Getting Started With Agentic AI

Harnessing Automation With Intelligent Agents

LAHIRU FERNANDO
COUNTRY DIRECTOR, RPA LEAD (APAC), & PRACTICE LEAD - AI, BOUNDARYLESS GROUP

CONTENTS

Automation and AI are two technological fronts that have gained a lot of attention recently, driven by rapid advancements in machine learning (ML), natural language processing (NLP), and integration capabilities. These innovations have expanded the scope of what automation and AI can achieve, enabling solutions that are smarter and more adaptive. Recent developments in AI and automation have paved the way toward agentic automation, a transformative approach that extends the capabilities of traditional rule-based and process-centric technologies like robotic process automation (RPA). Integrating advanced AI techniques, agentic automation enables autonomous agents to handle complex, unstructured tasks with minimal human intervention.

This Refcard explores the core principles, key components, and practical applications of agentic automation, providing a roadmap for organizations aiming to leverage this technology for enhanced efficiency and innovation.

WHAT IS AGENTIC AUTOMATION?

Traditional automation is mostly focused on handling repetitive, rule-based tasks within structured environments. However, it often falls short when confronted with unstructured data or unpredictable scenarios. **Intelligent automation** was introduced to handle more advanced requirements by using AI and ML capabilities within automation solutions.

The ML models in these solutions are specialized for specific tasks, such as document processing, classification, and object detection. However, training these models and introducing them within automation solutions requires significant effort and time due to the need for large datasets, iterative model training, and the work to ensure seamless integration with systems.

While ML models have helped address complex scenarios, they remain limited to their specialized tasks. **Agentic AI** bridges this gap by enabling AI-driven agents that can interpret complex data, recognize patterns, execute tasks, and make informed decisions autonomously. This technology extends automation to a broader spectrum of organizational processes, including those previously identified as too variable and probabilistic to automate.

Agentic automation is the means by which enterprises can leverage agentic AI and AI agents at scale, securely and productively. It provides the technology and infrastructure to orchestrate the activities of agents, RPA robots, and people. Understanding agentic AI requires a solid understanding of agents' characteristics and their components, some of which are listed below:

Table 1: Core characteristics of agents

ASPECT	DESCRIPTION
Autonomy	Agents operate independently, assessing situations and executing tasks without constant human oversight.
Goal oriented	Agents are designed to achieve specific objectives, formulating strategies to meet defined goals efficiently.
Proactive decision making	Agents are capable of identifying user needs and initiating actions rather than merely responding to external commands.
Adaptability	Agents automatically adjust to changing conditions, learning from interactions to improve performance over time.
Self-learning	Agents use different techniques to learn from experience and automatically enhance their capabilities.

UiPath® Community

Are you looking to stay ahead in the fast-paced world of automation?

UiPath Agent Builder is a secure, simple way to build, test, and launch agents.

Why UiPath Agent Builder?

Create your own agent from scratch using a low-code development environment.

Build a combination of deterministic and non-deterministic workflows.

Bring your own data with the help of context grounding.

Ready?

Sign up for your free trial and transform the way you work.
https://bit.ly/3WBTbI9

Join a vibrant community of 3 million automation experts at
https://bit.ly/3CIPPfw

Stay updated with latest AI features releases at
https://bit.ly/4jHi5Qy

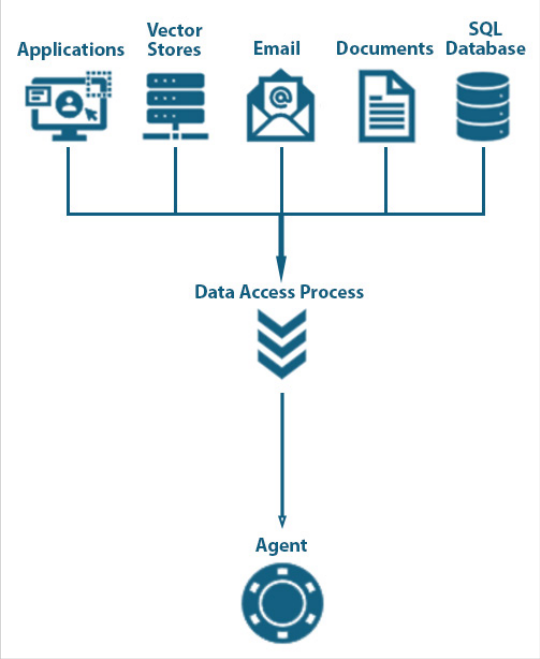
KEY COMPONENTS OF AGENTS

Implementing agents in agentic automation solutions requires several foundational components. These components form the backbone of intelligent agent functionality.

DATA SOURCES AND INTEGRATION

Agents consume structured, semi-structured, and unstructured data. The data sources that the agents consume can vary from structured SQL databases to unstructured emails and documents. Agentic solutions integrate with different data sources through various integration techniques to access the important data required for their operations. For example, a logistics agent might combine shipment data from APIs with customer feedback stored in text files to optimize delivery routes.

Figure 1: Example data sources for agents



INTENT RECOGNITION AND DECISION MAKING

Agents interpret user input and determine the best course of action using techniques such as:

- Natural language understanding to process unstructured queries
- Decision models to evaluate and prioritize tasks based on goals and constraints

For example, a customer service agent understands a question like, "What is the status of my order?" and determines whether to fetch data or escalate the query.

MEMORY AND CONTEXT MANAGEMENT

Memory enables agents to recall past interactions, ensuring consistency and personalization. Common memory storage options include in-memory storage for short-term interactions, database (SQL or graph) for structured and retrievable long-term memory, and distributed storage solutions for large-scale and collaborative agent ecosystems. Context management complements memory by allowing agents to adjust workflows based on evolving circumstances. For instance, an agent generating monthly reports remembers client-specific preferences, such as formatting styles or delivery methods, ensuring a seamless and tailored experience.

FEEDBACK LOOPS AND SELF-IMPROVEMENT

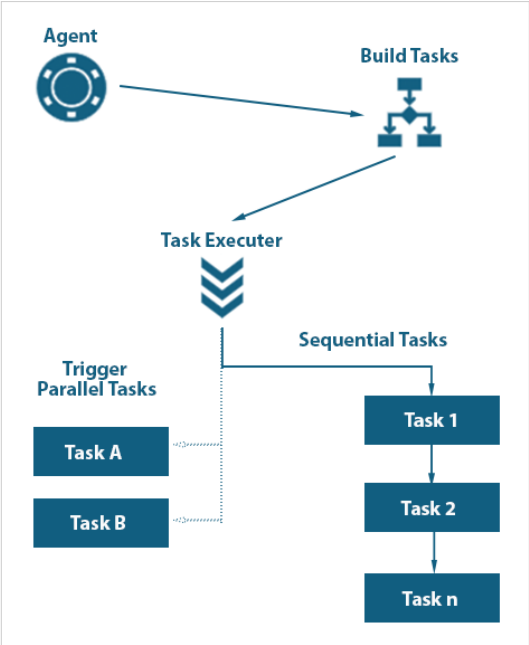
A key component of agents is self-improvement. Agents continuously evaluate their performance using feedback mechanisms, refining their decision-making processes over time. Feedback mechanisms might include user input, system logs, and performance metrics.

For example, agents could use reinforcement learning to optimize decisions based on successful task outcomes, or leverage feedback from human users to refine understanding of ambiguous intents. These mechanisms allow agents to identify inefficiencies, correct errors, and adjust to new data or scenarios. This iterative learning process for agents ensures that agentic automation remains highly adaptable to changing requirements and continuously improves its effectiveness.

DYNAMIC TASK ORCHESTRATION

Agents must coordinate complex tasks, often involving dependencies or parallel execution. This also sometimes includes coordinating tasks with multiple agents specialized for certain activities. Dynamic orchestration ensures that tasks are completed with the right resources, in the right order, using the most optimal methods.

Figure 2: Dynamic task orchestration example



DESIGNING AN INTELLIGENT AGENT

Creating an intelligent agent requires a methodical approach, blending technical expertise with a clear understanding of business needs. The following step-by-step guide helps you design and implement agents effectively.

STEP 1: DEFINE OBJECTIVES

The first step is to understand the purpose of the agent. It is important to clearly articulate what the agent is supposed to achieve. This step lays the foundation for the agent's capabilities and scope.

The following steps are helpful in identifying an agent's objectives:

1. **Identify use cases.** Start by pinpointing specific tasks or processes the agent will handle. These could range from automating invoice generation to managing customer queries.
2. **Set clear goals.** Define measurable outcomes. For example:
   - Reduce manual intervention in customer support by 30%
   - Generate billing statements with 100% accuracy within two hours
3. **Understand limitations.** Clearly define the tasks the agent will *not* perform to avoid scope creep. By establishing boundaries up front, development stays focused, timelines are maintained, and the agent delivers on its intended purpose without unnecessary complexity.

Example: A virtual assistant for HR onboarding has the primary goal of reducing administrative overhead by automating document verification, training schedules, and FAQs. However, this assistant is not designed to handle complex legal document reviews or employee performance management.

STEP 2: MAP DATA AND INTEGRATION REQUIREMENTS

Agents rely on data to make decisions and take actions. Mapping the data and integration landscape ensures that the agent has access to the right resources. Performing the following tasks can help understand the data requirements:

- **Data identification.** Determine the data sources your agent will access, such as databases, APIs, documents, or user inputs.
- **Integration strategy.** Decide how the agent will interact with these sources. This might involve REST APIs, file parsing, or middleware solutions.
- **Data security.** Ensure that all integrations comply with security standards to protect sensitive and confidential information.
- **Data validation.** Implement mechanisms to check data integrity before processing it.

Example: An agent generating financial reports retrieves data from ERP systems, validates transaction records, and integrates with tax calculation APIs to produce accurate results. Controls are introduced to enforce restricted access to sensitive data, limiting access only to authorized users.

Additionally, the agent also includes built-in data validation steps to verify the information to detect and flag inconsistencies or errors.

STEP 3: CONFIGURE INTENT RECOGNITION AND TASK MANAGEMENT

Intent recognition enables the agent to understand user requests and respond accordingly. This is where the intelligence of the agent begins to shine.

- **Training NLP models.** Use machine learning techniques, pre-trained models, or generative AI (GenAI) models to enable the agent to interpret user intents accurately.
- **Task prioritization.** Define a task hierarchy. Ensure high-priority actions have more weight during execution.
- **Fallback mechanisms.** Design rules for scenarios where the agent fails to understand an intent. For example, prompt the user for clarification or escalate to a human operator.
- **Workflow automation.** Develop workflows that the agent will execute based on identified intents. Workflows should include decision trees, conditional logic, and task dependencies.

Example: A scheduling agent receives the input, "Book a meeting with John next week." The agent interprets the intent to schedule using a pre-trained NLP model, accurately identifying the task and key details like "John" and "next week." The agent prioritizes tasks by checking for conflicts with other high-priority actions, such as pre-scheduled meetings or deadlines.

If John cannot be found in the contacts database, the agent asks the user to provide clarification. Using workflow automation, the agent queries both calendars for availability and confirms final meeting details with the user.

STEP 4: INCORPORATE CONTEXT AND MEMORY

Context and memory management are crucial for creating a seamless user experience. An agent with robust memory capabilities can personalize interactions and maintain continuity across sessions. There are different memory and context management approaches to achieve this:

- **Short-term memory.** Store session-specific information such as recent queries or ongoing tasks.
- **Long-term memory.** Retain user preferences, historical interactions, and recurring patterns to enhance personalization.
- **Context awareness.** Use memory to provide responses that consider the user's current situation and past interactions.
- **Data privacy.** Implement guardrails to ensure sensitive user data is stored and accessed securely.

It is important to identify what capabilities are required for the agent and implement them accordingly.

Example: An e-commerce chatbot remembers a customer's preferred payment method and automatically suggests it during the checkout process. The agent uses short-term memory to recall the items added

to the cart during the current session, as well as long-term memory to recognize recurring purchase patterns and suggest relevant promotions or loyalty discounts. Context awareness ensures it can seamlessly transition from answering a product query to finalizing a purchase. The agent encrypts the sensitive payment details, and only anonymized interaction data is stored for personalization.

STEP 5: ESTABLISH COMMUNICATION PROTOCOLS

Agents interact with users, other applications, and other specialized agents to perform tasks. The communication between these entities must be robust to achieve the best outcomes. It is important to consider the following when building the communication methods for agents:

- **User communication.** Use clear, concise responses that are tailored to the user's level of understanding. The agent can assess this by analyzing user input, such as vocabulary complexity or response patterns, and adapting its language accordingly. For example, a technical user might receive more detailed explanations, while a less experienced user gets simplified guidance.
- **Inter-agent communication.** Define APIs or messaging standards for agents to share data and collaborate effectively. The messaging standards define how the data is structured, along with additional metadata to support the communication and understanding of information.
- **Error handling.** Provide meaningful error messages to the user and suggest next steps when issues arise.
- **Multilingual support.** If necessary, enable the agent to communicate in multiple languages.

Example: A customer support agent uses NLP to assess the user's familiarity with technical terms. For a new user, it explains billing discrepancies in plain language, such as: "Your last payment was $50 short, which is why there's an outstanding balance."

For a more advanced user, the agent provides additional technical details and links to relevant FAQs. Inter-agent communication enables the support agent to fetch transaction data from a billing system and coordinate with a notification agent to send follow-up emails. If the system encounters incomplete data, the agent presents an error message like: "We are missing your billing ID. Please provide it to proceed."

STEP 6: TEST, DEPLOY, AND OPTIMIZE

Testing is crucial to ensure that the agent performs reliably in real-world scenarios. It is important to consider the following testing methods based on the requirement to test the agent:

- **Simulation testing.** Create test cases to mimic actual user interactions and validate the behavior of the agent.
- **Monitoring and feedback.** Use analytics tools to track the performance of the agent to identify bottlenecks, execution issues, and areas for improvement.
- **Iterative improvements.** Regularly update the agent to improve accuracy, expand functionality, and address emerging issues.

Example: A customer service assistant is tested for response accuracy, escalation workflows, and response time before being deployed to live customer interactions. Simulation testing is used to simulate certain user inputs and scenarios to test how the agent behaves. Monitoring and feedback are used to track the assistant's performance metrics, user satisfaction scores, and error rates.

PRACTICAL APPLICATION: BUILDING A BILLING STATEMENT GENERATOR

Billing is one of the most critical processes for any organization. A billing statement generator automates the creation and management of customer invoices, ensuring accuracy, timeliness, and compliance. This agentic solution eliminates repetitive manual work, reduces errors, and provides customers with professional and consistent billing experiences.

USE CASE OVERVIEW

A company processes hundreds of invoices every month. These invoices must include:

- Customer-specific details
- An itemized list of transactions, discounts, and taxes
- Payment terms, due dates, and instructions

The billing clerk currently spends hours manually gathering data from multiple systems, verifying its accuracy, creating invoices, and following up on payments.

This manual approach is:

- Time consuming as processing hundreds of invoices takes days
- Error prone as manual data entry increases the risk of inaccuracies
- Inconsistent as formatting and delivery methods vary, impacting professionalism and standards

The billing assistant is focused on addressing these challenges. The agent automatically retrieves data, generates invoices, and delivers them through the appropriate channels, ensuring efficiency and accuracy. The agent acts as a virtual accounts clerk who:

- Automatically prepares and delivers invoices to customers
- Monitors paid, unpaid, and overdue invoices
- Sends reminders for overdue payments
- Validates billing details and flags discrepancies for review

STEPS TO IMPLEMENT THE BILLING STATEMENT GENERATOR

The following steps provide a comprehensive framework to help developers conceptualize and build the virtual accounts clerk agent.

STEP 1: DEFINE OBJECTIVES

The first step is to clearly define what the agent will achieve. This establishes the scope of the project and aligns expectations with business needs.

- **Primary objectives**
  - Automate invoice generation
  - Ensure accurate calculations and formatting
  - Deliver invoices promptly and track their status
- **Key metrics**
  - Accuracy rate of invoices (e.g., 98%)
  - Time savings (e.g., 75% reduction in manual effort)
  - Payment collection success rate
- **Functional scope**
  - Support itemized transactions, discounts, and taxes
  - Offer multiple delivery channels (e.g., email, print, portal uploads)
  - Provide error handling and escalation for discrepancies

STEP 2: MAP DATA AND INTEGRATION REQUIREMENTS

The billing statement generator connects with various sources to perform its actions. It is important to identify these data sources and understand how to map them and integrate them with other applications.

- **Data sources**
  - Customer database that contains customer details
  - Transaction records that log all sales, refunds, and payments
  - Billing rules that define discounts, tax rates, and payment terms
- **Integration points**
  - ERP systems that retrieve transaction and payment data
  - APIs that access customer records and update payment statuses
  - Document generation libraries that create invoices in PDF or HTML formats
  - Communication tools that deliver invoices via email servers or upload them to customer portals
- **Data validation**
  - Ensure all retrieved data is accurate and complete before proceeding

Example: The agent connects to an ERP system, retrieves transaction data, applies discounts based on billing rules, and cross-checks tax rates from the company's compliance database.

STEP 3: CONFIGURE INTENT RECOGNITION AND TASK ORCHESTRATION

The agent requires a few configurations to understand intents that are in scope and decide the actions. The actions can include one or many tasks that need execution in a specific sequence.

1. **Intent recognition**
   - Decide the approach — handle intents using GenAI and predefined intent rules
   - Train the agent to understand commands such as "send overdue payment reminders"
2. **Task orchestration**
   - Break down the billing activities into modular automation tasks:
     - Fetch customer and transaction data
     - Apply billing rules
     - Generate invoices in a specified format
     - Validate the output for accuracy
     - Deliver invoices to customers
   - Design the model to dynamically build the execution order based on tasks and intentions
   - Train the agent to dynamically decide the execution order and sequence (parallel and sequential executions)
3. **Error handling**
   - Define fallback mechanisms — for instance, if customer data is incomplete, notify a human operator to provide the information

STEP 4: INCORPORATE CONTEXT AND MEMORY

Context and memory capabilities enable the agent to personalize and adapt its actions. This agent requires the following context and memory management methods:

- **Short-term memory** to store session-specific information, such as the billing period or customer selection, in the in-memory database to ensure fast access to data
- **Long-term memory** for handling multiple recurring requirements (available in graph or vector databases) such as:
  - Saving recurring customer preferences
  - Tracking loyalty discounts and applying them automatically
- **Context awareness** to maintain continuity in multi-step interactions, combining short-term and long-term memory, such as generating invoices for multiple customers

Note: The methods required are identified and decided during the requirements gathering and analysis stage of the project.

STEP 5: GENERATE AND DELIVER OUTPUT

The output of the agent is the final billing statement. The delivery of the statement must be done through a secure and efficient process:

1. **Invoice generation**
   - Use predefined templates for consistency
   - Include all necessary details: customer information, itemized charges, taxes, payment terms, and due dates
   - Validate the document for accuracy
2. **Delivery methods**
   - Attach the statement as a PDF and email it directly to the customer
   - Save the invoice to a secure portal for self-service access
   - Prepare a print-ready version for physical delivery and notify the operator
3. **Follow-up actions**
   - Automatically remind the user about overdue payments and update payment statuses dynamically

CONCLUSION

Agentic AI represents a transformative leap in automation, merging intelligence, adaptability, and autonomy to handle complex workflows with greater efficiency. Organizations can leverage these capabilities to transcend the limitations of traditional automation, unlocking new levels of scalability, accuracy, and innovation. The key to successful agent design lies in integrating robust data management, advanced intent recognition, and adaptive task execution. These components form the foundation for intelligent agents that deliver measurable outcomes.

The future of AI is boundless, with applications across industries like healthcare, finance, retail, and manufacturing. By starting small, iterating, and embracing scalability, organizations can seamlessly integrate agentic AI into their processes. Explore open-source communities, frameworks, and resources to deepen your understanding and take the first step toward the future of agentic automation.

Related Refcards and helpful resources:

- *AI Automation Essentials* by Tuhin Chattopadhyay, DZone Refcard
- *Machine Learning Patterns and Anti-Patterns* by Tuhin Chattopadhyay, DZone Refcard
- *Getting Started With Large Language Models* by Tuhin Chattopadhyay, DZone Refcard
- *Neural Network Essentials* by Tuhin Chattopadhyay, DZone Refcard
- "Agentic, Agentic Process Automation and Agent-Oriented Programming," Medium
- "The Dawn of Agentic Process Automation (APA)," XenonStack
- "Agentic AI: the rise of agents," Postman
- "Agentic automation with UiPath Autopilot™ for Everyone," UiPath
- "Agentic automation: The path to an orchestrated enterprise," UiPath
- "The future of the UiPath Platform is agentic and robotic," UiPath

WRITTEN BY LAHIRU FERNANDO,
COUNTRY DIRECTOR, RPA LEAD (APAC), & PRACTICE LEAD - AI, BOUNDARYLESS GROUP

Lahiru Fernando is a globally recognized expert in automation and artificial intelligence with over 13 years of experience. As a UiPath MVP and AI Ambassador, he specializes in designing intelligent automation and AI solutions that transform business processes. A frequent speaker at international events, Lahiru is passionate about advancing the future of agentic AI and empowering organizations to innovate.

DZone
3343 Perimeter Hill Dr, Suite 100
Nashville, TN 37211
888.678.0399 | 919.678.0300

At DZone, we foster a collaborative environment that empowers developers and tech professionals to share knowledge, build skills, and solve problems through content, code, and community. We thoughtfully — and with intention — challenge the status quo and value diverse perspectives so that, as one, we can inspire positive change through technology.