**. Database server **240** in this embodiment stores the configurations of the robots, robot groups, associated processes, users, roles, schedules, etc. This information is managed through web application **232** in some embodiments. Database server **240** may manages queues



and queue items. In some embodiments, database server **240** may store messages logged by the robots (in addition to or in lieu of indexer server **250**).

[0039] Indexer server **250**, which is optional in some embodiments, stores and indexes the information logged by the robots. In certain embodiments, indexer server **250** may be disabled through configuration settings. In some embodiments, indexer server **250** uses ElasticSearch®, which is an open source project full-text search engine. Messages logged by robots (e.g., using activities like log message or write line) may be sent through the logging REST endpoint (s) to indexer server **250**, where they are indexed for future utilization.

[0040] FIG. 3 is an architectural diagram illustrating the relationship **300** between a designer **310**, activities **320**, **330**, and drivers **340**, according to an embodiment of the present invention. Per the above, a developer uses designer **310** to develop workflows that are executed by robots. Workflows may include user-defined activities **320** and UI automation activities **330**. Some embodiments are able to identify non-textual visual components in an image, which is called computer vision (CV) herein. Some CV activities pertaining to such components may include, but are not limited to, click, type, get text, hover, element exists, refresh scope, highlight, etc. Click in some embodiments identifies an element using CV, optical character recognition (OCR), fuzzy text matching, and multi-anchor, for example, and clicks it. Type may identify an element using the above and types in the element. Get text may identify the location of specific text and scan it using OCR. Hover may identify an element and hover over it. Element exists may check whether an element exists on the screen using the techniques described above. In some embodiments, there may be hundreds or even thousands of activities that can be implemented in designer **310**. However, any number and/or type of activities may be available without deviating from the scope of the invention.

[0041] UI automation activities **330** are a subset of special, lower level activities that are written in lower level code (e.g., CV activities) and facilitate interactions with the screen. UI automation activities **330** facilitate these interactions via drivers **340** that allow the robot to interact with the desired software. For instance, drivers **340** may include OS drivers **342**, browser drivers **344**, VM drivers **346**, enterprise application drivers **348**, etc.

[0042] Drivers **340** may interact with the OS at a low level looking for hooks, monitoring for keys, etc. They may facilitate integration with Chrome®, IE®, Citrix®, SAP®, etc. For instance, the “click” activity performs the same role in these different applications via drivers **340**.

[0043] FIG. 4 is an architectural diagram illustrating an RPA system **400**, according to an embodiment of the present invention. In some embodiments, RPA system **400** may be or include RPA systems **100** and/or **200** of FIGS. 1 and/or 2. RPA system **400** includes multiple client computing systems **410** running robots. Computing systems **410** are able to communicate with a conductor computing system **420** via a web application running thereon. Conductor computing system **420**, in turn, is able to communicate with a database server **430** and an optional indexer server **440**.

[0044] With respect to FIGS. 1 and 3, it should be noted that while a web application is used in these embodiments, any suitable client/server software may be used without deviating from the scope of the invention. For instance, the

conductor may run a server-side application that communicates with non-web-based client software applications on the client computing systems.

[0045] FIG. 5 is an architectural diagram illustrating a computing system **500** configured to perform multiple AI layer-based process extraction for RPA, according to an embodiment of the present invention. In some embodiments, computing system **500** may be one or more of the computing systems depicted and/or described herein. Computing system **500** includes a bus **505** or other communication mechanism for communicating information, and processor(s) **510** coupled to bus **505** for processing information. Processor(s) **510** may be any type of general or specific purpose processor, including a Central Processing Unit (CPU), an Application Specific Integrated Circuit (ASIC), a Field Programmable Gate Array (FPGA), a Graphics Processing Unit (GPU), multiple instances thereof, and/or any combination thereof. Processor(s) **510** may also have multiple processing cores, and at least some of the cores may be configured to perform specific functions. Multi-parallel processing may be used in some embodiments. In certain embodiments, at least one of processor(s) **510** may be a neuromorphic circuit that includes processing elements that mimic biological neurons. In some embodiments, neuromorphic circuits may not require the typical components of a Von Neumann computing architecture.

[0046] Computing system **500** further includes a memory **515** for storing information and instructions to be executed by processor(s) **510**. Memory **515** can be comprised of any combination of Random Access Memory (RAM), Read Only Memory (ROM), flash memory, cache, static storage such as a magnetic or optical disk, or any other types of non-transitory computer-readable media or combinations thereof. Non-transitory computer-readable media may be any available media that can be accessed by processor(s) **510** and may include volatile media, non-volatile media, or both. The media may also be removable, non-removable, or both.

[0047] Additionally, computing system **500** includes a communication device **520**, such as a transceiver, to provide access to a communications network via a wireless and/or wired connection. In some embodiments, communication device **520** may be configured to use Frequency Division Multiple Access (FDMA), Single Carrier FDMA (SC-FDMA), Time Division Multiple Access (TDMA), Code Division Multiple Access (CDMA), Orthogonal Frequency Division Multiplexing (OFDM), Orthogonal Frequency Division Multiple Access (OFDMA), Global System for Mobile (GSM) communications, General Packet Radio Service (GPRS), Universal Mobile Telecommunications System (UMTS), cdma2000, Wideband CDMA (W-CDMA), High-Speed Downlink Packet Access (HSDPA), High-Speed Uplink Packet Access (HSUPA), High-Speed Packet Access (HSPA), Long Term Evolution (LTE), LTE Advanced (LTE-A), 802.11x, Wi-Fi, Zigbee, Ultra-Wide-Band (UWB), 802.16x, 802.15, Home Node-B (HnB), Bluetooth, Radio Frequency Identification (RFID), Infrared Data Association (IrDA), Near-Field Communications (NFC), fifth generation (5G), New Radio (NR), any combination thereof, and/or any other currently existing or future-implemented communications standard and/or protocol without deviating from the scope of the invention. In some embodiments, communication device **520** may include one or more antennas that are singular, arrayed, phased,



switched, beamforming, beamsteering, a combination thereof, and or any other antenna configuration without deviating from the scope of the invention.

**[0048]** Processor(s) **510** are further coupled via bus **505** to a display **525**, such as a plasma display, a Liquid Crystal Display (LCD), a Light Emitting Diode (LED) display, a Field Emission Display (FED), an Organic Light Emitting Diode (OLED) display, a flexible OLED display, a flexible substrate display, a projection display, a 4K display, a high definition display, a Retina® display, an In-Plane Switching (IPS) display, or any other suitable display for displaying information to a user. Display **525** may be configured as a touch (haptic) display, a three dimensional (3D) touch display, a multi-input touch display, a multi-touch display, etc. using resistive, capacitive, surface-acoustic wave (SAW) capacitive, infrared, optical imaging, dispersive signal technology, acoustic pulse recognition, frustrated total internal reflection, etc. Any suitable display device and haptic I/O may be used without deviating from the scope of the invention.

**[0049]** A keyboard **530** and a cursor control device **535**, such as a computer mouse, a touchpad, etc., are further coupled to bus **505** to enable a user to interface with computing system **500**. However, in certain embodiments, a physical keyboard and mouse may not be present, and the user may interact with the device solely through display **525** and/or a touchpad (not shown). Any type and combination of input devices may be used as a matter of design choice. In certain embodiments, no physical input device and/or display is present. For instance, the user may interact with computing system **500** remotely via another computing system in communication therewith, or computing system **500** may operate autonomously.

**[0050]** Memory **515** stores software modules that provide functionality when executed by processor(s) **510**. The modules include an operating system **540** for computing system **500**. The modules further include a process extraction module **545** that is configured to perform all or part of the processes described herein or derivatives thereof. Computing system **500** may include one or more additional functional modules **550** that include additional functionality.

**[0051]** One skilled in the art will appreciate that a “system” could be embodied as a server, an embedded computing system, a personal computer, a console, a personal digital assistant (PDA), a cell phone, a tablet computing device, a quantum computing system, or any other suitable computing device, or combination of devices without deviating from the scope of the invention. Presenting the above-described functions as being performed by a “system” is not intended to limit the scope of the present invention in any way, but is intended to provide one example of the many embodiments of the present invention. Indeed, methods, systems, and apparatuses disclosed herein may be implemented in localized and distributed forms consistent with computing technology, including cloud computing systems.

**[0052]** It should be noted that some of the system features described in this specification have been presented as modules, in order to more particularly emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom very large scale integration (VLSI) circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field

programmable gate arrays, programmable array logic, programmable logic devices, graphics processing units, or the like.

**[0053]** A module may also be at least partially implemented in software for execution by various types of processors. An identified unit of executable code may, for instance, include one or more physical or logical blocks of computer instructions that may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may include disparate instructions stored in different locations that, when joined logically together, comprise the module and achieve the stated purpose for the module. Further, modules may be stored on a computer-readable medium, which may be, for instance, a hard disk drive, flash device, RAM, tape, and/or any other such non-transitory computer-readable medium used to store data without deviating from the scope of the invention.

**[0054]** Indeed, a module of executable code could be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

**[0055]** FIG. 6 is an architectural diagram illustrating a system **600** configured to perform AI-based process extraction for RPA using multiple AI layers, according to an embodiment of the present invention. System **600** can include any desired data source, such as desktop computer **602**, tablet **604**, smart phone **606**, and video camera **608**. However, any desired computing system may be used without deviating from the scope of invention including, but not limited to, smart watches, laptop computers, Internet-of-Things (IoT) devices, vehicle computing systems, listening devices with microphones, electronic sensors, etc.

**[0056]** Each computing system **602**, **604**, **606**, **608** has a listener **610** installed thereon in this embodiment. Listeners **610** may be robots generated via an RPA designer application, part of an operating system, a downloadable application for a personal computer (PC) or smart phone, or any other software and/or hardware without deviating from the scope of the invention. Indeed, in some embodiments, the logic of one or more of listeners **610** is implemented partially or completely via physical hardware.

**[0057]** Listeners **610** generate data that they then send via a network **620** (e.g., a local area network (LAN), a mobile communications network, a satellite communications network, the Internet, any combination thereof, etc.) to a server **630**. The data that is logged may include, but is not limited to, which buttons were clicked, where a mouse was moved, the text that was entered in a field, that one window was minimized and another was opened, the application associated with a window, etc. Data generated by listeners **610** may pertain to, but is not limited to, user interactions with the computing system, audio, video, light, heat, motion, acceleration, radiation, etc. In some embodiments, server **630** may run a conductor application and the data may be sent periodically as part of a heartbeat message. In certain embodiments, the data may be sent to server **630** once a



predetermined amount of data has been collected, after a predetermined time period has elapsed, or both. Server 630 stores the received data from listeners 610 in a database 640.

[0058] When instructed by a human user (e.g., an RPA engineer or a data scientist), when a predetermined amount of data has been collected, when a predetermined amount of time has passed since the last analysis, etc., server 630 accesses data stored in database 640 and runs the data through multiple AI layers 632. AI layers 632 process the log data and identify one or more potential processes therein, potentially with an associated confidence in some embodiments. In certain embodiments, each individual AI layer 632 may have an associated confidence threshold in order to provide a positive process identification. In some embodiments, a weight may be assigned to each AI layer 632, such as when AI layers 632 have different accuracies. A workflow may then be generated if the collective confidence level of AI layers 632 exceeds a certain threshold.

[0059] AI layers 632 may perform statistical modeling (e.g., hidden Markov models (HMMs)) and utilize deep learning techniques (e.g., long short term memory (LSTM) deep learning, encoding of previous hidden states, etc.) and perform case identification to identify an atomic instance of a process. For invoice processing, for example, completion of one invoice may be a case. The system thus determines where one case finishes and the next case begins. Opening an email may be the start of a case, for example, and the patterns of the cases may be analyzed to determine variations and commonalities.

[0060] In some embodiments, identified processes may be listed for a user to peruse, and may be sorted by various factors including, but not limited to, an RPA score indicating how suitable a given process is to RPA (e.g., based on complexity of the automation, execution time, perceived benefit to key performance indicators such as revenue generated, revenue saved, time saved, etc.), process name, total recording time, the number of users who executed the process, process execution time (e.g., least or most time), etc. The process workflow may be displayed when a user clicks on a given process, including steps, parameters, and interconnections. In certain embodiments, only process activities that appear to be important from a clustering perspective may be used.

[0061] If a similar process already exists, server 630 may identify this similarity and know that the identified process should replace an existing process that works less optimally. For example, similarities between processes may be determined by a common beginning and end and some amount of statistical commonality in the steps taking in between. Commonality may be determined by entropy, minimization of a process detection objective function, etc.

[0062] Entropy is a measure of the randomness in the information being processed, where the higher the entropy, the harder it is to draw any conclusions from that information. Thus, lower entropy indicates higher commonality. An objective function helps an AI algorithm to reach a “winning state” that achieves a given objective. The objective function threshold may be set automatically in some embodiments, and this may be modified during training if processes that were identified as dissimilar by the system are indicated as being similar by a user. Server 630 may then automatically generate a workflow including the identified process, generate a robot implementing the workflow (or a replacement

robot), and push the generated robot out to computing systems 602, 604, 606, 608 to be executed thereon.

[0063] Alternatively, in certain embodiments, suggested processes from AI layers 632 may be presented to an RPA engineer via a designer application 652 on a computing system 650. The RPA engineer can then review the workflow, make any desired changes, and then deploy the workflow via a robot to computing systems 602, 604, 606, 608, or cause the robot to be deployed. For example, deployment may occur via a conductor application running on server 630 or another server, which may push a robot implementing the process out to computing systems 602, 604, 606, 608. In some embodiments, this workflow deployment may be realized via automation manager functionality in the designer application, and the RPA engineer may merely click a button to implement the process in a robot.

#### AI Layers

[0064] In some embodiments, multiple AI layers may be used. Each AI layer is an algorithm (or model) that runs on the log data, and the AI model itself may be deep learning neural networks (DLNNs) of trained artificial “neurons” that are trained in training data. Layers may be run in series, in parallel, or a combination thereof. For instance, consider an embodiment that has four AI layers. In a series implementation, data may be run through layer 1, its output run through layer 2, its output run through layer 3, and its output run through layer 4. In a parallel implementation, the data may be run through layers 1-4 individually. In a combination implementation, the data may be fed through layers 1 and 2 in series and then the output run through layers 3 and 4 in parallel, the data could be run through layers 1 and 2 in parallel and the output from each of layers 1 and 2 could be fed through layers 3 and 4 in series, etc.

[0065] The AI layers may include, but are not limited to, a sequence extraction layer, a clustering detection layer, a visual component detection layer, a text recognition layer (e.g., OCR), an audio-to-text translation layer, etc. However, any desired number of layers may be used without deviating from the scope of the invention. Using multiple layers may allow the system to develop a global picture of what is happening in a screen or process. For example, one AI layer could perform OCR, another could detect buttons, etc.

[0066] Patterns may be determined individually by an AI layer or collectively by multiple AI layers. A probability or an output in terms of a user action could be used. For instance, to determine the particulars of a button, its text, where a user clicked, etc., the system may need to know where the button is, its text, the positioning on the screen, etc.

[0067] The sequence extraction layer may identify the same or similar sequences of actions that computing systems or users thereof (if applicable) tend to take from the collected data. For example, if after a call, users tend to open a certain software application, enter a summary from the call, and then generate an email including the summary, the sequence extraction layer may recognize this pattern and suggest a workflow sequence including logic that that automatically records the call, converts the call audio to text, opens the application, enters the text into the application, and prompts the user to review the content thereof. If confirmed, the workflow may include logic for automatically generating and sending an email with the content.



**[0068]** The clustering layer may detect that data points of a certain type or class tend to appear with a certain structures and groupings in the dataset based on similarity or dissimilarity. This may be an unsupervised machine learning technique in some embodiments. Examples may be placed definitively in a class. Alternatively, probability distributions may be used over the class.

**[0069]** The visual component detection layer may detect visual components in images and look for common patterns in detected components in data from the computing systems (or computing systems of a certain type). In some embodiments, this may be a CV model that is looking for visual components and the relationships between them.

**[0070]** The text recognition layer may recognize text in images and look for common patterns in detected text in data from the computing systems (or computing systems of a certain type). For example, if an image includes a document having a certain field, and users tend to populate that field with a certain name, word, or phrase, the text recognition layer may determine that this entry should be automated in a workflow. The automation may include workflow steps of clicking on the text field based on its screen location in coordinates and entering the text “UiPath”, for example.

**[0071]** The audio-to-text translation layer may convert audio files in the data into digital text in a form a computing system can understand (e.g., ASCII, Unicode, etc.). This layer may then determine that certain words or phrases tend to appear. When given the context of certain user activities, this may indicate that users are frustrated with a certain part of a software application. An automation may then be suggested that recognizes the frequency pattern of the phrase (s) and opens up a help window when the user speaks the phrase(s). For example, if a user is swearing, it may indicate that he or she is frustrated with the software.

**[0072]** The analysis performed by AI layers **632** can be considered to be the “detection side” of the extraction process. After analysis is performed by AI layers **632**, automations generated from this analysis in the form of robots can be pushed out to computing systems **602**, **604**, **606**, **608**. In some embodiments, robots may only be pushed out to a single computing system, or computing systems having a certain type or hardware capability. The robots may then perform the “actuation side” of the extraction process by implementing it.

**[0073]** FIG. 7 is a flowchart illustrating a process **700** for AI-based process extraction for RPA using AI layers, according to an embodiment of the present invention. The process begins with retrieving data collected by computing system listeners at **710**. In some embodiments, the data from the listeners includes user interactions with the computing systems, audio, video, light, heat, motion, acceleration, radiation, or any combination thereof. The data is run through multiple AI layers to develop a global picture of the data at **720**. The AI layers include a sequence extraction layer, a clustering detection later, a visual component detection layer, a text recognition layer, an audio-to-text translation layer, or any combination thereof. The AI layers process the data and identify one or more potential processes therein that exceed a confidence threshold at **730**. In some embodiments, each individual AI layer has an associated modifier based on its estimated accuracy. In certain embodiments, a workflow is only generated if a collective confidence threshold for all layers has been exceeded. Workflow(s) are

generated for the process(es) exceeding the confidence threshold at **740** and robot(s) are generated from the workflow(s) at **750**.

**[0074]** FIG. 8 is a flowchart illustrating a process **800** for AI-based process extraction for RPA using AI layers, according to an embodiment of the present invention. The process begins with retrieving data collected by listeners from a plurality of computing systems at **810**. In some embodiments, the data collected by the listeners includes user interactions with respective user computing systems, audio, video, light, heat, motion, acceleration, radiation, or any combination thereof.

**[0075]** The data collected by the listeners is run through the multiple AI layers at **820** to process the collected data and identify one or more RPA processes in the collected data that exceed a confidence threshold. In some embodiments, the multiple AI layers include a sequence extraction layer, a clustering detection later, a visual component detection layer, a text recognition layer, an audio-to-text translation layer, or any combination thereof. In certain embodiments, each individual AI layer has an associated modifier based on its estimated accuracy. In some embodiments, an RPA workflow is only generated when a collective confidence threshold for all AI layers has been exceeded. In certain embodiments, the multiple AI layers are configured to perform statistical modeling and utilize deep learning techniques to identify the one or more potential RPA processes in the collected that exceed a confidence threshold. In some embodiments, the collected data is run through the multiple AI layers in series, in parallel, or in a combination of both series and parallel AI layers.

**[0076]** When an RPA process is identified that exceeds the confidence threshold at **830**, an RPA workflow is generated for the identified RPA process at **840** and a robot that is configured to execute the generated RPA workflow for the identified RPA process is generated at **850**. When a similar existing RPA process to the identified potential RPA process is identified at **860**, it is determined whether the existing RPA process works less optimally than the identified potential RPA process at **870** (i.e., the identified process is more optimal). If so, the existing RPA process is replaced with the identified potential RPA process at **880** by replacing an existing robot executing the existing RPA process with the generated robot. In some embodiments, the similarity between the existing process and the identified potential RPA process is determined by entropy, minimization of a process detection objective function, or a combination thereof.

**[0077]** The process steps performed in FIGS. 7 and 8 may be performed by a computer program, encoding instructions for the processor(s) to perform at least part of the process(es) described in FIGS. 7 and 8, in accordance with embodiments of the present invention. The computer program may be embodied on a non-transitory computer-readable medium. The computer-readable medium may be, but is not limited to, a hard disk drive, a flash device, RAM, a tape, and/or any other such medium or combination of media used to store data. The computer program may include encoded instructions for controlling processor(s) of a computing system (e.g., processor(s) **510** of computing system **500** of FIG. 5) to implement all or part of the process steps described in FIGS. 7 and 8, which may also be stored on the computer-readable medium.



**[0078]** The computer program can be implemented in hardware, software, or a hybrid implementation. The computer program can be composed of modules that are in operative communication with one another, and which are designed to pass information or instructions to display. The computer program can be configured to operate on a general purpose computer, an ASIC, or any other suitable device.

**[0079]** It will be readily understood that the components of various embodiments of the present invention, as generally described and illustrated in the figures herein, may be arranged and designed in a wide variety of different configurations. Thus, the detailed description of the embodiments of the present invention, as represented in the attached figures, is not intended to limit the scope of the invention as claimed, but is merely representative of selected embodiments of the invention.

**[0080]** The features, structures, or characteristics of the invention described throughout this specification may be combined in any suitable manner in one or more embodiments. For example, reference throughout this specification to “certain embodiments,” “some embodiments,” or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases “in certain embodiments,” “in some embodiment,” “in other embodiments,” or similar language throughout this specification do not necessarily all refer to the same group of embodiments and the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

**[0081]** It should be noted that reference throughout this specification to features, advantages, or similar language does not imply that all of the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussion of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

**[0082]** Furthermore, the described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize that the invention can be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

**[0083]** One having ordinary skill in the art will readily understand that the invention as discussed above may be practiced with steps in a different order, and/or with hardware elements in configurations which are different than those which are disclosed. Therefore, although the invention has been described based upon these preferred embodiments, it would be apparent to those of skill in the art that certain modifications, variations, and alternative constructions would be apparent, while remaining within the spirit and scope of the invention. In order to determine the metes and bounds of the invention, therefore, reference should be made to the appended claims.

1. A computer-implemented method for performing artificial intelligence (AI)-based process extraction for robotic process automation (RPA) using at least one AI model, comprising:

running data through the at least one AI model, by a computing system, to process the data and identify one or more RPA processes in the data that exceed a confidence threshold; and

generating an RPA workflow for the identified RPA process that exceeds the confidence threshold, by the computing system.

2. The computer-implemented method of claim 1, further comprising:

generating an automation for an RPA robot configured to implement the generated RPA workflow for the identified RPA process, by the computing system.

3. The computer-implemented method of claim 1, wherein

the data is collected by one or more listeners, and

the data comprises user interactions with respective user computing systems, audio, video, light, heat, motion, acceleration, radiation, or any combination thereof.

4. The computer-implemented method of claim 1, wherein the at least one AI model comprises a sequence extraction layer, a clustering detection layer, a visual component detection layer, a text recognition layer, an audio-to-text translation layer, or any combination thereof.

5. The computer-implemented method of claim 1, wherein each AI model of the at least one AI model has an associated modifier based on an estimated accuracy of the respective AI model.

6. The computer-implemented method of claim 1, wherein

the at least one AI model comprises a plurality of AI models, and

the RPA workflow is only generated when a collective confidence threshold for the plurality of AI models has been exceeded.

7. The computer-implemented method of claim 1, wherein the at least one AI model is configured to perform statistical modeling and utilize deep learning techniques to identify the one or more RPA processes in the data that exceed the confidence threshold.

8. The computer-implemented method of claim 1, further comprising:

identifying a similar existing RPA process to the identified RPA process that exceeds the confidence threshold, by the computing system;

determining that the existing RPA process works less optimally than the identified RPA process, by the computing system; and

replacing the existing RPA process with the identified RPA process by replacing an automation for the existing RPA process with an automation for the identified RPA process, by the computing system.

9. The computer-implemented method of claim 8, wherein similarity between the existing RPA process and the identified RPA process is determined by entropy, minimization of a process detection objective function, or a combination thereof.

10. The computer-implemented method of claim 1, wherein



the at least one AI model comprises a plurality of AI models, and  
the data is run through the plurality of AI models in series.

**11.** The computer-implemented method of claim 1, wherein  
the at least one AI model comprises a plurality of AI models, and  
the data is run through the plurality of AI models in parallel.

**12.** The computer-implemented method of claim 1, wherein  
the at least one AI model comprises a plurality of AI models, and  
the data is fed through a combination of both series and parallel AI models of the plurality of AI models.

**13.** One or more non-transitory computer-readable media storing one or more computer programs, the one or more computer programs configured to cause at least one processor to:

- run data through at least one artificial intelligence (AI) model to process the data and identify one or more robotic process automation (RPA) processes in the data that exceed a confidence threshold; and
- generate one or more respective RPA workflows for the one or more identified RPA processes that exceed the confidence threshold.

**14.** The one or more non-transitory computer-readable media of claim 13, wherein the one or more computer programs are further configured to cause the at least one processor to:

- generate one or more respective automations that are configured to execute the one or more respective generated RPA workflows for the one or more identified RPA processes.

**15.** The one or more non-transitory computer-readable media of claim 14, wherein the one or more computer programs are further configured to cause the at least one processor to:

- identify a similar existing RPA process to an identified RPA process of the one or more identified RPA processes that exceeds the confidence threshold;
- determine that the existing RPA process works less optimally than the identified RPA process; and
- replace the existing RPA process with the identified RPA process by replacing an existing automation for the existing RPA process with an automation for the identified RPA process.

**16.** The one or more non-transitory computer-readable media of claim 15, wherein similarity between the existing RPA process and the identified RPA process is determined by entropy, minimization of a process detection objective function, or a combination thereof.

**17.** The one or more non-transitory computer-readable media of claim 13, wherein  
the at least one AI model comprises a plurality of AI models, and  
the data is run through the plurality of AI models in series or the data is run through the plurality of AI models in parallel.

**18.** The one or more non-transitory computer-readable media of claim 13, wherein  
the at least one AI model comprises a plurality of AI models, and  
the data is fed through a combination of both series and parallel AI models of the plurality of AI models.

**19.** One or more computing systems, comprising:  
memory storing computer program instructions for performing artificial intelligence (AI)-based process extraction for robotic process automation (RPA) using at least one AI model; and  
at least one processor communicably coupled to the memory and configured to execute the computer program instructions, wherein the computer program instructions are configured to cause the at least one processor to:  
run data through the at least one AI model to process the data and identify one or more RPA processes in the data that exceed a confidence threshold, and  
generate an RPA workflow for an identified RPA process that exceeds the confidence threshold.

**20.** The one or more computing systems of claim 20, wherein the computer program instructions are further configured to cause the at least one processor to:  
generate an automation that is configured to implement the generated RPA workflow for the identified RPA process;  
identify a similar existing RPA process to the identified RPA process that exceeds the confidence threshold;  
determine that the existing RPA process works less optimally than the identified RPA process; and  
replace the existing RPA process with the identified RPA process by replacing an automation for the existing RPA process with the generated automation.

\* \* \* \* \*