Process and Robotic Process Automation

Selecting what to automate, robotic process automation, scripting, and orchestration.

This guide accompanies Module 3, which moves from data pipelines to the wider processes that surround them. It examines how to model a process and choose a first automation, what robotic process automation (RPA) is and when it is appropriate, how to script a repetitive task safely, and how to schedule and orchestrate automated work. Claims are supported by peer-reviewed studies of automation and process analysis in healthcare, and worked examples and short code illustrate the ideas.

A theme running through the module is that selection matters as much as implementation. Cross-industry evidence shows that the appropriate level of automation depends on how well a task is defined, how repetitive it is, and how much human decision-making it requires, and that identifying goals and monitoring outcomes are critical to success [1]. Choosing the right first task is therefore a large part of the work, and this guide gives that choice sustained attention before turning to tools.

Learning outcomes

After working through this guide you will be able to:

- Model a process as steps with actors, inputs, and outputs, and score its automation potential.
- Explain how process mining can reveal real clinical pathways from event data.
- Weigh value against effort to choose a safe first automation.
- Explain what RPA is, the tasks it suits, and its limitations relative to a direct interface.
- Turn a repetitive task into a readable, error-handled, logged script.
- Schedule automated work and orchestrate several steps into a dependable, guarded flow.

1. Process modelling and selection

1.1 What a process model is

A process model is a simple, explicit description of a workflow as ordered steps, each with its inputs, its outputs, and the actor responsible. It extends the workflow map of Module 1 by naming who or what performs each step. That addition is what reveals automation opportunities, because it shows which steps are currently done by people and might move to software, which are already automated by instruments or existing systems, and which must remain human.

1.2 From map to model

Building a model from a map is a small addition: for every step, record the step itself, the actor, the input, and the output. You already captured steps, inputs, and outputs when mapping; adding the actor completes the picture. With these four columns filled in for a real, recent case rather than the idealised official procedure, the model is complete enough to reason about.

1.3 Scoring automation potential

Candidate steps are then scored against criteria that the automation literature consistently emphasises: a step is a strong candidate when it is repetitive, rule-based, stable over time, and has well-defined inputs and outputs [1]. Stability deserves emphasis, because a step can be repetitive and rule-based yet change so often that maintaining its automation costs more than it saves. Score each step high, medium, or low against these criteria; a step that scores high on several rises to the top of the list.

1.4 Weighing value against effort

Scoring potential is not the same as deciding what to automate first. A second axis, value against effort, governs that choice. The best first project is one that offers high value for low effort, because it proves the approach quickly and builds the confidence and trust needed for harder work later. Case studies of automation in healthcare reinforce the value of a structured selection method: a lean digital-transformation study used a formal define-measure-analyse-improve-control approach to identify wasteful, non-value-added steps before automating them, reducing process time substantially and raising process cycle efficiency from 69 to 96 percent [3].

1.5 Discovering real processes with process mining

A recurring difficulty is that the process people describe is not always the process they actually follow. Process mining addresses this by reconstructing the real process from the event logs that information systems already record. A tertiary review by Santos and colleagues, synthesising eighteen secondary reviews, found that process discovery is the most common activity and that process mining is an effective tool for analysing healthcare workflows and improving understanding of clinical guidelines and protocols, while noting the challenge of noisy or missing data [10]. A recent case study applied process mining to the care pathways of long-term respiratory patients and identified critical points where care could be improved [11]. For a course on automation, the lesson is that event data can validate or correct a hand-drawn workflow map before any automation is built on it, reducing the risk of automating a process that does not match reality.

> **Key idea.** Model the process and name the actor for each step; where event logs exist, use process mining to check the real pathway; then score steps and choose one high-value, low-effort, safely checkable step to automate first.

2. Robotic process automation: strengths and limits

2.1 What RPA is

Robotic process automation is software that carries out a routine task by following a defined sequence of steps, often by operating existing applications through their user interface, clicking, typing, and copying as a person would. There is no physical robot; the robot is a script that drives software. Its defining trick, operating applications the way a person would, is both its greatest strength and its main weakness.

2.2 Where RPA shines

RPA's strength is that it can automate systems that offer no other way in. In healthcare administration, RPA has been applied to appointment scheduling, data entry, billing, and claims handling, reducing costs and freeing staff for higher-value work [2]. Similar benefits are reported in health insurance, where RPA

automates predictable administrative duties such as claims processing and policy renewals [4], and in the lean case study noted above, where RPA replaced manual operations in a complex claims process [3]. The sweet spot is a stable, repetitive task on a system that exposes no usable interface, where building a proper integration is impractical.

2.3 The limits of RPA

The same literature is candid about RPA's limitations. Because a bot depends on the surface of an application, it can break when that surface changes, and large collections of bots become difficult to maintain. Reported barriers to adoption include data privacy and security concerns, resistance to change, and technical complexity [2]. RPA can also postpone the harder work of building a real integration, leaving an organisation dependent on fragile bots. It is best understood as a workaround: valuable, sometimes necessary, but not a permanent foundation.

2.4 RPA compared with a script or API

The choice between RPA and a plain script follows from a single difference. RPA drives an application through its surface, which is good when no interface exists but fragile when the screen changes and often reliant on a dedicated tool. A script or API connects to a system directly, which is more robust to visual changes and cheaper to maintain, but needs an interface to exist in the first place. Where a system exposes a proper interface, such as a FHIR API, a script is almost always the sturdier choice; RPA is for the cases where nothing better is available.

2.5 Barriers, governance, and realistic expectations

Adopting RPA is an organisational undertaking as much as a technical one. The healthcare literature repeatedly notes that data privacy and security concerns, resistance to change, and technical complexity are significant barriers to adoption [2]. Because a bot acts with the access of a human user, it must be governed like one: its permissions should follow the principle of least privilege introduced in Module 6, its actions should be logged, and its behaviour should be monitored so that a silent failure does not go unnoticed. Realistic expectations also matter. The most successful reported deployments pair RPA with genuine process redesign rather than automating a wasteful process unchanged; the lean case study achieved its large efficiency gains precisely because it first removed non-value-added steps and only then automated what remained [3]. Automating a bad process simply produces a faster bad process.

> **When to choose RPA.** Prefer RPA when a system lacks a usable interface, the task is stable, and a direct integration is genuinely impractical. Where an interface exists, prefer a script or API for robustness and lower maintenance.

3. Scripting repetitive tasks

3.1 Think first, code second

Where a direct route exists, a small script is often the best tool. Turning a manual task into a script begins not with code but with clear thinking: describe the task in a sentence, break it into small actions, and define what success looks like. When people struggle to write a script, it is usually because the task was never made clear, not because the coding was hard.

3.2 A worked example

Consider a realistic, safe task: each day, new result files arrive in one folder, and each must be moved into a folder named for its date. A short script does this:

```
from pathlib import Path
import shutil, datetime

inbox = Path("inbox")
day = str(datetime.date.today())
target = Path(day); target.mkdir(exist_ok=True)

for f in inbox.glob("*.csv"):
    shutil.move(str(f), str(target / f.name))
    print("moved", f.name)
```

A small, readable file-sorting script. It makes a dated folder, moves each CSV into it, and prints each move as a simple log.

3.3 Making it safe with error handling

The basic script works, but to run unattended it needs error handling, because reality is messier than a demonstration. It should check that the inbox folder exists before starting, skip and note any file that cannot be moved rather than stopping the whole run, never overwrite a file already at the target, and stop clearly if something truly unexpected happens. A script without error handling is a demonstration, not a tool.

3.4 Leaving a trail with logging

The final habit is logging, which answers a simple question: could you tell tomorrow morning whether last night's run succeeded? To pass that morning test, record how many files were processed, note any that were skipped and why, timestamp each run, and write the log where a person will see it. These habits, readability, error handling, and logging, are echoed in tutorials on practical software engineering for scientists, which recommend version control, testing, documentation, and continuous integration as the practices that keep research software maintainable as it grows [5].

> **Practical rule.** Think first, code second. Make the script readable, check its inputs before it acts, never overwrite or lose data, and ensure every run leaves a log you can read.

4. Scheduling and orchestration

4.1 Scheduling

Scheduling arranges for a task to run automatically at set times or intervals, removing the human trigger and finally making automation hands-off. It also raises the stakes, because a scheduled task runs while no one is watching, so error handling and logging become essential rather than optional.

4.2 Ways to trigger

Automation can start in several ways, and matching the trigger to the work matters. A timer suits regular batch work; an event, such as a new file arriving, suits irregular but time-sensitive work; a sequence trigger, where a step starts when the previous finishes, is the basis of orchestration; and an on-demand trigger lets a person or another system start the work when needed.

4.3 Orchestration and the guarded chain

Orchestration coordinates several automated steps so that each runs in the right order, at the right time, with failures handled between them. Its essential element is the guard between steps: run a step, check whether it succeeded, decide whether to continue, stop, or alert, and log the outcome. This small loop, repeated between every pair of steps, prevents a failure in one step, such as failed validation, from cascading into corrupted output downstream.

4.4 Silent failure and the safeguards against it

The characteristic risk of unattended automation is not failure itself but silent failure, a process that breaks without telling anyone and is trusted for days afterwards. The remedy is the discipline established across this course: a log of every run, an alert when something fails, a safe stop that prevents cascades, a check that the expected input was present, and a way to re-run from clean data. These safeguards, together with the selection and scripting practices above, turn the tasks you choose into automation that runs dependably without supervision, which is the purpose of the whole exercise.

> **Key idea.** Between every two orchestrated steps sits a check. Silent failure, not failure, is the enemy of unattended automation; logging and alerting are what keep it trustworthy.

5. Case studies from the literature

It is worth grounding this module's principles in specific published deployments, because the details of what worked, and what did not, are instructive. Three strands of evidence are especially relevant: administrative RPA, lean-driven RPA, and process mining.

5.1 Administrative RPA in healthcare

Domb and colleagues survey the embedding of robotic process automation in healthcare administration, describing applications in appointment scheduling, data entry, billing, and claims handling, and reporting reductions in cost and staff burden [2]. Their account is valuable precisely because it pairs the benefits with candid barriers, data privacy and security concerns, resistance to change, and technical complexity, that any real deployment must manage. Kumar's study of RPA in health insurance reaches similar conclusions for predictable administrative duties such as claims processing and policy renewals, where the work is high in volume and stable in its rules [4]. Together these studies map the natural territory of RPA: repetitive administrative work on systems that expose no better interface.

5.2 Lean-driven RPA

Huang and colleagues provide a particularly clear demonstration of the selection principle at the heart of this module. Applying a formal lean method, they first analysed a complex claims process to identify wasteful, non-value-added steps, and only then automated what remained. The result was a substantial reduction in process time and an increase in process cycle efficiency from sixty-nine to ninety-six percent [3]. The lesson is that automation delivered its gains because it followed process redesign rather than replacing it; automating the original, wasteful process would have produced a faster version of the same waste.

5.3 Process mining as a check on assumptions

Finally, the process-mining literature offers a way to ground selection in evidence rather than assumption. Santos and colleagues, in a tertiary review synthesising eighteen secondary reviews, find that process discovery is the most common activity and that process mining effectively supports the analysis of healthcare workflows and compliance with clinical guidelines, while cautioning about noisy or missing data [10]. A recent case study reconstructs the care pathways of long-term respiratory patients from clinical data and identifies concrete points where care could be improved [11]. For an automation project, such techniques can confirm or correct a hand-drawn map before any code is written, reducing the risk of automating a process that does not match reality.

> **Key idea.** The strongest reported automations combine careful selection, process redesign, and evidence about the real process, not merely a tool applied to an unexamined workflow.

6. Further reading and practice

This module is best consolidated by carrying one small task from selection through to a scheduled, guarded script. The exercises below are designed to take an afternoon and to produce material you can reuse in the module assignment, which asks you to automate a small repetitive task and explain what could go wrong. Use only synthetic or non-sensitive data.

For deeper reading, the workflow-automation review [1] sets out criteria for selecting tasks, the RPA studies [2], [3], [4] give concrete healthcare deployments and their pitfalls, and the process-mining literature [10], [11] shows how event data can reveal the process people actually follow. The software-engineering tutorial [5] is a compact guide to the habits that keep scripts maintainable.

- Model a real process of your own as steps with actors, inputs, and outputs, and score each step for automation potential.
- Decide, with reasons, whether your chosen step is better served by RPA or by a script or API.
- Write and run a small script for the step, adding one deliberate error-handling case and a simple log line.
- Describe how you would schedule the script and what single alert you would send on failure.
- Read the abstract of the lean RPA study [3] and note how process redesign preceded automation.

Completing these exercises will leave you with a selected task, a working script, and a clear plan for running it unattended, which together form the core of the module assignment.

Key terms

Process model — An explicit description of a workflow as ordered steps, each with an actor, inputs, and outputs.

Process mining — Reconstructing the real process followed in practice from the event logs that information systems record.

Robotic process automation (RPA) — Software that performs a routine task by operating existing applications through their interface, as a person would.

Value versus effort — A selection heuristic that favours a first automation offering high value for low effort.

Error handling — Code that makes a script fail safely and informatively when reality departs from the expected case.

Scheduling — Arranging for a task to run automatically at set times without a human trigger.

Orchestration — Coordinating several automated steps in order, with failures checked and handled between them.

Silent failure — A failure that occurs without notifying anyone, so that broken automation continues to be trusted.

Self-check questions

1. Model a process from your work as steps with actors, inputs, and outputs. Score each step for automation potential and justify your top candidate.
2. Explain how process mining could check whether your hand-drawn map matches the process people actually follow.
3. Give a task for which RPA is the right tool and one for which a direct API script is better, and explain the difference.
4. List the three qualities that make a script safe to run unattended, and explain the 'morning test'.
5. Rewrite the file-sorting example in your own words and add one sentence describing an error it should handle.
6. Describe the guarded-chain pattern of orchestration and explain what it prevents.
7. Match each of the four trigger types to a task for which it is appropriate.
8. What is silent failure, and which specific safeguards reduce its risk in a scheduled job?

References

Peer-reviewed sources located via the Consensus academic search tool; standards cited from their issuing bodies. Formatted in IEEE style.

[1] T. Zayas-Cabán, K. Marquard, and E. Radhakrishnan, "Identifying opportunities for workflow automation in health care: lessons learned from other industries," Applied Clinical Informatics, vol. 12, no. 3, 2021. [Online]. Available: https://consensus.app/papers/details/9c7a21d887485050b87bd465c8713460/

[2] M. Domb et al., "Empowering healthcare administration by embedding robotic process automation," International Journal of Biomedical Research & Practice, 2024. [Online]. Available: https://consensus.app/papers/details/cd0d89ca396f5a269b710ac1b9ec253e/

[3] W.-L. Huang et al., "A case study of lean digital transformation through robotic process automation in healthcare," Scientific Reports, vol. 14, 2024. [Online]. Available: https://consensus.app/papers/details/49eb76e74b395040b4eb98f31ce22c12/

[4] M. Kumar, "Robotic process automation (RPA) in healthcare insurance: streamlining administrative tasks," Progress in Medical Sciences, 2022. [Online]. Available: https://consensus.app/papers/details/acdc5b5385085ba9ba969c1aa0c4e35c/

[5] R. P. Peláez, "Practical software engineering for software-writing scientists," The Journal of Chemical Physics, 2025. [Online]. Available: https://consensus.app/papers/details/f97a0be33de452c9819b4320969fde4a/

[10] A. Santos et al., "Process mining in healthcare: a tertiary study," BMC Medical Informatics and Decision Making, 2025. [Online]. Available: https://consensus.app/papers/details/94870eacb6115504bffd4295956081b8/

[11] L. Murazzano et al., "Clinical pathways discovery for long-term and chronic patients: a process mining approach," Computers in Biology and Medicine, 2026. [Online]. Available: https://consensus.app/papers/details/c887b3c73b605049a8e8cb9e4bc14651/