CASE STUDY

Background: Sample Preparation for Next Generation DNA Sequencing

By Kevin Bodner and Chris Lynch, R&D Engineering, ThermoFisher

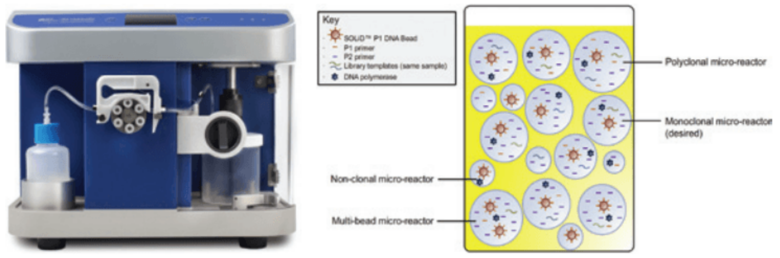

Figure 1: The SOLiD™ EZ Bead™ Emulsifier and a Finished Emulsion of Aqueous Droplets in Oil

INDUSTRY

Medical Technology

OBJECTIVE

Our challenge was to produce a complete software solution to operate a trio of products that prepares samples for next-generation DNA sequencing. The solution needed to include commercial shipping software along with software tools for R&D feasibility, manufacturing and field service. We had to deliver robust software within a compressed schedule of nine months – without an established software development infrastructure.

APPROACH

Leverage the commercial software development tools and expertise from JKI's NI Certified LabVIEW Architects to create a flexible, extensible suite of software products in LabVIEW. LabVIEW and JKI's Rapid Application Framework for LabVIEW enabled the seamless development of a fully configurable software solution that satisfied ever evolving hardware and user interface needs.

THE SYSTEM

NI LabVIEW 2009 SP1

JKI VI Package Manager Enterprise

JKI Rapid Application Framework for LabVIEW

NI PCMCIA-232/4 Serial Interface

NI USB-6008 and NI USB-6009

Multifunction DAQ

NI USB-6501 DIO

NI USB-9162 Module Carrier NI 9211 and NI 9213 Thermocouple Input Modules

NI 9215 AI Module

NI 9472 DO Module

Background: Sample Preparation for Next Generation DNA Sequencing

There are two basic steps in DNA sequencing: (1) preparing the DNA sample for sequencing, and (2) using a sequencer to detect the DNA sequence of the sample. Typically, sample preparation is a very manual and time consuming process. The Life Technologies SOLiD™ EZ Bead™ System for SOLiD DNA sequencing consists of three bench-top instruments: an Emulsifier, an Amplifier, and an Enricher, which automate the otherwise laborious task of preparing a sample for DNA sequencing.

The SOLiD EZ Bead Sample Preparation System

The goal of sample preparation using the SOLiD EZ Bead System for SOLiD sequencing is to prepare the DNA-templated beads that will be deposited onto the glass slide. There are three main steps to doing this, and each is performed by one of the three instruments in the SOLiD EZ Bead System.

Step One – Emulsification: The Emulsifier instrument (Figure 1) consists of a mixer and a peristaltic pump. An emulsion is an oil-aqueous-bead mixture and contains uniformly-sized aqueous droplets, with each droplet containing one bead. The Emulsifier creates the emulsion by metering the rate of aqueous addition and agitating the oil-aqueous-bead mix for a specified time.

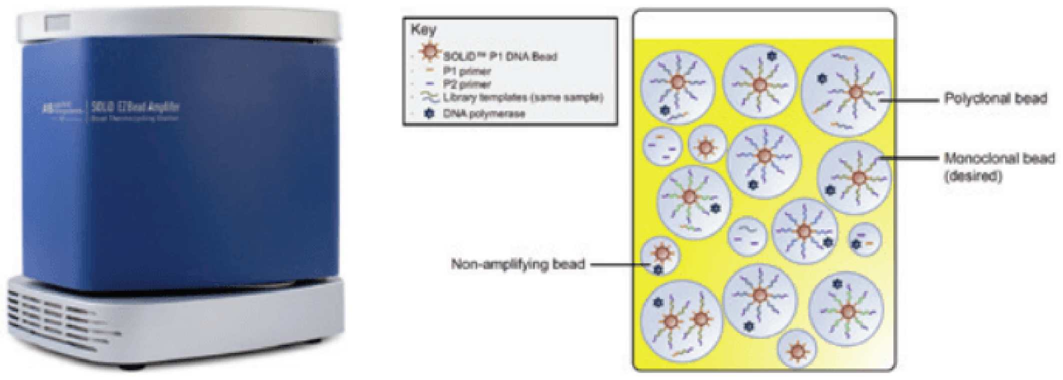

Figure 2: The SOLiD™ EZ Bead™ Amplifier and Post-PCR Emulsion of Amplified DNA Templates

Step Two – Amplification: The Amplifier instrument (Figure 2) consists of two Peltier-based, vertically oriented thermal blocks that are sandwiched around a bag of emulsion. Polymerase chain reaction (PCR) chemistry creates many copies of DNA fragments, increasing exponentially, by repeated heating and cooling cycles. The Amplifier performs PCR on the emulsion by ensuring temperatures are ramped rapidly and accurately. The number of cycles used for PCR, along with the temperatures and hold times for each cycle, are easily edited. LabVIEW, running on a netbook computer, controls the Amplifier.

Step Three – Enrichment: The Enricher instrument (Figure 3) consists of a syringe pump, 12-way valves, solenoid valves, vibrator motors, thermomixer, conductivity sensor, door locks and switches, and LED indicators. Enrichment is the extraction of the desirable beads, those that underwent successful amplification, from the non-desirable beads, those that did not amplify. The Enricher separates the amplified emulsion into aqueous and oil phases. The aqueous phase is sent through a filter that captures the successfully amplified beads. Following several washes, the enriched beads are removed from the filter and delivered to a collection vessel. LabVIEW runs on a laptop computer to control the Enricher.

LabVIEW has played an important role during the last ten years at Life Technologies, used to develop software during early instrument prototyping and then for manufacturing test and service. The SOLiD EZ Bead software was the first of its kind, as LabVIEW was used to develop the application used by external customers. The use of LabVIEW and VIPM for both the prototype and customer software enabled code reuse while eliminating the need to rewrite software in another programming language. This code reuse allowed us to shorten our development cycles and meet our aggressive schedule

There were many hardware configurations during the development of the instrument that needed to be supported by the software. Initial breadboards and prototypes made use of off-the-shelf hardware from

NI. Direct serial communication to several components was done through a NI PCMCIA-232/4. Digital inputs and outputs were provided by NI USB-6008 Multifunction IO devices. Temperature monitoring was done using NI 9211 and NI 9213 Thermocouple Input Modules housed in NI USB-9162 Module Carriers. For the shipping instruments, a custom electronics board and embedded firmware were used, with a single USB connection. The numerous hardware permutations presented a constant challenge for the software.

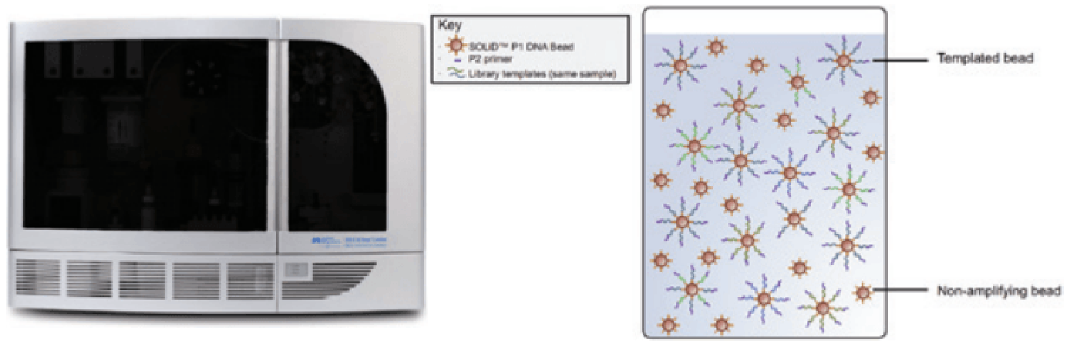

Figure 3: The SOLiD™ EZ Bead™ Enricher and Enriched Template Beads
Development of Instrument Control Software for SOLiD™ EZ Bead™ System

R&D scientists and engineers demanded that the software worked continuously on a variety of different hardware configurations, and remained online 24x7 over time as the hardware evolved. We selected JKI's Rapid Application Framework for LabVIEW (RAFL) to meet this challenge. It contains a rich set of built-in features to quickly develop professional and robust commercial software. RAFL has built-in support for parallel processes, scripting, multiple user screens, and asynchronous communication and control.

RAFL utilizes LabVIEW object-oriented programming to build extensible classes of subsystems. Hardware classes were defined for generic hardware components, such as a thermal mixer. Then, as the thermal mixer hardware changed, the methods of the generic thermal mixer could be dynamically dispatched to the method for the new thermal mixer (Figure 4).

The development of all three modules needed to be completed on a compressed schedule with limited resources. As a result, we resorted to a pragmatic approach to software development. We employed best in class software engineering practices and tools as time allowed, based on informal risk assessments.

We used several Agile practices, scrums to facilitate communication and short sprints to prioritize features and break the work down into smaller tasks. Unified Modeling Language (UML) sketches were

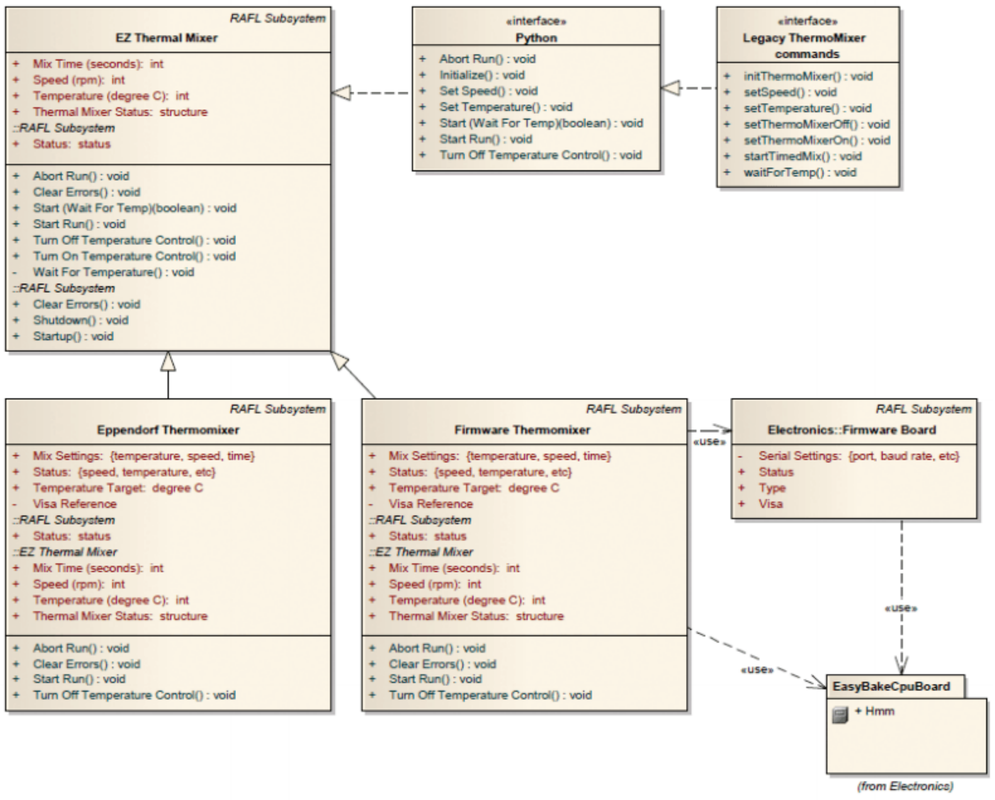
RAFL Subsystem
EZ Thermal Mixer
+ Mix Time (seconds): int
+ Speed (rpm): int
+ Temperature (degree C): int
+ Thermal Mixer Status: structure
::RAFL Subsystem
+ Status: status
+ Abort Run() : void
+ Clear Errors() : void
+ Start (Wait For Temp)(boolean) : void
+ Start Run() : void
+ Turn Off Temperature Control() : void
+ Turn On Temperature Control() : void
+ Wait For Temperature() : void
::RAFL Subsystem
+ Clear Errors() : void
+ Shutdown() : void
+ Startup() : void
«interface»
Python
+ Abort Run() : void
+ Initialize() : void
+ Set Speed() : void
+ Set Temperature() : void
+ Start (Wait For Temp)(boolean) : void
+ Start Run() : void
+ Turn Off Temperature Control() : void
«interface»
Legacy ThermoMixer commands
+ initThermoMixer() : void
+ setSpeed() : void
+ setTemperature() : void
+ setThermoMixerOff() : void
+ setThermoMixerOn() : void
+ startTimedMix() : void
+ waitForTemp() : void
RAFL Subsystem
Eppendorf Thermomixer
+ Mix Settings: {temperature, speed, time}
+ Status: {speed, temperature, etc}
+ Temperature Target: degree C
- Visa Reference
::RAFL Subsystem
+ Status: status
::EZ Thermal Mixer
+ Mix Time (seconds): int
+ Speed (rpm): int
+ Temperature (degree C): int
+ Thermal Mixer Status: structure
+ Abort Run() : void
+ Clear Errors() : void
+ Start Run() : void
+ Turn Off Temperature Control() : void
RAFL Subsystem
Firmware Thermomixer
+ Mix Settings: {temperature, speed, time}
+ Status: {speed, temperature, etc}
+ Temperature Target: degree C
- Visa Reference
::RAFL Subsystem
+ Status: status
::EZ Thermal Mixer
+ Mix Time (seconds): int
+ Speed (rpm): int
+ Temperature (degree C): int
+ Thermal Mixer Status: structure
+ Abort Run() : void
+ Clear Errors() : void
+ Start Run() : void
+ Turn Off Temperature Control() : void
«use»
RAFL Subsystem
Electronics::Firmware Board
- Serial Settings: {port, baud rate, etc}
+ Status
+ Type
+ Visa
«use»
«use»
EasyBakeCpuBoard
+ Hmm
(from Electronics)

Figure 4: UML Sketch of a Generic RAFL Subsystem and Specific Hardware Implementations

used during design, while JKI engineers created an invaluable automated software unit test and build tool. Code reuse was enabled by using JKI's VI Package Manager (VIPM) to create and manage packages of software libraries. The software team tracked defects during development, while a single software test resource created test plans and executed test cases.

Research and Development Software

R&D software needs to expose all available functions of the hardware to the user. The user requires this level of control to determine the optimal operation of the hardware. RAFL's built-in system view (Figure 5) provided ready-access to low level manual control and configuration screens for each subsystem. This mechanism enabled users to manually control each subsystem individually and to troubleshoot any problems with connections, communications, or configurations.

After the optimal operation of subsystems was determined, they were combined to create chemistry protocols. Scripting was key to enabling the efficient development of

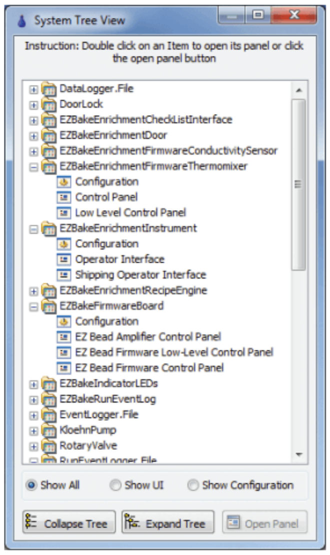
System Tree View
Instruction: Double click on an Item to open its panel or click the open panel button
DataLogger.File
DoorLock
EZBakeEnrichmentCheckListInterface
EZBakeEnrichmentDoor
EZBakeEnrichmentFirmwareConductivitySensor
EZBakeEnrichmentFirmwareThermomixer
Configuration
Control Panel
Low Level Control Panel
EZBakeEnrichmentInstrument
Configuration
Operator Interface
Shipping Operator Interface
EZBakeEnrichmentRecipeEngine
EZBakeFirmwareBoard
Configuration
EZ Bead Amplifier Control Panel
EZ Bead Firmware Low-Level Control Panel
EZ Bead Firmware Control Panel
EZBakeIndicatorLEDs
EZBakeRunEventLog
EventLogger.File
KloehnPump
RotaryValve
Show All
Show UI
Show Configuration
Collapse Tree
Expand Tree
Open Panel

Figure 5: The RAFL System View Allows Access to Low Level Screens

chemistry protocols. RAFL has built-in support for both Python™ and Standard Commands for Programmable Instruments (SCPI) through Transmission Control Protocol (TCP) connections. Python, an established programming language, was chosen for its simplicity while enabling conditional statements, looping, and passing of variable parameters. Scientists developed custom Python scripts for all protocols. Python scripts send commands to a server written in LabVIEW, listening on a TCP port, which in turn directs commands to the appropriate subsystem.

Once the basic scripts were finished, the R&D Recipe Editor Screen (Figure 6) could be used to modify the parameters and run the script. In this case, LabVIEW loaded and edited Python scripts and sent them to Python for execution.

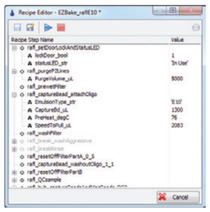
Figure 6

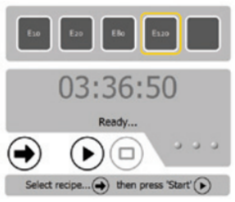

Figure 7

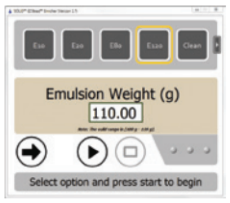

Figure 8

Customer Software

A simplified user screen (Figure 7) and workflow were needed for the shipping customer software. Also, the look and feel of the screens needed to be consistent (Figure 8) across all three modules. Again, flexibility was a key contributor to success. We knew final hardware and firmware was not going to be ready in time for release to early access customers, but RAFL helped us work around this issue. The early access customers used the final customer screen, without the final instrument configuration. This strategy allowed us to collect important feedback on both the software and instrument. Instrument configurations are defined in Extensible Markup Language (XML) files and the required XML

“

LabVIEW and JKI's Rapid Application Framework for LabVIEW enabled the seamless development of a fully configurable software solution that satisfied ever evolving hardware and user interface needs.

Kevin Bodner,
R&D Engineering, ThermoFisher

“

The use of LabVIEW and VIPM for both the prototype and customer software enabled code reuse while eliminating the need to rewrite software in another programming language. This code reuse allowed us to shorten our development cycles and meet our aggressive schedule.

Chris Lynch,
R&D Engineering, ThermoFisher

file is loaded at startup. As a result, the same customer screen can be used to control any combination of hardware components.

Manufacturing Test and Service Software

The manual control screens for all of the subsystems enabled manufacturing test and servicing of the instrument. When a need surfaced to test the Enricher fluidic system as a whole, a new screen (Figure 9) was developed. As in R&D, custom Python scripts were created for manufacturing tests.

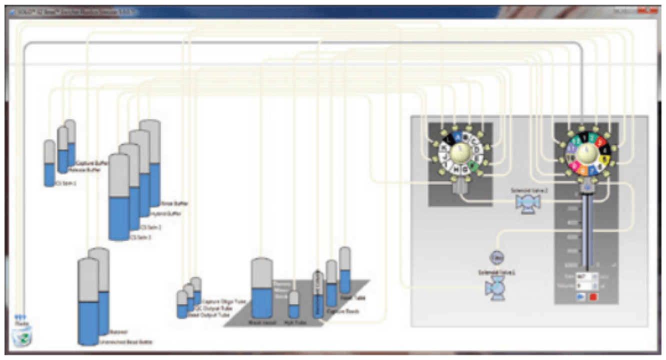

Figure 9: Fluidics Test Screen

Benefits Realized

With the release of the EZ Bead system, SOLiD customers can realize an 80% reduction in sample preparation time, with less than 1 hour of hands-on time now required. Over 200 SOLiD EZ Bead Systems have been shipped. Combined, customers are performing over 1000 runs per month. Six months after product launch, no software patches or upgrades have been required.

LabVIEW and JKI's RAFL architecture enabled a small software team to deliver commercial software within a nine month period. This achievement resulted in a reduced software development expenditure that will result in a shorter payback period and an increased return on investment. Further, the flexibility and scripting capabilities of the software allows continued protocol development, without support from R&D software engineers.

“

LabVIEW and JKI's Rapid Application Framework for LabVIEW enabled the seamless development of a fully configurable software solution that satisfied ever evolving hardware and user interface needs.

Kevin Bodner

R&D Engineering, ThermoFisher

JKI

3687 Mount Diablo Blvd, Suite 208
Lafayette, CA 94549
888.891.7821
jki.net
info@jki.net

jki.net/blog
facebook.com/JKISoftware
twitter.com/JKISoftware