Lab Automation: Which Processes Pay, and Which Ones Do Not

A category built on optimism deserves a skeptical test. Three conditions decide whether a process is worth automating, and knowing which of yours qualify is the whole exercise.

Brandon Holland, Marketing & Operations Digital Solutions Architect

August 20, 2026 · 5 min read

IN BRIEF

Lab automation pays where a task is high volume, low judgement and methodologically stable. Where any one of those three is missing it usually costs more than it returns, because the exceptions consume the saving. Most laboratory work fails at least one condition, which is why candidate selection matters more than technology choice.

Key takeaways

- Three conditions decide it: high volume, low judgement, stable method. All three, not two.
- The saving is destroyed by exception handling, so count the exceptions before the volume.
- A process changing method more than once a year is a poor candidate regardless of volume.
- Sample preparation is usually a better first target than analysis, and gets chosen second.
- If nobody can describe the current process in one page, automating it will encode the confusion.

Every lab has a list of things it would automate given the budget. Most of those lists are ordered by how annoying the task is, which is a reasonable instinct and a poor predictor of whether automating it will pay.

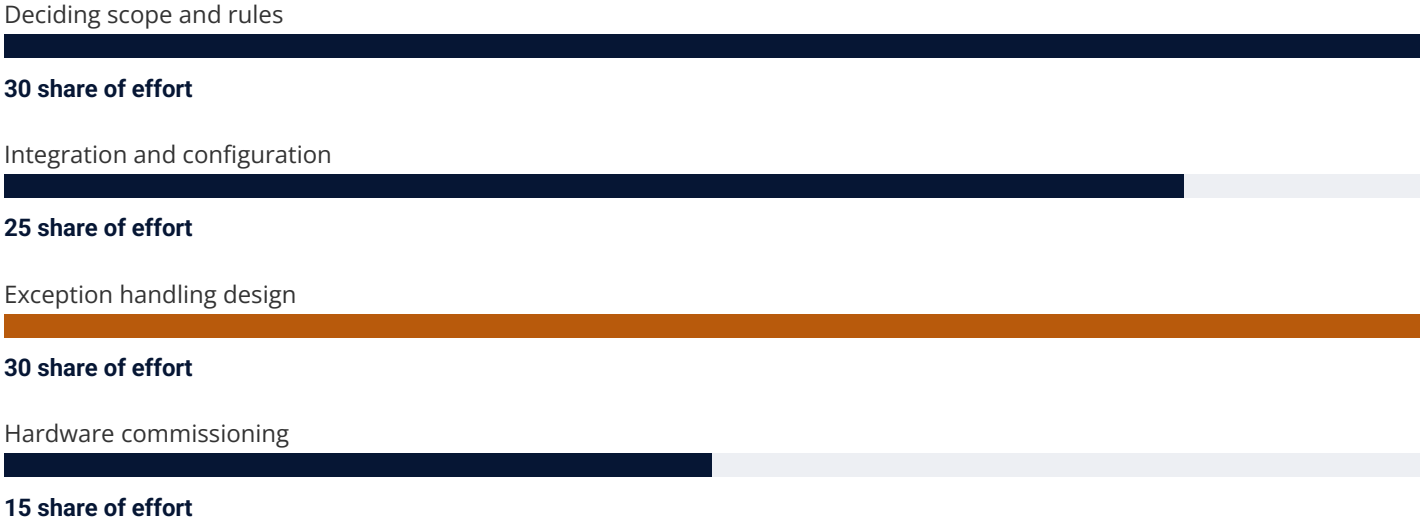
Automation pays where three conditions hold together. Miss any one and the saving is consumed by exceptions a person still handles.

> **A qualifying process**
>
> High volume, low judgement, and a stable method. All three, not two. Most laboratory work fails at least one, which is why candidate selection matters more than technology choice.

What makes a process worth automating?

- **Volume.** High enough that a per-run saving accumulates into something visible. This is the condition labs assess correctly, and often the only one.
- **Low judgement.** The decision can be written as a rule. If an experienced analyst decides on the basis of experience, that step becomes an exception queue with a person at the end of it.
- **Method stability.** The configuration will not be rewritten. A method changing more than about once a year is a poor candidate at any volume, because each change is a reconfiguration and, in a regulated lab, a revalidation.

WHERE EFFORT LANDS ON A FIRST AUTOMATION PROJECT



Composite of the phases described here

The third bar is the one to look at. Exception handling is about as large as the configuration and gets estimated last, usually once the happy path already works.

Why does exception handling destroy the business case?

Because the saving is calculated on the happy path and the cost is incurred on everything else.

EXCEPTION RATE	WHAT THE CASE USUALLY CLAIMS	WHAT ACTUALLY HAPPENS
Under 2%	Fully automated	Broadly true; the residue is absorbed
5% to 10%	90 to 95% automated	A second, more skilled manual path now exists alongside the first
Over 15%	Automated with edge cases	Two processes, and staff must first work out which one a sample is in

Counting exceptions before counting volume is the more useful order. A process with a ten percent exception rate and no exception design is not ninety percent automated. It has a new failure mode.

Which processes qualify more often than labs expect?

- **Sample preparation.** Repetitive, error-prone, judgement-free. Chosen second because the instrument is the expensive object, though instrument time is rarely the real bottleneck.
- **Transcription.** Every point where a number is read off one screen and typed into another. The cheapest available win and often not framed as automation at all, because the fix is an interface rather than a robot.
- **Review routing.** Deciding who looks at what, by rule instead of by convention, so exceptions surface rather than wait.

How do you find the candidates you already have?

By counting touches, not by asking people what annoys them. Walk one sample through the lab and record every point at which a human moves it, reads it, writes about it, or decides something. Then sort the touches:

1. **Moves** need hardware, and are the expensive category.
2. **Transcriptions** need an interface, and are almost always the cheapest win.
3. **Decisions** need a rule. Where the rule cannot be written, the decision stays with a person and that is the correct answer.

Most labs doing this find their best first candidate is a transcription they had stopped noticing, not the instrument they were planning to buy a handler for. LIMS integration covers the mechanics and the lab automation hub the wider category.

What does automation change about validation?

It moves the burden from the result to the decision. A manual step is validated by demonstrating that a competent analyst following the procedure produces the right answer. An automated step has to demonstrate that the rule produces the right answer across the range of inputs it will actually see, including the ones nobody thought about.

That is a wider evidence requirement, not a narrower one, and it is the part most often underestimated:

- **Boundary conditions.** What the rule does at exactly the threshold, not comfortably either side of it.
- **Failure behaviour.** What happens when an instrument returns nothing, or returns something malformed, rather than something wrong.
- **Change control.** Who can alter the rule, and what revalidation their change triggers.

Where these projects actually go wrong

- **Scope that grows quietly.** Phase one starts as four instruments and is eleven by month three, because each addition was individually reasonable and nobody decided to double the project.
- **Automating an undescribed process.** If the current process cannot be written on one page that two people agree with, automation encodes the disagreement and hides it.

- **Assuming headcount falls.** What changes is usually what people do, moving effort from handling to review and troubleshooting. A case built on redeployment survives contact with reality; one built on reduction invites a reckoning.

> NOTE
>
> Some processes should stay manual. A low-volume, high-judgement test performed by one experienced analyst is a sensible allocation of a skilled person, not a failure of modernisation.

What a LIMS carries in this work

The unglamorous half, and usually the half that pays first: receiving instrument output directly so transcription disappears, routing review by rule so exceptions surface, and recording what the automation did as auditable events. That last one matters because 21 CFR 11.10(a) asks for systems validated to give "consistent intended performance", and an automated step with no record cannot demonstrate it.

Hardware automates the handling. Software decides what happens when the handling does not go to plan, which is where the difference between a project that pays and one that does not usually sits. Our integration and automation page covers scope, and automation agents where newer tooling genuinely helps.

> **How LabLynx handles this**
>
> LabLynx handles the capture, routing and exception surfacing above, which is the software half rather than the hardware half. If your bottleneck is genuinely physical throughput, a LIMS will not fix it and we would point you at instrument vendors first.

Frequently asked questions

What makes a laboratory process a good automation candidate?

Volume high enough that the per-run saving accumulates, judgement low enough that the decision can be expressed as a rule, and a method stable enough that the configuration will not be rewritten. All three have to hold. Two out of three is the profile of a project that runs over, because the missing condition becomes exception handling that a person still performs.

Why do automation projects over-scope?

Because each addition is individually reasonable. A second instrument, three more report templates, one more sample type: none is unreasonable and the sum doubles the work without anyone deciding to. The defence is a written phase-one boundary with a named owner, and treating additions as a change to the schedule rather than as details.

Should we automate sample preparation or analysis first?

Usually preparation, and labs usually choose analysis. Preparation is more repetitive, more error-prone and less judgement-dependent, which is precisely the profile that automates well. Analysis feels like the bigger prize because the instrument is expensive, but instrument time is rarely the actual bottleneck in a working lab.

Does automation reduce headcount?

Rarely in the way business cases assume, and claiming it invites a reckoning. What it more often changes is what people spend time on, moving effort from repetitive handling to review, troubleshooting and method work. A case built on redeployment survives contact with reality better than one built on reduction.

Sources and references

1. 21 CFR 11.10 Controls for closed systems

   U.S. Government Publishing Office, Electronic Code of Federal Regulations

Brandon Holland

Marketing & Operations Digital Solutions Architect

Brandon Holland is LabLynx's Marketing & Operations Digital Solutions Architect. He builds and runs the digital systems behind the company's public surface, from the website and content architecture to the tooling that keeps product information accurate everywhere it appears. He writes about laboratory informatics from the systems side: how lab records get structured, searched, and kept defensible as a lab scales.