Application Development as a Solution to Unmet Needs in Laboratory Workflows

Amrom Obstfeld AACC 2021, Session 35104

30-Sep-2021

Laboratory Information Systems in the Clinical Laboratory

If clinical instrumentation and analyzers can be considered the heart of a clinical laboratory then the Laboratory Information Systems (LIS) would be considered the central nervous system. In most settings the LIS is the central database and the command center of almost all workflows including core processes such as:

- Accessioning specimens into the laboratory
- Receiving orders
- Result verification
- Testing logic, autoverification, and reflexes
- Instrumentation interfacing
- Automation
- Asset tracking
- Quality Control

The Gap

- Have you experienced frustration as a result of limitations in your LIS?
- Has this led to the need for workaround or additional staffing to accommodate these limitations?

Given the critical role of the LIS in laboratory function it is no surprise that limitations and functionality gaps in these systems can cause frustration and, if significant enough, can lead to dangerous workarounds, increased staffing needs, and even can compromise patient safety. Most laboratories must confront these deficiencies at some point.

Several factors account for this situation, including:

1. LIS are designed for the most common workflows in the most common laboratories, however often laboratories must maintain unique services and processes for the local patient population, clinical practice, or regulatory environment.
2. Standard LIS design can be incompatible with more complex processes found in specialty laboratories.
3. In certain laboratory settings there is no appropriate software solution available.
4. The presence, and limitations, of interfaces, whether between instruments and software or between the laboratory system and the electronic medical record can lead to bottlenecks, "language barriers", and potentially the need for manual transcription.

The Choices

There are a limited number of options when facing shortcomings in the LIS. Broadly these can be categorized as:

- Buy - Purchase a 3rd party software or migrate to an entirely new LIS
- Build - Create a custom application to meet the laboratory's needs

Each option comes with its own strengths and weaknesses which are summarized in the following table:

	Strengths	Weaknesses
Buy	Off the shelf solution without the need for continued support by laboratory staff	• New system is just as likely to have gaps • Laboratory is unlikely to receive support on par with that from within the department
Build	Fit for purpose custom solution with potential to grow to evolve with the laboratory	• Coding not a common skillset in laboratory personnel • Laboratory "owns" the continued maintenance of the software to ensure it meets the service level required in a clinical setting

Lowering the Activation Energy for Custom Apps

The challenges involved in app development in the above table are real, but there are solutions! Although software development may at first appear to be firmly outside the scope of a laboratory, two concepts make this venture not only feasible but also a practical solution for "the Gap". These concepts are *open source software (OSS)* and *software engineering best practices*.

Open Source Software

Writing software at first blush may seem like an extremely daunting undertaking. Luckily, much of the work has already been done for us in the form of OSS. OSS is software developed by communities of programmers that anyone can freely view, copy, modify, or share. Often OSS is made available as "packages" which can be easily plugged in or knit into other software. Furthermore, popular OSS can be very stable, well supported, and frequently updated. The use of OSS by laboratories makes custom apps easier to write as it comes "baked in" with professional crowdsourced code. This is a topic that Daniel Holmes will speak about at length in his talk.

Software Engineering Practices

Even after the code is written the work is not done. Maintenance of the app can be just as critical as the code itself to ensure that it is fit for clinical purposes. Luckily a group of concepts, tools, and skill-sets collectively included in the framework of software engineering have rapidly evolved to meet this challenge. Stephan Kadauke will survey this field for us which can serve to harden our apps so they are ready to be used in the real world.