# How Does a LIMS Work, and What Makes an Implementation Fail?

A LIMS is not hard to buy and it is hard to implement. The difference is a short list of decisions that get made badly, early, and quietly, long before anyone signs a contract.

**Brandon Holland**, Marketing & Operations Digital Solutions Architect

August 18, 2026 · Updated August 26, 2026 · 8 min read

IN BRIEF

How does a LIMS work? It makes the sample the unit of record. Every aliquot, transfer, method, instrument reading and approval attaches to that sample as a time-stamped entry, so any result can be reconstructed years later. How well it works depends on four decisions made before purchase, not on the software.

### Key takeaways

- A LIMS makes the sample, not the report, the unit of record. Everything else follows from that.
- Four decisions set the outcome: data model ownership, instrument scope, what the lab stops doing, and validation sign-off.
- Integration effort is driven by how many instruments produce a proprietary file, not by how many instruments exist.
- FDA expects audit trail review at the same frequency as review of the data itself, which is a staffing decision.
- A lab that cannot describe its current sample flow on one page is not ready to write requirements.

You have been asked to look at a LIMS. Maybe an auditor wrote something up, maybe a technician spent a Friday reconciling two spreadsheets that disagreed, maybe the lab simply grew past what the current system holds. Either way the question in front of you is not what the software does in a demo. It is what it will take

to run your lab on it.

**Most LIMS projects that go badly do not go badly on software.** They go badly on four decisions made before anyone saw a demo: who owns the data model, which instruments are genuinely in scope, what the lab agrees to stop doing, and who signs off validation. Every one of those is a laboratory decision, and every one of them is cheap to change in month one and expensive in month seven.

This covers how a LIMS actually works, then those decisions in the order you have to make them.

## What does a LIMS actually do?

A LIMS makes the sample the unit of record. That single sentence is the whole design. In a spreadsheet or a paper logbook, the unit of record is the report, and the sample is a row inside it. In a LIMS it is the other way around: the sample exists first, and every aliquot, transfer, method, instrument reading, calculation and approval attaches to it as a time-stamped entry.

If you are earlier than this and still comparing categories of system, the LIMS resource hub is the better starting point, and what a LIMS is covers the definitions this article assumes.

The practical consequence is reconstruction. Two years after a result leaves your lab, someone can ask who received the sample, what condition it arrived in, which instrument ran it, which method version applied, who reviewed the number and what it was before it was corrected. A system built around samples answers that in one place. A system built around reports answers it by asking three people to remember.

- **Chain of custody.** Every handoff recorded with who, when and in what condition, rather than reconstructed from initials on a form.
- **Method and version binding.** The result carries the method revision that produced it, which matters the first time a method changes and an old result is questioned.
- **Instrument capture.** Readings arrive from the instrument rather than being retyped, which removes the transcription error nobody can find later.

- **Review as a recorded act.** Approval is an event with a person and a timestamp attached, not a signature at the bottom of a printout.

## What has to be decided before you shortlist?

Four things, and the order matters because each one constrains the next. If the lab is coming off spreadsheets rather than replacing an older system, the migration path has a different shape and is worth reading alongside this.

1. **Who owns the data model.** Someone has to decide what a sample is in your lab, how it relates to a batch, a project and a client, and which of those relationships are allowed to be many-to-many. This is not an IT decision and it is not the vendor's decision. It needs someone in the lab with the authority to settle a disagreement between two sections that each have a reasonable definition.
2. **Which instruments are in scope.** Not which instruments exist. Which ones will send data to the LIMS in phase one. This list is almost always too long on the first pass.
3. **What the lab stops doing.** A LIMS that runs alongside the old spreadsheet has not replaced it, and now there are two systems of record that disagree. Naming the things that stop, and who confirms they have stopped, is the difference between a migration and an addition.
4. **Who signs off validation.** Decided at the start, because it determines how much documentation the project produces, and documentation written retrospectively costs several times what it costs written as you go.

> NOTE
>
> The requirements document is an output of these four decisions, not a substitute for them. A lab that cannot describe its current sample flow on a single page is not ready to write requirements, and will end up describing the vendor's workflow back to itself.

## Why does integration take longer than expected?

Because the effort is not proportional to the number of instruments. It is proportional to the number of instruments producing a format nobody else reads.

An instrument with a documented export, a stable file layout and a vendor who answers email is a few days of work. An older instrument writing a proprietary binary that changes between firmware versions can absorb weeks, and the work is not reusable for the next instrument. Counting instruments gives you a schedule that is wrong in a specific direction: too optimistic, and wrong by more the older your lab is.

Before committing to a date, sort every in-scope instrument into three buckets:

1. **Documented export.** A stable layout and a vendor who answers email. Days of work.
2. **Undocumented but text.** Parseable, fragile, and it breaks on firmware updates.
3. **Proprietary binary.** Weeks, and the work is not reusable for the next instrument.

Then plan on the third bucket consuming most of the integration time regardless of how few instruments are in it. For interface patterns, the integration services page covers the approaches, and application integration covers connecting a LIMS to systems that are not instruments at all.

WHERE THE MONTHS ACTUALLY GO ON A MID-SIZE IMPLEMENTATION

**Weeks 1 to 4**

Data model and requirements. Cheap to change here and nowhere else.

**Weeks 4 to 10**

Configuration. Usually the part that runs to schedule.

**Weeks 8 to 20**

Instrument interfaces. Overlaps configuration and is the usual source of slip.

**Weeks 16 to 28**

Validation and user acceptance. Compresses only by deferring work, not removing it.

**Weeks 24 to 36**

Parallel running, then retirement of what the LIMS replaced.

## What does validation actually require?

If your lab holds electronic records under a regulation, the obligation is written down and it is short. 21 CFR 11.10(a) requires *"validation of systems to ensure accuracy, reliability, consistent intended performance, and the ability to discern invalid or altered records."* That last clause is the one labs underestimate: the system has to be able to show that a record was altered, not merely prevent it.

The companion requirement is the audit trail, and 11.10(e) is specific in a way worth reading closely. It requires secure, computer-generated, time-stamped audit trails recording operator entries and actions, and then adds a sentence that decides a great deal: *"Record changes shall not obscure previously recorded information."* A system where correcting a value replaces it does not satisfy that, however careful the correction.

The part that becomes a staffing decision rather than a software one is review frequency. FDA's Data Integrity and Compliance With Drug CGMP guidance, issued December 2018, answers it directly: where a regulation specifies a review frequency for the data, the same frequency applies to the audit trail; where it

does not, the lab determines the frequency using its own risk assessment. Either way someone is reading audit trails on a schedule, and that person has to exist in the plan. Our compliance services page covers the documentation set this produces.

## Where implementations actually go wrong

Not in the places the project plan worries about.

The most common failure is scope that grows quietly. Phase one starts as four instruments and eight report templates, and by month three it is eleven instruments and thirty templates, because each addition was individually reasonable. Nobody decided to double the project. It happened by accumulation, and the schedule was never revised to match.

The second is parallel running with no end date. Parallel operation is sensible for a defined period and corrosive once it becomes the normal state, because staff will use whichever system is easier and the LIMS becomes a place where data is copied to rather than created. Set the retirement date at the start and treat it as a deliverable.

The third is honest and worth saying plainly: **some labs should not implement a LIMS yet.** A lab with three people, one instrument and stable methods may be better served by fixing its sample numbering and its backup discipline first. The signal that a LIMS is genuinely warranted is usually not volume. It is that more than one person needs to answer questions about the same sample, and they currently answer them differently.

- The current sample flow fits on one page. Two people agree it is accurate
- A named person in the lab owns the data model and can settle a dispute
- In-scope instruments are sorted by export format, not counted
- The list of things the lab stops doing exists, with a date and an owner
- Audit trail review frequency is decided, with a person named against it
- Validation sign-off is agreed before configuration starts, not after

## What a LIMS carries in this work

The mechanism, rather than the benefit. A LIMS enforces the sample as the unit of record, which is what makes reconstruction possible. It records custody transfers as events rather than as signatures. It binds a result to the method version that produced it. It captures instrument output without a person retyping it. And it keeps a change history that shows the prior value rather than replacing it, which is the specific thing 11.10(e) asks for.

None of that removes the four decisions. The software cannot decide what a sample is in your lab, and a system configured against a data model nobody owns will reproduce the confusion it was bought to fix, faster and with an audit trail.

> **How LabLynx handles this**
>
> The LabLynx LIMS is configured against your data model rather than shipping one, which is why the first phase of our implementations is a data model workshop and not a software install. That suits labs willing to make the four decisions above. It suits a lab looking for something to switch on next week considerably less.

## Frequently asked questions

### How long does a LIMS implementation take?

For a mid-size lab with a handful of instrument interfaces, four to nine months from kickoff to routine use is typical. Configuration is rarely the long pole. The time goes into agreeing the data model, mapping instrument outputs, and validation. Labs that compress the schedule usually do it by deferring validation, which moves the work rather than removing it.

### Do we need to validate a LIMS if we are not FDA regulated?

Formal computer system validation is required where a regulation demands it, and 21 CFR 11.10(a) is explicit for closed systems holding electronic records. Outside that, most accreditation bodies still require documented evidence that the system does what you say it does. The document set is lighter. The obligation to demonstrate control does not disappear.

### What is the difference between LIMS configuration and customisation?

Configuration uses settings the vendor supports and tests: fields, workflows, roles, report layouts. Customisation writes code the vendor does not maintain. The practical difference appears at upgrade time, when configuration carries forward and customisation has to be retested. Prefer configuration until a genuine requirement cannot be met, then document why.

### Can a LIMS replace our instrument software?

No, and treating it as a replacement is a common scoping error. Instrument control software runs the instrument. A LIMS receives what the instrument produces, attaches it to a sample, and holds it under audit. The two coexist, and the integration between them is where implementation effort concentrates.

### Who should own the LIMS data model?

Someone in the lab who understands how samples actually move, with authority to settle disagreements between sections. Not IT, and not the vendor. IT owns infrastructure and the vendor owns the software, but

the decision about what a sample is and how it relates to a batch is a laboratory decision that outlives both.

## Sources and references

1. 21 CFR 11.10 Controls for closed systems

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

2. Data Integrity and Compliance With Drug CGMP: Questions and Answers, Guidance for Industry

   U.S. Food and Drug Administration

**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.