



Co-funded by  
the European Union

Project Nr. 2022-1-FR01-KA220-HED-000086863



lightcode

Strengthening the Digital  
Transformation of Higher Education  
Through Low-Code

# 11. Automating Processes

KarmicSoft



Dauphine | PSL  
UNIVERSITÉ PARIS



KarmicSoft symplexis

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.



Erasmus+ Project  
**lightcode**



Co-funded by  
the European Union

## PROJECT'S COORDINATOR

**Dauphine** | PSL   
UNIVERSITÉ PARIS

**Paris Dauphine  
University**  
France

## PROJECT'S PARTNERS



**University of  
Macedonia** Greece



**University of  
Niš**  
Serbia



**Karmic Software  
Research**  
France



**REACH  
Innovation**  
Austria



University of Zagreb  
Faculty of  
Economics & Business  
**University of  
Zagreb**  
Croatia

**symplexis**  
**Symplexis**  
Greece

*symplexis.eu*

## Student Course Training Package

**TABLE OF CONTENTS**

OVERVIEW	4
STRUCTURE OF THE MODULE	7
AUTOMATING PROCESSES, STEP BY STEP	9
CONCLUSION	27
REFERENCES	28

## OVERVIEW

### **Introduction:** *Automating Processes – Turning Actions into Smart Workflows*

In every organization and daily life situation, we unconsciously follow processes. Whether it's approving a leave request, managing a customer order, or even making a cup of coffee, there's a sequence of actions that lead to a result. But what if these repetitive tasks could happen automatically, saving us time and effort? That's where **process automation** comes in.

With the LightCode platform, you don't need to be a developer or process engineer to automate workflows. By using simple objects, states, and methods, you can model real-life scenarios and bring automation to life—without writing a single line of code.

This module invites you to explore how everyday events can be turned into automated workflows. Through a playful and relatable example—the story of Emma, a thirsty judoka—you will discover how simple objects like **Beverage**, **Order**, and **Payment** interact using state machines to automate a complete process, from placing an order to sipping a delicious drink.

By the end of this module, you'll understand that process automation isn't about complex algorithms—it's about connecting events and state changes in a logical, intuitive flow.

### **Why Is This Module Important?**

- Businesses around the world save **millions of hours** by automating repetitive processes.
- Professionals who understand how to **model and optimize workflows** have a distinct advantage in any industry.

- With low-code platforms like LightCode, automation becomes accessible to everyone—not just IT professionals.

Whether you're a business student looking to optimize team workflows, a healthcare manager aiming to streamline patient check-ins, or simply someone curious about how things work behind the scenes, this module will empower you to design and implement smart, automated processes using simple and visual tools.

Take your first step into the world of process automation and experience how LightCode turns ideas into working solutions—efficiently, elegantly, and with minimal effort.

## Learning Outcomes

By the end of this module, you will be able to:

- ✓ **Understand** what process automation is and why it's valuable in both business and everyday life.
- ✓ **Imagine workflows** by thinking in terms of events, actions, and object interactions.
- ✓ **Model simple automated processes** using state machines directly in the LightCode platform.
- ✓ **Implement triggers and state transitions** without writing complex code.
- ✓ **Recognize the value of formal specifications** (like BDD/GWT) to describe processes clearly and align business and technical teams.
- ✓ **Reflect critically on automation opportunities** in real-world contexts.

## Prerequisites

This module is designed for **absolute beginners in process modeling** and those curious about automation, regardless of technical background. However, to make the most of this learning journey, it's recommended that you have:

- Completed **Module 8: Let's Explore the LightCode Platform**.
- Basic familiarity with **objects, classes, and states** introduced in earlier modules.
- A general understanding of how simple actions lead to outcomes (e.g., "If I click this button, something happens").
- Most importantly, an **open and curious mindset**—ready to explore and think in terms of *events* and *sequences*, even if you're not using visual diagrams.

---

💡 *Tip:* If you've ever organized a small event, followed a recipe, or created a checklist, you've already experienced a form of process thinking. In this module, you'll simply learn to formalize that intuition using objects and state changes!

## STRUCTURE OF THE MODULE

- The journey starts with a simple need. Someone, somewhere, faces a small but persistent challenge. In our story, it's Emma and her unquenchable thirst after a long training session. But as you'll soon discover, this is about much more than a drink—it's about understanding why automation matters and how it quietly improves our daily lives.
- Every great experience begins with a clear purpose. You'll explore how identifying that purpose helps design better processes and why asking the right questions is often more important than finding immediate answers.
- Meet Emma, the focused and disciplined creator who doesn't have time for wasted clicks or long waits. Her story will guide you through the essential mindset of designing efficient, user-centered workflows.
- You'll follow her path step by step, not by diving into technical details just yet, but by imagining the journey from her perspective. What happens first? What's the ultimate goal? And how does everything in between connect to deliver that final moment of satisfaction?
- Of course, not everything is as smooth as it seems. Hidden behind every successful experience are challenges waiting to be discovered. How secure is the payment process? Can resources keep up with demand? Is the experience fast enough to meet rising expectations? You'll learn how to spot these hidden requirements and why they matter.
- Behind the scenes, a hidden team of objects works tirelessly to make it all happen. Each plays a specific role, holds key information, and quietly ensures that things stay on track. You'll meet them one by one and understand how they keep the process flowing, even when no one is watching.

- And then comes the real magic. A quiet force working in the background to make everything seamless—the Assistant. Without asking for attention, it handles tasks on Emma’s behalf, creating orders, processing payments, and delivering results before she even finishes cooling down from her workout.
- But can we trust this kind of magic? You’ll test the process, explore what happens behind the curtain, and find out whether automation really makes things faster—or if it just feels that way.
- And just when you think the journey is over, you’ll glimpse something more. A larger world where systems talk to each other, where actions happen across platforms, and where automation connects not just objects, but entire digital landscapes.
- With this new understanding, you’ll face the final question: Are you ready to handle the unexpected? To step beyond the happy path and design systems that work no matter what surprises appear?

## AUTOMATING PROCESSES, STEP BY STEP

### 1. The Learning Journey

Welcome to the **SwiftSip™** module—where you'll not only explore how process automation works but also experience how professional product owners think and design impactful solutions.

In this learning journey, we follow a clear, structured approach inspired by modern Agile methodologies.

*Ready? Let's open the module SwiftSip and sip the passionate story of Emma.*



## SwiftSip™



### A Thirsty Journey

Emma, a disciplined judoka and passionate light-coder, just wrapped up a challenging training session. Her muscles ache, and her mind races through the techniques she practiced. One thing is clear—she needs a refreshing drink. Fast.

She pulls out her phone and opens the **SwiftSip™** app. Within seconds, she finds her favorite Matcha tea on the menu. A simple tap, and her order is placed. Another tap, and payment is complete. No confusing menus, no endless confirmations. Just smooth, disciplined efficiency—just like her judo practice. Before she even finishes catching her breath, her tea is ready. She takes a long, satisfying sip.

**SwiftSip™** did exactly what it promised—delivering refreshment without distractions.



### Persona

**Background:** Emma is a focused creator who loves simplicity—in her projects and her life. She builds minimalist low-code apps and practices Judo 🥋 to stay sharp.

#### Needs:

- A fast, structured way to order drinks after training.
- Simple, clean interfaces without distractions.
- Instant feedback when actions are completed.

#### Pain Points:

- Dislikes cluttered menus and slow apps.
- Hates waiting and repetitive confirmations.
- Prefers smooth, focused experiences.



### Meet Emma



💡 *Design Tip:* For Emma, less is more. Keep it simple, fast, and efficient.

### **Step 1 – Start with the Why: The Pitch**

- You immediately discover the *reason* this app exists through an engaging story:

Emma, a disciplined judoka and light-coder, needs a fast and structured way to quench her thirst after training.

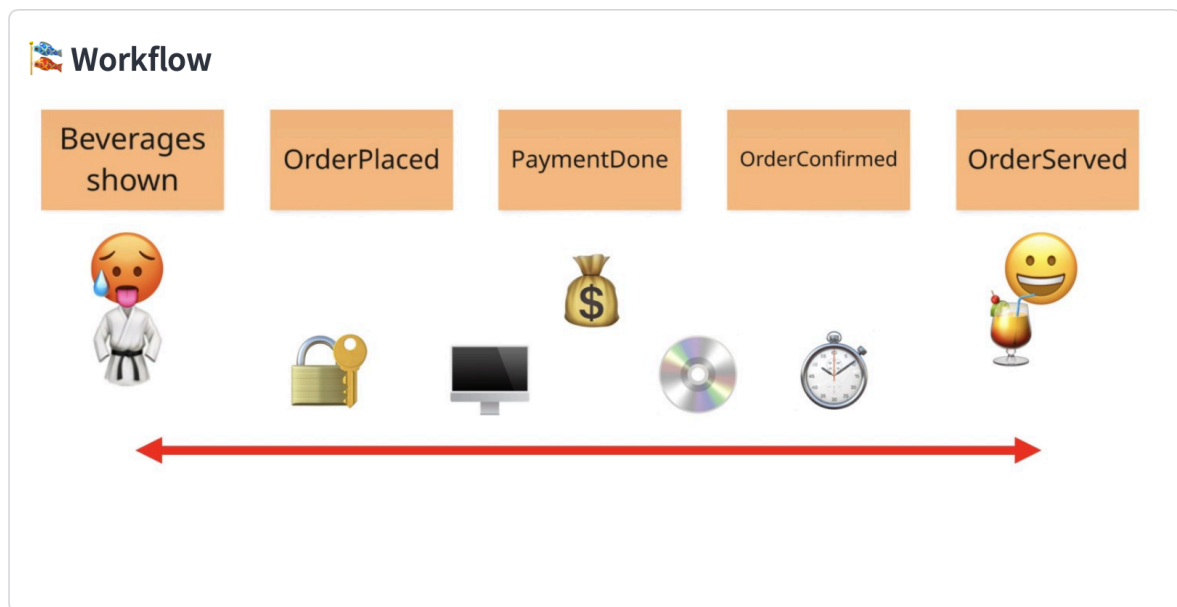
- This captures your attention and grounds the learning in a **real-world use case**.

### **Step 2 – Introduce the User: Meet the Persona**

- You meet Emma through a concise persona description, helping you understand her goals, frustrations, and expectations.
- A visual reinforces the human connection—this isn't just about technology; it's about solving **real human problems**.

### **Step 3 – Visualize the Flow: Event Storming and Process Thinking**

- Finally, you're invited to explore the process flow through the **Workflow** section (workflow.jpg), using a light formalism inspired by Event Modeling.
- This helps you **think in domain events and** prepare room for state transitions, exactly as a product owner or process designer would.



### How to Read This Workflow – Focus on Purpose, Then Process

When analyzing a workflow, follow this clear approach:

- **Start with the Triggering Event – What Sets Everything in Motion?** The journey begins when Emma **opens the SwiftSip™ app and sees the available beverages**. This is the first key event: *Beverages Shown*. The system presents clear, simple options to help Emma quickly make her choice.
- **Jump to the Final Event – What Is the Goal?** The process concludes successfully when **Emma receives her drink and enjoys a satisfying sip of her Matcha tea**. This final event—*OrderServed*—marks the achievement of Emma’s goal. Everything that happens in between should lead efficiently to this moment.
- **Now, Explore the Intermediate Events – How Does She Get from Start to Satisfaction?**

- 📝 **OrderPlaced:** Emma selects her drink and places the order with a single, simple action.
  - 💳 **PaymentDone:** She completes the payment seamlessly. This isn't just for her convenience—it's also where SwiftSip™ secures its revenue, ensuring the service remains sustainable.
  - 🍷 **OrderConfirmed:** The barista confirms the order and starts preparation. Emma knows her drink is on its way.
- 

💡 *This reading approach helps you stay focused on what really matters: the user's experience. Start with their first action, immediately visualize the desired outcome, and then analyze the journey that connects the two.*


## 📖 Considering Non-Functional Requirements (NFRs)

While the process flow looks simple and efficient, delivering a satisfying experience like this requires more than just getting the steps right. We also need to ensure the system meets certain critical **quality attributes**—these are known as Non-Functional Requirements.


Let's look at them through Emma's eyes:

- 🛡️ **Security:** Emma trusts SwiftSip™ with her payment details and personal preferences. The system must guarantee that her data is safe and payments are secure.


*If Emma ever felt her data was at risk, she'd stop using the app immediately.*

-  **Resource Allocation:** When Emma places her order, is there a barista available? Are there enough drinks in stock?

*The system should manage resources intelligently to avoid making promises it can't keep.*

-  **Lead Time (Speed of Service):** Emma is in a hurry after her training session. Long waiting times will lead to frustration.




*SwiftSip™ must minimize delays and ensure that Emma receives her drink quickly, living up to its promise of efficiency.*

 *NFRs may not be directly visible to Emma, but they are critical to her satisfaction. A great product doesn't just work—it works securely, reliably, and fast. By considering this, you become a seasoned product owner, key skill in modern organizations.*

## 2. The Supporting Objects and States

Emma's smooth experience with SwiftSip™ depends on a few essential objects working together behind the scenes. These objects represent the available products and support the decision-making process.

### Drinks

	Coffee
	Green Tea
	Lipton Tea

### Green Tea

**Name**  
Green Tea

**Price**  
2.50




**Available**  
☐


**Stock**  
3

**Doc**  
Green tea is a type of tea that is made from the Camellia sinensis plant. It has a light, refreshing flavor and is known for its health benefits.

## Available Beverages

When Emma opens the SwiftSip™ app, she immediately sees a clear and simple menu of drinks. Here's what's available:

Icon	Beverage	Price (€)	Description
	Coffee	3.50	A classic, energizing coffee.
	Green Tea	2.50	Light and refreshing; known for health benefits. <i>(Emma's Choice)</i>
	Lipton Tea	4.00	A classic tea for traditional tastes.

Emma chooses **Green Tea** —the perfect balance of refreshment and calm after her training session.

 *Tip for Product Owners:*

Notice how each beverage option includes a clear price and a short description. This reduces decision fatigue and helps the user make quick, confident choices.


Also, be sure to manage stock and availability dynamically to avoid offering products that aren't ready to serve!

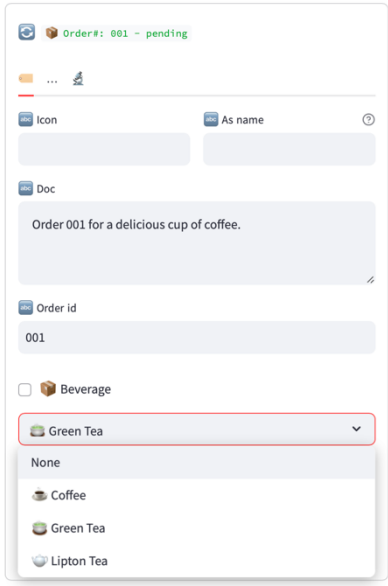
Here's the accurate explanation for the **Order** object, using only the defined states and reflecting the available methods:

## The Order: Regulating the Workflow

In SwiftSip™, the **Order** object plays a critical role in **regulating the flow of the process**. It keeps track of Emma's beverage choice and ensures that every action happens at the right time, in the correct order.

### Beverage Selection – The Picklist

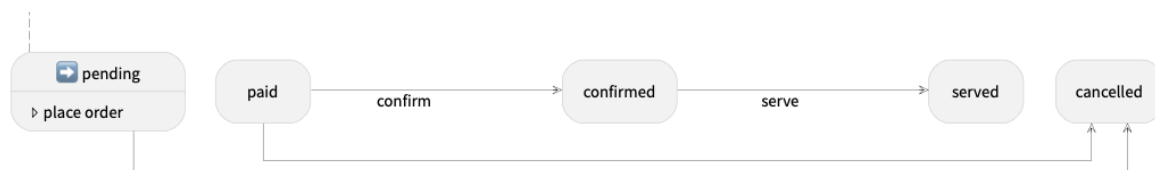
- When Emma creates her order, she chooses a drink from a **picklist of available beverages**.
- In this case, she selects **Green Tea** .
- This selection is stored directly in the order: beverage: Green Tea



The screenshot shows a mobile application interface for creating an order. At the top, it says 'Order#: 001 - pending'. Below this, there are fields for 'Icon', 'As name', and 'Doc'. The 'Doc' field contains the text 'Order 001 for a delicious cup of coffee.' Below the 'Doc' field is the 'Order id' field, which contains '001'. At the bottom, there is a 'Beverage' section with a dropdown menu. The dropdown menu is open, showing a list of options: 'None', 'Coffee', 'Green Tea', and 'Lipton Tea'. 'Green Tea' is currently selected and highlighted with a red border.

### Order State – Managing Process Progress

The order moves through the following well-defined states:



## State      Description

pending      Order created, awaiting action.

paid          Payment completed successfully.

confirmed      Barista confirmed the order.

served      The drink has been delivered.

cancelled      The order was cancelled before being served.

- Emma's order starts in the **pending** state.
- Once payment is made, it moves to **paid**.
- The barista then **confirms** the order and begins preparation.
- Finally, when Emma receives her drink, the order reaches the **served** state.

## Available Actions (Methods)

Method	What It Does	Allowed From State
.place_order()	Places the order and books the drink.	pending (Transitions to paid)
.confirm()	Confirms the order.	paid (Transitions to confirmed)
.serve()	Marks the order as served.	confirmed (Transitions to served)
.cancel()	Cancels the order before it's pending or paid (Transitions to served).	cancelled)

Each action ensures that the process follows a **controlled and predictable path**.

pending paid confirmed served cancelled

▶ Place order Confirm Serve ▶ Cancel

💡 *Key Takeaway:*

The Order object acts like a **process controller**.

It makes sure payments happen before preparation starts, and no drink is served before it's properly confirmed. This ensures smooth operations, prevents mistakes, and keeps both Emma and the business happy!

And here she is. Emma places the order, initiating the transition. Watch the underlying conversation with her app, including some metrics for NFRs.

Stock

2

Payment

Icon As name

Doc

Payment id

123

☐ Order

Order#: 001 - paid

Amount

0.00

pending **successful** failed refunded

Initiate Fail Refund

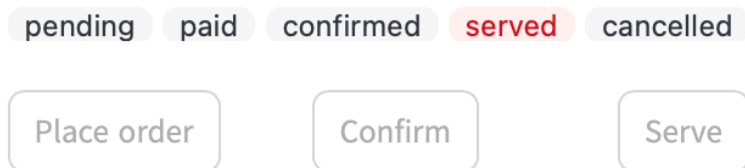
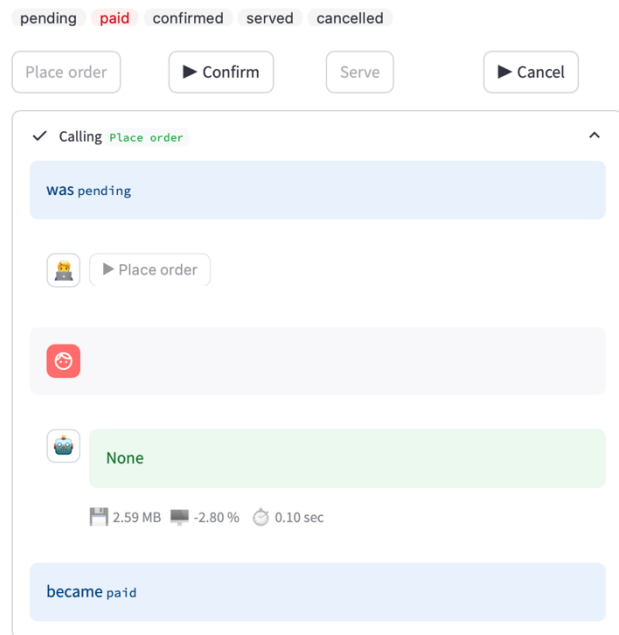
✓ Calling **Initiate**

## Student Course Training Package

As a side effect, could you please check the Stock? What do you observe?

And how about the payment? Did you tried to initiate? What's the effect on the order?

And, step by step, here we are. The Beverage is served, reaching out both customer's satisfaction and our journey so far.



Or did we? Any inconsistency so far? Please review the example, simulate execution, and act to fix problems, if any. You now know how to specify, design, test, and execute workflows using object states. You're becoming a great process designer, ready to automate it.

### 3. Introducing the Assistant – Automating the Happy Path

So far, Emma has manually selected her drink, placed the order, initiated the payment, and waited for her beverage to be served. But why should she spend time clicking through all these steps every time?

For repeatable, well-defined workflows, we can introduce automation directly into the process. This is where the **Assistant** 🤖 comes in.

#### 🎯 The Role of the Assistant

The Assistant is a smart helper that takes care of the entire ordering process for Emma. It already knows:

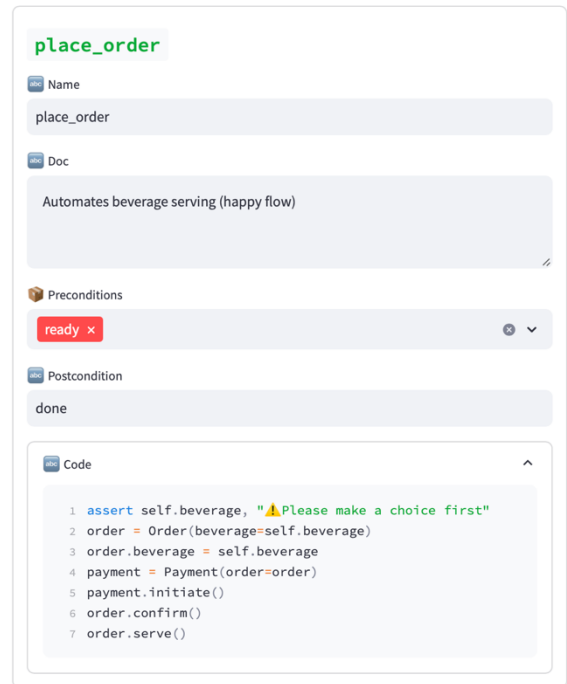
- Which beverage Emma wants (in this case, her favorite **Green Tea** 🍵).
- How to automatically create the necessary objects: **Order** and **Payment**.
- How to execute the entire flow using the happy path, without any manual intervention.

#### 📖 What Happens When the Assistant Acts?

- The Assistant starts in the **ready** state.
- Emma triggers the Assistant by simply selecting her drink.
- The Assistant performs the following actions automatically:

## Student Course Training Package

1. Creates an **Order** for the selected beverage.
2. Creates a corresponding **Payment**.
3. Initiates the payment directly using `.initiate()`.
4. Confirms the order with `.confirm()`.
5. Finally, serves the beverage with `.serve()`.



```

place_order

Name
place_order

Doc
Automates beverage serving (happy flow)

Preconditions
ready ×

Postcondition
done

Code
1 assert self.beverage, "⚠️Please make a choice first"
2 order = Order(beverage=self.beverage)
3 order.beverage = self.beverage
4 payment = Payment(order=order)
5 payment.initiate()
6 order.confirm()
7 order.serve()

```

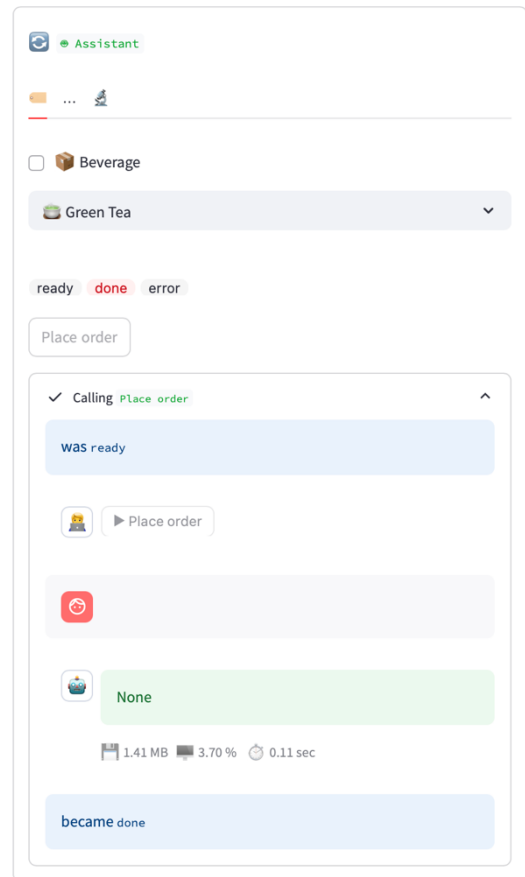
- The Assistant finishes in the **done** state, and Emma enjoys her drink—without further clicks.

💡 *In your LightCode app, this is exactly what happens when you call:*

```
assistant.place_order()
```

Everything happens in sequence, and Emma's drink is served before she even finishes cooling down after her judo session.

And to trigger the whole automation? Well, you only need to add an assistant to your module, and an inspector to command it. Therefore, using an Inspector, this is done through one click.



## 🔧 Testing the Automation

Before deploying automation, it's crucial to validate that everything works as expected. In this case, the SwiftSip™ system includes a dedicated test method to ensure the **Assistant** behaves correctly when automating the beverage ordering process.

0 error(s) found.

```

Loading swift_sip
System Under Test: 🍹 Swift sip

## @ Assistant
i Assists with beverage ordering
### Assistant._test_place_order
i
def _test_place_order():
    fixture = type(self)()
    fixture.beverage = Beverage(name="Test Drink", price=2.5)
    fixture.place_order()
    assert fixture.state == 'done'

`result = None`
#
# ✅ Done with no errors.
  
```

The test, called **\_test\_place\_order()**, creates a simulated Assistant instance, assigns a test beverage, and triggers the full ordering workflow by calling **.place\_order()**. The final assertion checks that the Assistant reaches the **done** state successfully.

The result?  **Done with no errors.**

This confirms that the Assistant can autonomously complete the happy path, from selecting a beverage to serving it, without any issues. With this validation in place, you can confidently rely on automation to streamline the process and deliver a seamless experience to users like Emma.



### Why This Matters

- Emma experiences faster service and fewer distractions.
- The business ensures a smooth, reliable process that's easy to monitor and control.
- You, as a future process designer, learn how to delegate responsibility to automation tools while keeping control through well-defined states: ready → done.



### Reflection Questions

- What would happen if the selected beverage was unavailable?
- How could the Assistant handle errors or unexpected situations (moving to the error state)?
- Should the Assistant also manage order cancellations or refunds, or is that a decision better left to the user?

## Closing Remarks

As we reach the end of this journey, take a moment to reflect on how the entire process was smoothly executed through a **collaboration of specialized objects**. Each object played a distinct role, holding critical information and managing its own **state** to ensure global consistency throughout the workflow:

- The **Beverage** object represented the available products and managed stock availability.
- The **Order** object controlled the lifecycle of the purchase, moving through key states like pending, paid, confirmed, and served.
- The **Payment** object ensured that no service was delivered without successful payment, regulating financial consistency.
- Finally, the **Assistant** orchestrated the process seamlessly, connecting these objects and automating the workflow by invoking the right methods at the right time.

All these elements worked together under the watchful eye of the state machines, ensuring that each transition happened in the correct sequence. This maintained not just operational flow but also **data integrity and business rules compliance**.

And to make sure everything works exactly as expected, **automated tests** validated the behavior of these components—proving that automation is not only about efficiency but also about building trust in the system’s reliability.


With this, you’re now equipped with the skills to design, automate, and validate smart processes that deliver both excellent user experiences and robust backend consistency.

## **Bonus Tip – Orchestrating Beyond the App**

Think of your Assistant as a skilled **quest master** in a game. It doesn't need to know how every challenge works—it simply knows which ones to trigger and when.

Sometimes, it calls **local functions**—like asking the kitchen to prepare Emma's drink. Other times, it sends a request to a **distant API**, like ordering rare ingredients from a faraway supplier. And occasionally, it doesn't wait around for results—it sends a command and moves on, trusting the kitchen will notify when the job is done (**asynchronous calls**).

Even more exciting, sometimes *other systems* knock on your door—an external app might call your Assistant to trigger a new process. In this interconnected world, automation isn't limited to one app; it's a lively marketplace of services and smart interactions happening all around you!

 *In process automation, it's all about knowing who to call, when to wait, and when to simply say: "I'll check back later!"*

## **Final Quiz – Are You Ready to Automate Like a Pro?**

### **1. In the SwiftSip™ story, what event triggers the entire process?**

- a) Payment is made
- b) Emma feels thirsty and opens the app
- c) The barista starts preparing the drink
- d) Emma takes a sip

### **2. What is the primary role of the Assistant object?**

- a) To display beverage prices
- b) To automate the full ordering process without manual steps
- c) To handle payment failures
- d) To check beverage stock manually

### **3. Which of the following is NOT a state of the Order object?**

- a) pending
- b) delivered
- c) paid
- d) confirmed

**4. Why are Non-Functional Requirements (NFRs) important in process automation?**

- a) They help choose beverage flavors
- b) They ensure the app works securely, reliably, and fast
- c) They define new objects in the app
- d) They allow more users to click buttons manually

**5. What is the final state of the Assistant object after successfully completing the happy path?**

- a) pending
- b) done
- c) error
- d) waiting

**Bonus Reflection:**

*Think of a simple daily process you would love to automate (at home or work).  
What objects and actions would you define?*

## CONCLUSION

In this module, you explored the essential concepts and practices behind **process automation**—not through abstract theory, but by stepping directly into a real-world scenario with Emma and the SwiftSip™ app.

You learned how to model workflows using simple yet powerful objects, each managing its own state and contributing to a consistent, reliable process. From manual interactions to full automation with the Assistant, you saw how business logic and user experience come together through state transitions and well-defined actions.

Most importantly, you discovered that automation isn't just about speeding things up—it's about delivering **clarity, consistency, and satisfaction** for both users and organizations. By introducing orchestration through the Assistant, you took your first steps toward designing smart, scalable solutions that reduce complexity and empower people to focus on what really matters.

With these skills, you're well on your way to becoming a thoughtful and capable process designer—able to build systems that are not only efficient but also resilient and user-centered.

🎓 *Your next challenge? Go beyond the "happy path." Explore how to handle errors, cancellations, and complex decision points. This is where great process designers truly shine.*

## REFERENCES

**Sutherland, J., & Schwaber, K. (2020). The Scrum Guide™.**

*Essential reading to understand agile product ownership and iterative process improvement.* <https://scrumguides.org>

**Rumbaugh, J., et al. (2004). UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design.**

*A foundational guide for modeling object states and mastering state machine exercises.*

**Fowler, M. (2002). Patterns of Enterprise Application Architecture.**

*An excellent resource for understanding how real-world systems handle processes, workflows, and service orchestration.*

**Brandolini, A. (2019). Event Storming: An Introduction.**

*Learn how to visualize complex business processes through collaborative event-based workshops.*

**Scott, A. (2020). Event Modeling: Structuring Event-Driven Information Systems.**

*A modern approach to design event-driven architectures and ensure clear, testable workflows.*

**Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). Fundamentals of Business Process Management (2nd ed.).**

*A comprehensive academic reference on workflow modeling, process optimization, and business automation.*

---

This chapter is licensed under the Creative Commons Attribution–NonCommercial 4.0 International License ([CC BY-NC 4.0](https://creativecommons.org/licenses/by-nc/4.0/)). Free use, reuse, adaptation, and sharing are permitted for non-commercial purposes. Author: Michel Zam (KarmicSoft)

