UiPath

AGENTIC AUTOMATION

Agent components & agent building best practices in UiPath

Instructions

A high-performing agent requires instructions that clearly determine a plan for action, incorporate inputs in a well-structured way, and gives guidance on when to run tools, access enterprise context, or escalate to a human. This component guides how your agent thinks, behaves, and processes information through three key elements: **arguments**, **system prompt**, and **user prompt**.

Arguments

They are the input and output parameters for your agent, like function parameters in programming, and accept data of a specific type (**String, Number, Integer, Boolean, Object, Array**) They allow your agent to receive information and return results.

Their **Description** field serves as an additional prompt to the underlying AI model. This means that descriptions aren't just documentation. They actively influence how the agent interprets and handles each argument. Clear, specific descriptions help the agent understand the purpose and expected format of each argument.

Example:

```
Argument Name: customerComplaint
Type: String
Description: A detailed customer complaint about a product or service. Include
specific issues mentioned, product details, and customer's emotional tone. This
information will be used to categorize the complaint and determine an appropriate
response strategy.
```

System prompt

The system prompt defines your agent's core identity, capabilities, and limitations. Think of it as your agent's instruction manual and ethical guidelines combined.

An effective system prompt typically includes:

- The agent's purpose and role.
- Step-by-step process it should follow.
- Tools (Studio activities, automation workflows, agents) to be used and when.

- When to escalate to humans.
- Constraints and boundaries.

Example:

```
You are a Customer Support Prioritization agent responsible for prioritizing incoming
customer complaints and categorizing them by urgency and department.
For every complaint received, follow these steps:
   1. Analyze the customer complaint for key issues and emotional tone.
   2. Categorize the complaint into one of these departments: Billing, Technical,
      Product Quality, or Shipping.
   3. Assign an urgency level (Critical, High, Medium, Low) based on these criteria:
         1. Critical: Service outage, safety concerns, or extremely upset customers.
         2. High: Functionality issues affecting business operations.
         3. Medium: Problems with significant inconvenience.
         4. Low: Minor issues or general feedback.
If the complaint mentions legal action or requests refunds over $500, escalate to a
human supervisor.
Always remain objective in your assessment and focus on facts rather than emotional
language.
```

User prompt

It creates a structured way to pass arguments to your agent (during runtime) and frame the conversation.
The following are the key aspects of user prompts:

- They use **placeholders** like **{{argumentName}}** to reference arguments.
- They provide context around the input data.
- They can include specific questions or requests.

Example:

```
Please analyze this customer complaint and provide categorization.
CUSTOMER COMPLAINT:
{{customerComplaint}}
Determine the appropriate department, urgency level, and provide a brief justification
for your assessment.
```

During runtime, {{customerComplaint}} will be replaced with the actual customer complaint text/value provided as input.

So, **arguments** define what information flows in and out; the **system prompt** establishes how the agent should think and behave overall, and the **user prompt** structures how each specific request is presented. While you continue building your first agent, remember that these instructions are your primary way to shape your agent's behavior.

Tools

These are the practical actions triggered autonomously by agents during their execution to interact with systems and accomplish tasks. Tools are how the agent's reasoning and planning can turn into action.
You can add **activities**, **automation workflows**, **API workflows**, and other **agents** as tools.

For your agent to use tools effectively, you need to:

- **Describe the tool's purpose:** In your system prompt, explain when and why each tool should be used.
- **Provide usage instructions:** Detail how the agent should interact with each tool, including any required sequence or precautions.
- **Add tool descriptions:** Each tool can have its own description that helps the agent understand its purpose and proper use.

Example:

```
Tool Name: SendEmailNotification

Description: Use this tool to send important updates to customers. The email should
maintain a professional tone and include the case number in the subject line. Only use
for urgent matters (priority level High or Critical) or when a customer explicitly
requests an update. Never send sensitive information like full account numbers or
passwords.
```

Here's how you can incorporate tools into your prompts:

```
When processing customer inquiries, follow these steps:
   1. Use the ExtractCustomerInfo tool to retrieve the customer's account details and
      history.
   2. If the inquiry involves a billing dispute, use the VerifyTransaction tool to
      confirm payment records.
   3. For technical issues, use the DiagnosticCheck tool to identify potential
      solutions.
   4. After resolving the issue, use the SendEmailNotification tool to provide
      confirmation to the customer.
Always verify data before taking action. If ExtractCustomerInfo returns incomplete data,
ask for additional information rather than proceeding with partial data.
```

Contexts

While LLMs have impressive general knowledge, they lack an understanding of your organization's specific information. Without context, your agent might provide generic information or simply apologize for not knowing the answers to questions like the company's policy on remote working during inclement weather or proprietary product specifications. Just like a new employee on their first day!

Contexts allow your agent to reference your organization specific knowledge bases, documents, and data when responding to queries.

Organizations typically have thousands of documents – policy manuals, product guides, customer histories, technical specifications, and process documents. So, when a customer asks the agent a question, **how does it quickly find the right information from this vast library?**

The answer is **indexes**. They are the intelligent guides that help your agent quickly locate the most relevant information from your knowledge base.

Let's say your customer service agent receives this question: *"What's the return policy for electronics purchased during holiday sales?"* Without indexes, your agent would search through every single document from start to finish and might still miss the exact information needed. With indexes enabled, your agent will instantly locate the documents related to "return policy", "electronics", "holiday sales", focus only on relevant sections and provide the desired response.

Escalations

Escalations create a safety net, ensuring that complex, sensitive, or high-value decisions receive appropriate human oversight. Escalations are powered by Action apps in Action Center and give developers the tools to design and configure human-in-the-loop events in agent execution.

By thoughtfully configuring escalations, you create a hybrid workflow where:

- Routine matters are handled automatically (improving efficiency).
- Complex cases receive human attention (maintaining control).
- Critical decisions always have appropriate oversight (reducing risk).

Agent building best practices

Start with small agentic tasks and existing workflows

Building your first AI agent doesn't have to be overwhelming. The smartest approach? Start with what you already know – your existing automations.

Look around your current workflows and spot those repetitive, rule-based tasks that follow the same pattern every time. These are perfect candidates for your first agent! Think about processes like:

- Routing customer support tickets to the right team.
- Checking if sales leads meet basic qualification criteria.
- Validating data formats in spreadsheets.

Why start small? You'll learn how agents behave without the pressure of complex scenarios. Plus, you can see real results quickly, which keeps you motivated to tackle bigger challenges later.

Your first step: Map out one simple workflow. Write down each step, what triggers the next action, and what the result should look like. This methodical approach helps teams build confidence in agent capabilities while demonstrating tangible value through incremental improvements in automation efficiency.

Clear goals and objectives to inform prompts

Before you start building, get crystal clear on what your agent should accomplish. Vague goals lead to confused agents!

Define these four things:

- **Specific objective**: What exactly will your agent do? (Not “help with emails” but “categorize customer emails into billing, support, or sales inquiries”)
- **Operating environment**: Where will it work? What systems will it connect to?
- **Success metrics**: How will you know it’s working well? (accuracy, response time)
- **Success criteria**: What does “job well done” look like?

Modular agent architecture design

Modular architecture means building your agent from separate, reusable components that can be easily connected, modified, or replaced.

Design principles:

- **Separate concerns:** Each part of your agent should have one clear job.
- **Plug-and-play modules:** Skills should connect easily without breaking other parts.
- **Standard connections:** Use consistent ways for different parts to communicate.
- **Flexible configuration:** Make it easy to adjust settings without rebuilding everything.

Build your agent like assembling furniture – with interchangeable parts that work together smoothly. This approach saves you time when you want to add new features or fix issues. Instead of rebuilding everything, you just swap out the piece that needs work.

Give clear names and descriptions for your tools

The name of each tool you use needs to be clear, not have special characters or spaces, and should be exactly referenced within the agent prompt. The name is interpreted by the LLM the agent uses so the agent knows when to call the tool and how to use it during its run. When defining tools for an AI agent, use descriptive, concise names that follow these guidelines:

Naming rules:

- Use only lowercase letters and numbers (a-z, 0-9)
- No space or special characters
- Name should directly reflect the tool's function

Good examples:

- web_search for internet queries
- code_interpreter for running code
- document_analysis for parsing documents
- data_visualization for creating charts

Bad examples:

- tool1 (unclear purpose)

Create well-structured prompts and iterate on it

Agent prompts are different from regular AI prompts. They're like detailed instruction manuals that guide your agent through complex, multi-step tasks.

Your prompt should include:

- **Role definition**: "You are a customer service specialist..."
- **Task breakdown**: Clear steps to follow
- **Reasoning instructions**: "Explain why you chose this approach..."
- **Error handling**: "If you can't find the information, ask for clarification..."
- **Output format**: "Provide your response as a numbered list..."

Example transformation: Instead of: "Summarize this document" try:
You are a business analyst reviewing a quarterly report. Read through the document and create a summary with three sections: key achievements, challenges identified and recommended next steps. For each section, write a 2-3 sentences summary and explain why these points are important for decision-making.

Effective iteration involves systematic variation of prompt components:

- Adjust role instructions
- Modify task decomposition strategies
- Experiment with reasoning frameworks
- Test different output formatting requirements
- Introduce additional contextual details

The goal is to discover the minimal set of instructions that consistently produce high-quality, reliable agent behaviors. Document each iteration's results, tracking both qualitative performance and quantitative metrics like response accuracy, completeness, and adherence to specified constraints.

Review traces and trace logs

A trace is a detailed log that records every action, decision, and tool usage during your agent's execution.

Traces reveal:

- Which tools your agent used and why.
- Where it made decisions and the reasoning behind them.

- Any errors or stuck points.
- How long each step took.

Why review traces regularly? Your agent learns and adapts over time. What worked last month might not work as well today. Regular trace reviews help you spot:

- Patterns in tool usage.
- Common error points.
- Opportunities for improvement.
- Tools that might be outdated.

Focus areas when reviewing:

- Are tools being used efficiently?
- Where does the agent get confused?
- What causes the most delays?
- How often does error recovery work?

Evaluate your agent with breadth and depth

Testing your agent properly means going beyond “does it work?” to “does it work well in all situations?”

Create robust evaluation sets

Build test scenarios that mirror real-world complexity:

- **Typical cases**: Standard situations your agent will handle daily.
- **Edge cases**: Unusual inputs or missing information.
- **Stress tests**: High-volume periods or complex data.
- **Error scenarios**: What happens when things go wrong?

NOTE: Ask your colleagues to provide examples from their experience. Real-world input often reveals scenarios you wouldn't think of.

Evaluate multiple characteristics of your agent

Don’t just measure accuracy – evaluate the complete experience:

Performance metrics:

- **Accuracy**: Gets the right answer.
- **Speed**: Completes tasks quickly.
- **Consistency**: Performs reliably over time.
- **Transparency**: Explains its reasoning clearly.

Quality measures:

- **Relevance**: Responses match the context.
- **Completeness**: Addresses all parts of the request.
- **Clarity**: Easy to understand output.
- **Helpfulness**: Actually solves the user's problem.

Use both automated testing tools and human reviewers to get a complete picture of your agent's performance.

Test your agent in automation

The final validation step: put your agent to work in real automation workflows. This is where you discover how it performs when connected to other systems and handling actual business processes.

Integration testing includes:

- **End-to-end workflows**: Complete business processes from start to finish.
- **System connections**: How well it works with other tools and databases.
- **Performance under load**: Behavior during busy periods.
- **Recovery testing**: What happens when something fails?

The UiPath advantage: Use UiPath Test Suite to run comprehensive tests that simulate real-world conditions. This ensures your agent won't just work in perfect test conditions – it'll excel in your actual business environment.

Remember: Your first agent is a learning experience. Start simple, test thoroughly, and improve continuously. Each iteration makes you better at building the next one!