Embracing Change

Software platform technology and other time-saving techniques can boost innovation and reduce development time.

Frances Cohen • *Contributing Writer*

Photo courtesy of Source Scientific LLC, a BIT Company.

Ancient Greek philosopher Heraclitus is credited with saying that "the only constant is change." With regard to software development for *in-vitro* diagnostic (IVD) instrumentation, this concept certainly rings true. Advances in technology and development models and processes, desirable new features, and demand for simpler user interfaces and inter-connective instrument accessibility present an ever-changing landscape to the instrumentation software developer. In order for developers to remain cost-competitive while responding to the industry's changing demands, they increasingly must be creative, innovative and progressive in their methodologies.

Software's Role in IVD Instrumentation

Every basic function in a diagnostic instrument is controlled by software. All of the mechanical processes involved with pipettes, syringes and motors are controlled by programmers' code. On top of that basic foundation, there is the graphical user interface (GUI). In this more advanced layer, there are menus with selections used by the administrator of the instrument to control the underlying processes. The graphical menus serve to enrich the user experience and help to reduce operator error.

Basic functions of a common clinical analyzer might include pipetting (moving fluid from one place to another), mixing and adding reagents, and incubating the final mixture for a determined amount of time. Results may be ascertained through calculations measuring changes in the mixture's optical density; the outcome may determine the clinical diagnosis.

Software coding for basic functions controls the speed of the motor, the operation of the pumps, and the direction and precise movement of the loaders and stepper motors so as to minimize jostling of the fluids. Additionally there is control over the pipetting function and higher-level incubation, as well as the action of picking up a sample and dropping it off somewhere else inside the machine for more processing. When combined, all of these basic functions constitute a complicated instrument.

From a programming perspective, diagnostic instruments are very complex. The more tasks a machine performs, the more program code is required. And the more code there is, the greater the chance for software "bugs" to appear. Programmers often think they can write pages of code in a day; but after debugging and making that code functional, the amount of code written is reduced to only a few lines per day. Given the time (and expense) to identify the bugs and eradicate them, it is critical to reuse clean code that already has been validated and certified.

This is possible through software platform technology, an innovative approach to IVD instrumentation development where developers can use a layered system—building advanced features on top of a solid, tested foundation of codes dictating fundamental processes. By organizing a library of proven code sequences for reuse, one can effectively expedite a bug-free design of an IVD instrument.

The Need for Speed

Development time for a large-scale diagnostic system generally is measured in years; so, with the complexity of instruments steadily increasing, the need to establish ways with which to reduce that time period is crucial for IVD instrumentation developers.

This issue can be addressed best through the use of a multi-pronged approach:

1. Simplify complexities: Make object-oriented modules with clearly defined interfaces and functions of each object. Layer it, keeping each layer clear and simple in its function.
2. Create reusable objects—pre-tested/validated objects greatly reduce development time.
3. Create an efficient and effective development process.
4. Design in testability at each object and at each layer.

Since speed to market and competitive cost structures are primary drivers for outsourcing the development of a diagnostic instrument, more IVD OEMs are taking advantage of software platform technology. By exploiting the fact that many instruments share the basic building blocks with regard to software, a manufacturer can repurpose a foundation that is proven and validated and spend its development dollars on the functionality or feature that makes its instrument unique. With the availability of an extensive library of basic, certified components that are scalable and updateable, software platform technology has changed the way many IVD OEMs do business.

Create Less New Code

Because diagnostic machines share common basic functions, pre-written code can be reused to control the basic functions of multiple different instruments by slightly modifying it to adjust for the instruments' variable parameters. For instance, code can be adjusted to take into consideration motors' differing speeds, syringes' dissimilar sizes, and rotor arms' variable lengths.

An example of this software platform technology is a recent scenario where the same underlying software code was used on three different diagnostic instruments—a clinical chemistry analyzer (designed for high throughput of various chemistries with the same incubation schedules), an automated PCR/DNA (polymerase chain reaction/deoxyribonucleic acid) replicator and a flexible combo immuno-assay/clinical chem analyzer. All three machines use the basic functions of pipetting and incubating for chemical reactions. All have motors, syringes and moving mechanisms. The same basic code modules were adjusted for all three instruments to control all of their basic functions.

The software that makes this possible is based on reusable code components, written in object oriented C++, designed to integrate into an optimized real time operating system. Each separate module can be fully tested independently from the complete system, running scripts from a PC application. These pre-tested and validated modules contain the various software drivers and controls required to support the module's function. This reusable software can be easily extended as the technology or the instrument's requirements change.

The reusability of the programming code for basic functions allows developers to spend more time on programming advanced features—making a customer's instrument truly unique and technologically advanced. Considering the time and expense of developing a large-scale instrument, code reusability certainly is a key advantage.

Since speed to market and competitive cost structures are primary drivers for outsourcing the development of a diagnostic instrument, more IVD OEMs are taking advantage of software platform technology.

The Development Process

An efficient software development process also is a key component to creating an IVD instrument that is cost effective, operational and marketable.

Since medical devices are under U.S. Food and Drug Administration (FDA) regulatory control, the first step is to document the master plan. This plan assigns responsibility to individuals involved and identifies input and output of the project. Outputs generally are binaries built into the device as well as the development control documents of the design process. Perhaps most important to the developer is the fact that the requirements specification identifies precise requirements of the device according to the OEM customer. Often, the OEM customer has a basic idea of what it wants the specifications to be, but from a development perspective, it overlooks many aspects.

In addition to operational planning, developers also need to implement risk analysis. This is a critical phase for all medical devices and affects overall requirements of the software. For instance, if a motor crashes during pipetting, there needs to be a fail-safe built into the system that will prevent contamination, false readings or injury to the user. A developer must think through all of the possible adverse scenarios and plan for how to mitigate them. Once a solid basis of requirements has been established, the programming process begins. Agile development methods can help with the code development process.

Scrum programming, an agile approach to software development, relies on self-directed software development teams and dispenses with much advanced planning, task definition and management reporting. A "sprint" planning meeting is described in terms of the desired short-term outcome (i.e., a commitment to developing a defined set of features) instead of an all-encompassing, upfront set of completion criteria, task definitions, validation criteria, and exit criteria as would be provided in most methodologies. Because requirements usually change drastically from the start of the project to when it is complete, it is more efficient to break it down and work on it in short bursts or "sprints", which allow the customer to review the progress and make changes along the way. In the scrum process, the first task is to scrutinize the requirements, separating the basic features from the high level ones. A discussion follows regarding the priorities of those features and which ones should be developed first. Since most customers prefer to see the functional instrument first, that's usually where the work begins.

Once the first feature set is determined, a three- to four-week sprint commences. The purpose of the sprint is defined ahead of time and at the end, a "demo" of the results will take place followed by a retrospective that reviews what might have gone wrong and tackles such issues as if the team tried to do too much, if there was missing hardware or what could be improved for the next sprint. In addition, the team determines what requirements or priorities may have changed and what major problems need to be resolved. Finally, after determining a new set of objectives, another sprint is started, building on what already has been developed.

Experience over time has proven the scrum process to vastly enhance the efficiency of the software development process. As the developers grow accustomed to the cadence of the sprints, communications and teamwork improve and the resulting increase in output is tangible.

The ideal partnership between contractor and client is one with a high level of communication and involvement in the scrum process. When a customer is present at the end of a sprint and views the demo, he or she can see what has been accomplished and plays a part in setting the priorities for the next sprint.

Build for Test

During development, software programmers require tools to help build and test code. One construction tool might be a compiler or editor to help write the software code. A test tool would be an emulator, which allows the programmer to step through his codes and see what is happening within the instrument during the code sequence. There also are bug trackers so that ongoing issues can be documented, solved and catalogued.

Additional diagnostic tools are available that communicate to the instrument on a software level to test functionality. Such tools can be used during development as well for service and maintenance after the instrument has been developed and is in operation. One such application is called FingerTip, which was designed to be used with a PC, Apple iPad, or most recently, an Android-based device. The system queries the instrument for functions available and parameters to control, such as motor speed, electrical current, degrees of rotor rotation, etc. It can retrieve that information and allow the user to make the function calls to any object. This allows for comprehensive testing and debugging. For instance, if the developer wants to move the motor and rotate the rotor for a week to make sure home position is not lost, he can set up a script to do that and let it run.

This built-in interface to the device also can be used to allow customers to validate their chemistries long before the intricacies of the actual device application are complete. Using the built-in libraries, a simple application can run the functions necessary for their tests. This allows chemistries to be tested in parallel with device development.

Service technicians who need to interact with the instrument, running specific tests for calibration or other maintenance, also use the technology, which is accompanied by real-time service manuals, service updates, and videos providing more detailed information about how to make adjustments.

The Future of Instrumentation Software

More options, faster to market, and intuitive, easy-to-use instruments top the wish lists of today's IVD OEM companies. Perhaps more importantly, as consumer technology advances with time, OEM's expectations for "smarter" instrumentation increase. Thus, today's inter-connective embedded technology in the hands of the consumer also drives OEM IVD customers to demand a higher level of inter-connective communication with their sophisticated new instruments. Such communication is highly desirable because this interconnectivity can translate into remote set-up, repair, maintenance and calibration by service technicians and more productivity and less down time for the end user.

With FDA regulatory restrictions, however, such features represent added levels of complexity, such as security and safety restrictions. The wrong commands or incorrect software updates adversely could influence test results, rendering them inaccurate. The FDA-approved conservative route may take a bit more time than that taken by consumer electronics, but the demand certainly will drive the trend in the direction of remote accessibility. Because "easier-to-use" is always more difficult to develop, it is more crucial than ever before that developers acquire programming shortcuts and more efficient methodologies pursuant to the basic tasks of instrumentation development so they can concentrate on the upper-level complexities.

Software platform technology, reusable code, more efficient development processes and design for testability will help IVD software developers to invest their resources in responding to the complex demands of today's sophisticated IVD OEM; but as advances in technology continue to drive change in the demands of customers, so must IVD software developers continue to embrace that change and pursue innovation in their own fields. ❖

Frances Cohen serves as software manager of Source Scientific LLC, a BIT Company, located in Irvine, Calif. She can be reached at fcohen@SourceScientific.com