LabLynx

LIMS Software: Build or Buy?

What are the trade-offs and considerations?

Author: Joe Liscouski July 2022

Introduction

There is a progression in technology utilization that many labs go through on their way to choosing LIMS software. A lot depends on the size of the organization, the resources it has available to it, and its experience in lab operations. Smaller groups in particular may begin with paper-based workflows and documentation (e.g., for test methods, etc.). As they tire of chasing paper ("who has the test request for ....?") and manually searching for samples, they may look for a better way of managing test requests, inventory, etc.

From there, the next step up is to use spreadsheets for different system functions (e.g., sample tracking, inventory management, etc.) since most personnel know how to use them and the means to create them are likely available on staff computers. This will work until:

- The updating process leads to lost or missing information, or the lab's headcount grows and the system becomes unwieldy and difficult to maintain. Adding data for a few dozen samples is manageable, but when the number starts getting into the hundreds or thousands, it's a nightmare. In regulated labs, all entries have to be verified, and as such the bookkeeping problem grows rapidly.

- Report generation (e.g., monthly summaries, results reports to sample submitters, etc.) and just sharing information with those who need it becomes a major time-sink.

- It becomes difficult to meet regulatory requirements. For example, audit trails are not an inherent feature of spreadsheets, but they are an essential fact of life in lab operations (i.e., "Has any of the data been changed?", "What were the previous values?", "Who authorized the change?"). They are part of FDA, EPA, ISO, and other regulations.

- Security is a problem. Maintaining access controls without compromising people's ability to work is difficult, and new versions of the underlying software can create frustration as new "features" change the functionality of those you are using.

- You want to improve lab efficiency by connecting instrument data systems/devices and having automatic data entry/updates.

After the headaches of spreadsheets, you are ready for something that will elevate your lab's operations. You decide the right informatics solution is a LIMS. And then someone raises this issue: Do you build it or buy it?

Defining user requirements in detail is essential

Before we get to that consideration, we have to define what "it" is. No matter which way you go (build or buy), you have to have an understanding of what your needs are, and those will define the project. Earlier articles have defined what LIMS capabilities are, but even within that sphere you have to clarify what your lab's requirements are in the coming years for testing, instrument connections, etc. That discussion is important and beyond the scope of this note, but it is one you and your colleagues have to have. If we were discussing a house, you'd want to define the number of bedrooms, garages, how the kitchen should be arranged, and so on; all houses aren't the same. The same holds true with a LIMS: we know the basic functionality, but the details of how it applies to your lab matters. Fortunately, there are tools to help you. In addition to books on the topic, ASTM International has created two guides to help you get started:

- ASTM E1578-18 Standard Guide for Laboratory Informatics[1]
- ASTM E1578-06 Standard Guide for Laboratory Information Management Systems (LIMS)[2]

ASTM E1578-18 largely supersedes E1578-06, but the latter has some useful information that is specific to LIMS. Both contain checklists of features and functions found in informatics products that are useful in deciding what you need for your lab.

Now that we've defined the project, what are the considerations in making the decision to build or buy?

Why would someone consider building their own LIMS?

1. Corporate management may have invested in an enterprise resource planning (ERP) system with the idea that it would provide the basis for all corporate data systems. This may hold true for many business applications, but labs are different due to the regulatory requirements and instrument connections.

2. The organization has an overly ambitious IT group. They may be supported by management, but do they understand the nature of the project? We'll cover some points related to this in a bit.

3. The organization believes that a system developed in-house is less costly, particularly if the organization has several labs.

4. Strong egos drive organizational decisions.

5. The organization believes an in-house solution will have better integration and security with corporate systems.

The drive to develop systems in-house is understandable. Management often wants to cut costs and utilize available resources when they can, but sometimes the effort is misplaced. My wife likes to shop at stores that carry both unique furniture pieces as well as mass-market items. She'll pick something out and my reaction is often, "I can make that." While this is true enough, it's not as simple at that price point. All she has to do is pick it up, pay for it, put it in the car, and then put it in place once we get home (some assembly required). If I were to build it, I'd have to design it, get materials and begin cutting, fitting, making improvements, and putting things together, along with inevitable delays as other urgent projects crop up. I'll save my woodworking for things we can't find elsewhere. The same types of issues come up in LIMS software development.

Considerations for systems developed in-house

There are several considerations to make here. First, a LIMS developed in-house will always be playing catch-up with a commercial system. Vendors have spent years developing and improving their products. They have a user community as well as a sales force that will tell them about issues with their offerings and what needs to be added to keep up with user requirements, and they have a dedicated team of people to implement those changes. Competitive pressures will keep vendors moving forward; if your development is in-house, what is going to drive upgrades to your LIMS? Unless you have an extraordinarily insightful user/developer team, your LIMS will always be behind products on the market.

Second, can all labs in the organization agree on needs? If part of the justification for in-house development is the ability to use the same basic product across multiple labs, can all those labs agree on the user requirements? It's possible to develop a core product that can be made adaptable to different lab needs, but which lab will come online first? Will support/bug fixing issues delay deployment in other labs?

Third, in-house development means a development team with only one customer. There was a company in the town I live in that was a large supplier of business stationery and forms. Rather than use a commercial software package to manage their business data, they built their own. Among the problems they encountered was that there was no outside expertise available to hire or consult with; they could hire people who knew the OS and underlying database systems, but no one knew their application. After all, they had to hire and train their developers and support staff. When someone left, that bit of knowledge and understanding went with them. It's hard to maintain a development staff for that kind of project; their expertise mattered only to that company, and if they looked for employment elsewhere, their backgrounds had limited applicability. How do you maintain the morale and drive to keep developing and supporting a limited application system?

Will the developer and project managers be able to provide long-term support? One of the problems with in-house development is that you are using resources that other projects want access to. The development work is never truly “finished,” it just moves from a period of intense effort to one with less pressure. However, the pressures of needing new features, bug fixes, updates to match changes in underlying systems, and general support never go away. Will those resources be available when you need them, or will you have to wait in the queue to get issues resolved?

Finally, is your development staff up to the task before them? Do they in fact understand the scope of the project? Questions arise, such as:

- What is their track record of handling large-scale projects (i.e., project management, development, and implementation)? Is there a history of successfully completed projects, or are those projects in a constant state of development and revision? Will you be working with the same development team, or is there a history of personnel turn-overs that can mean project restarts?

- What is the likelihood that developers will be pulled off the project to address “more pressing, higher priority” needs? Is senior management solidly behind the work, or will they become frustrated and shift resources in the face of delays in schedules and the need for more resources? (This is common on large projects: schedules are just the best guess and need to be refined as things progress.)

- Will they supply training and the documentation needed to support the system and users? This is a serious issue and needs to be appreciated and discussed early in the project’s definition. Supporting someone’s software is a mind-numbing task: you have to understand how they were thinking in order to make sense of things. The antidote for that problem is thorough documentation at the system level that addresses how things work, why things were done the way they were, and how to address problems if they develop. Ideally, the documentation should be done in parallel with the development while the concepts are still fresh in mind, but they often aren’t, particularly with a team that isn’t fully experienced in software engineering. And then there is user-level documentation. This should be prepared by a professional technical writer who has access to the system as it’s developed, who can think like a user and prepare the manuals so that they are easy to understand. They will also be a valuable resource in debugging the software and seeing things that don’t work the way they are supposed to. By extension, they then have to explain how things work, and if the writer is having a problem, then something in the design is wrong. Training is another related matter. “Read the user manual” is a backup to training, not training in itself. You want to bring people from the “deer in the headlights” stage to competent users. That may, at first, be in-person training that can be recorded, edited, and made available online. It might also be incorporated as part of the LIMS help function.

There are two additional points that deserve more than "bullet-level" attention in regards to project scope. The first is an item that separates laboratory software from almost anything else: instrument connections. The second has broader implications: the need to meet regulatory requirements (which may come from the FDA, EPA, ISO and other sources).

Does your development team understand the importance of instrument connections and how they can affect LIMS development? The measurements made in laboratory work are what makes labs what they are. Originally, they were made manually, but for the past 70 years electronic controls and interfaces to computers have changed that process. The trend has been toward the automation of sample runs with the aid of autosamplers and robotics. That has had an impact on laboratory informatics. We now expect LIMS to do more than catalog samples and document the testing that has to be done. They are expected to support instrument systems in a variety of ways:

- For single-point measurements (e.g., pH meters, etc.), instruments may be connected to the LIMS, with data passing between them to record measurements and perform computations on them, sometimes using data collected directly from balances and scales. The data transfer interfacing can be done using RS-232, 422, or 485 (i.e., serial data exchange), USB, or TCP/IP protocols. Those provide the electrical/data path connections, while the programming is provided by the user through the device's command/response protocols.

- Intermediate computers may be needed between the instruments and the LIMS. In this case, single-point measurement devices may be part of a more extensive data collection and processing than noted in the point above. An experiment may take measurements from several devices, process them, and produce single or multiple results. That can require more programming than you want to do in the LIMS. Instead, a separate computer can carry out the data collection and processing and then transfer it to the LIMS. The combined package begins to look like an instrument data systems (IDS; next bullet), but all the programming is done in the lab.

- An IDS with automated sample injection and data processing may be necessary. The LIMS would be expected to produce worklists of samples to be run, download them to the IDS, and automatically retrieve the completed test results and insert the values into the appropriate database locations. Each of the IDSs from various vendors and devices has its own method of making connections to a LIMS.

The point of all this is that the one thing that separates computing in the lab from the rest of the company, the thing that is getting increased attention, is likely the subject that the IT department (or outside consultants) may have the least understanding of. It isn't just "connect system A to the LIMS"; it's the nuances of how the equipment is operated that lab personnel take for granted that may not be expressed in the user requirements document.

Does your IT department or contractor understand the impact of regulatory requirements[3] and terms such as “validation,” “installation qualification,” “operational qualification,” and “performance qualification”? Do they understand that an inspector may review their documentation and examine their programming, with the authority to halt the project and insist upon corrective actions if they deem it necessary? This process begins at the point that a project is initiated, through the development of user requirements, testing, etc., until a validated system is delivered. Often you will hear people unfamiliar with the subject say “we’ll deal with that at the end of the project,” not realizing it is part of the project definition.

A final thought on in-house development: Something sparked the need for a LIMS in your laboratory. If you choose the in-house development option, what are you going to use to fill that need for the time it takes to develop a system and make it ready for production use?

The advantages of a commercial LIMS system

All of the points noted above, and more, have been addressed by vendors providing commercial products. They provide the product, support, and consulting, as well as continued development to ensure that what you are working with is the best product possible to fit your needs. They have the expertise to handle instrumentation and connections to other lab systems with a track record of success. Their engineering staff is focused on two things: product improvement and your success.

There are two words that often come up in these discussions: “configure” and “customization.” The ability to configure a product to fit your needs is a normal part of the business. It means that a product can be modified (e.g., adding tests, test groups, sample characteristics, etc.) with vendor-supplied tools to fit your needs, without compromising supportability and upgradability, and avoiding future obsolescence.

Customization can mean that changes are being made to the source code to add some functionality to the system. Doing so can compromise the integrity of the system and its ability to be upgraded; you may have changed something that “breaks” as a result of an upgrade, forcing you to re-implement those changes or live with an out-of-date system.

A reliable vendor will work with you to adapt the system to your needs. If a particular feature is needed, they will see how it can be realized without compromising the system's integrity.

In this piece, we’ve laid out two paths to obtaining a LIMS that satisfies your laboratory informatics needs. In the early days of LIMS, building it was a viable option, as the LIMS community was only beginning to figure out the functionality needed and reduce the cost of implementation. The community and market have since matured considerably in the past half-century as they’ve taken advantage of technology changes such as cloud computing. That and the stress on IT organizations now make “buy” a more attractive approach.

References

[1]https://www.astm.org/e1578-18.html

[2]https://www.astm.org/e1578-06.html

[3]Regulatory requirements are not just another bureaucratic set of hoops to jump through, they serve a very practical purpose: helping to ensure that the right tools are being used in lab work and that those tools are proven to work and meet their operational requirements (validation). Not only when they are first put into service, but throughout their service life.