10 Ways to Guide AI Agent Behavior

Using `RequirementAgent` for predictable multi-turn workflows

Why we built the RequirementAgent

When we started building agentic systems, we quickly hit a problem:

1. Fixed workflows were **too rigid**.
2. Unconstrained agents were **too unpredictable**.

We needed a way to **guide behavior at every step** of an interaction, without writing control logic or constantly tweaking prompts.

The solution: #RequirementAgent

It lets you define **constraints** — simple rules agents must follow.

Examples:

- Only run Tool A after Tool B
- Use a tool exactly once, no more
- Ask a human before sending an email
- Stop immediately if sensitive content appears

Why it matters

This makes agent behavior predictable and reliable, without:

- Hardcoding logic
- Relying on fragile prompts

It’s ideal for:

- Multi-agent handoff and delegation
- Minimizing hallucinations in tool-driven workflows
- Human-in-the-loop processes
- ReAct-style planning and reasoning

Available now in #BeeAI

We've been using this pattern internally for weeks — it's been invaluable.

Now, it's available to everyone in the **BeeAI framework**.

👉 This deck contains **10 real examples** — from simple ordering to full agent coordination.

1. Require context before tool use

💡 Get the user's location before checking the weather

```
agent = RequirementAgent(
    tools=[fetch_user_location, weather_tool],
    requirements=[
        ConditionalRequirement(
            weather_tool,
            only_after=[fetch_user_location],
        )
    ]
)
```

2. Require tool use exactly once

💡 Must use `price_estimator` exactly once at any time

```
agent = RequirementAgent(
    tools=[price_estimator],
    requirements=[
        ConditionalRequirement(
            price_estimator,
            min_invocations=1,  # Must use at least once
            max_invocations=1,  # Cannot use more than once
        )
    ]
)
```

3. Start with analysis, only once

💡 Analyze the task before taking action

```
agent = RequirementAgent(
    tools=[analyze_task],
    requirements=[
        ConditionalRequirement(
            analyze_task,
            force_at_step=1,  # Must be the first tool used
            max_invocations=1,  # Can only be used once
        )
    ]
)
```

4. Retry search with rephrasing

💡 Retry up to 3 times if search is empty

```
class RetryWithRephrasing(Requirement):
    # ...
    async def run(...):
        if last_step.is_empty():
            return [Rule(target="rephrase_tool", forced=True)]

agent = RequirementAgent(
    tools=[rephrase_tool, wikipedia_search_tool],
    requirements=[
        RetryWithRephrasing(wikipedia_search_tool),
        ConditionalRequirement(wikipedia_search_tool, max_invocations=3),  # Max 3 times
    ]
)
```

5. Multi-agent handoff with constraints

💡 Use Destination Expert before Weather Expert

```
agent = RequirementAgent(
    tools=[
        HandoffTool(destination_expert),
        HandoffTool(weather_expert)
    ],
    requirements=[
        ConditionalRequirement(
            "weather_expert",
            only_after="destination_expert"
        ),
    ]
)
```

6. ReAct loop control

💡 Alternate tools and reasoning; avoid consecutive thinking

```
agent = RequirementAgent(
    tools=[ThinkTool(), wikipedia_tool, weather_tool],
    requirements=[
        ConditionalRequirement(
            ThinkTool,
            force_at_step=1,  # Must start with thinking
            force_after=[Tool],  # Force thinking after any tool
            consecutive_allowed=False,  # Prevent consecutive thinking
        )
    ]
)
```

7. Require one tool before another

💡 Search flights before booking

```
agent = RequirementAgent(
    tools=[search_flights, book_flight],
    requirements=[
        ConditionalRequirement(
            book_flight,
            only_after=[search_flights],
        )
    ]
)
```

8. Enforce final action before answer

💡 Send summary before final answer

```
agent = RequirementAgent(
    tools=[get_sales_data, send_email_summary],
    requirements=[
        ConditionalRequirement(
            send_email_summary,
            min_invocations=1,
            max_invocations=1,
        ),
        ConditionalRequirement("final_answer", force_after=send_email_summary),
    ]
)
```

9. Ask user before sensitive tools

💡 Always ask before sending an email to the manager

```
agent = RequirementAgent(
    tools=[send_email_to_manager, draft_report],
    requirements=[
        AskPermissionRequirement(
            send_email_to_manager,
            remember_choices=False,
        )
    ]
)
```

10. Stop if sensitive data detected

💡 Halt if credit card number appears in output

```
class PrematureStopRequirement(Requirement):
    # ...
    async def run(...):
        text = last_output.get_text_content()
        if re.search(self.pattern, text):
            return [Rule(target="final_answer", forced=True)]

agent = RequirementAgent(
    tools=[log_reviewer],
    requirements=[
        PrematureStopRequirement(
            pattern=r"\b\d{4}[\s-]?\d{4}[\s-]?\d{4}[\s-]?\d{4}\b",  # Credit card pattern
            reason="output contains credit card numbers"
        )
    ]
)
```

Thanks!

Which one would you use first?

Repository: github.com/matoushavlena/beeai-requirement-agent-examples

Let's connect on LinkedIn — **@matoushavlena**