



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Designing And Implementing LabVIEW Solutions For Re-Use

M. Flegel, G. Larkin, L. Lagin, B. Demaret

October 2, 2013

ICALEPCS 2013
San Francisco, CA, United States
October 6, 2013 through October 11, 2013

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

DESIGNING AND IMPLEMENTING LABVIEW SOLUTIONS FOR REUSE*

M Flegel, G Larkin, L Lugin, B Demaret, LLNL, Livermore, CA 94550, USA

Abstract

Many machines have a lot in common – they drive motors, take pictures, generate signals, toggle switches, and observe and measure effects. In a research environment that creates new machines and expects them to perform for a production assembly line, it is important to meet both schedule and quality. NIF has developed a LabVIEW layered architecture of Support, general Frameworks, Controllers, Devices, and User Interface Frameworks. This architecture provides a tested and qualified framework of software that allows us to focus on developing and testing the external interfaces (hardware and user) of each machine.

THE NATIONAL IGNITION FACILITY

The NIF is the largest and most energetic laser system in the world, capable of creating temperatures and pressures normally constrained to stars, giant planets, and nuclear weapons with a goal of achieving controlled inertial confinement fusion in a laboratory setting. In order to accomplish this, the NIF uses a significant number of near perfect optics to deliver the laser energy, and a near perfect capsule used to contain the fuel.

AUXILIARY PRODUCTION FACILITIES

Optics Mitigation Facility (OMF)

In order to achieve “near perfection” with the optics, the NIF built the “Optics Mitigation Facility” in 2010. This system guides an operator through the inspection flaws, examines and characterizes them with a Fetura microscopes and Basler cameras, and decides whether they can be mitigated. Mitigations as small as 360 microns are applied using a real-time motion chassis with a real-time laser light delivery system.

The OMF was implemented in LabVIEW and was the focus of the highly respected case study – “Using LabVIEW in a Critical Laser Application for the National Ignition Facility at Lawrence Livermore National Laboratory”[1] – co-written between LLNL and National Instruments (NI). From the success of the OMF, the NIF undertook to commission four more optics processing systems, three target processing systems, and one line replaceable unit (LRU) transporter.

Optics Processing Systems

Grated Debris Shield (GDS) Etch drives an optic across a set of meniscus processing heads that chemically treats and rinses a photoresist coated optic to develop and etch a grating pattern into the glass substrate of the optic.

*This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. #LLNL-ABS-632634

Photoresist (PR) & SolGel Meniscus Coaters, similar to GDS Etch, apply a thin layer of a chemical to one side of an optic. Given the different fluid behaviours the optic clearance relative to the meniscus process head is much smaller (0.5mm vs. 2mm) than GDS Etch.

Flaw Inspection and Characterization System (FICS) scans an optic for flaws (IMS-LS and FADLiB) and examine the flaws in detail (PSDI). The metrology information is used to determine how to mitigate the flaw with a CO₂ laser drill on OMF, with a diamond drill on a Crystal Mitigation System, or chemically removing the flaw in the coating (with FICS’ Flaw Removal Tool (FLRT)).

Target Processing Systems

CFTA Cleaning chemically cleans the surface of a Capsule Fill Tube Assembly (CFTA) using a fine spray nozzle[2].

CFTA Mapping takes over 350 confocal images of a capsule surface to characterize the capsule’s surface features[3][4][5].

CFTA Leaktest monitors temperature, pressure, and leak rate sensors over time for a capsule under test to automatically determine the integrity of the capsule.

Transporter Systems

ARC PV Transport & Handling transports, installs and removes Advanced Radiographic Capability (ARC) LRUs weighing around one ton in the NIF’s Parabola Vessel (PV) with clearances as small as 3 mm.

FRAMEWORK

Motivation

The OMF set the stage for the viability of advanced application development in LabVIEW. It took advantage of:

- Prebuilt drivers for hardware and instruments;
- Highly customizable drag-and-drop user interfaces;
- Easy, rapid prototyping for testing and demonstrating new features and concepts;
- Built-in vision and analysis routines;
- Easy manipulation for large data sets with built-in array functionality.

Software engineering best practices including requirements analysis, design, test, and change management were used; and the system was delivered in 15 months, roughly one-third of the estimate to develop using Java or C++.

Observing that OMF’s success was based on re-using LabVIEW’s built-in routines and applying software engineering best practices, a reusable layered architecture of additional abstractions and components was designed

and used to build the next eight systems, with LabVIEW. This approach was introduced at NIWeek 2011[6] (a major international conference), applied to the eight

systems described above, and summarized in the configurations and reuse metrics listed in Table 1.

Table 1: System Configurations and Reuse Factors

System Release		Common Release	System		Common		Total		Reuse	
			Classes	VI's	Classes	VI's	Classes	VI's	Classes	VI's
Common					186	1294				
CFTA Mapping	1.1.0	1.0.7 RC002	45	448	86	650	131	1098	66%	59%
CFTA Cleaning	1.1.0	1.0.3 RC002	17	101	66	510	83	611	80%	83%
GDS Etch	2.0.0	1.0.6 RC004	60	628	83	703	143	1331	58%	53%
FICS	2.0.2	1.0.4 RC003	69	378	104	788	173	1166	60%	68%
CFTA Leaktest	1.1.0	1.0.6 RC003	69	378	104	788	173	1166	60%	68%
PR Coater	2.0.0	1.0.6 RC004	25	216	58	436	83	652	70%	67%
SolGel Coater	2.0.0	1.0.6 RC004	57	322	104	788	161	1110	65%	71%
TH ARC	1.0.0	1.0.7 RC002	41	225	75	658	116	883	65%	75%
Average			47	337	85	665	133	1002	64%	66%

There are three major components to the architecture:

- **Layering** fosters efficient reuse of code[7];
- **Abstractions** codify regularly occurring patterns such as actors, hardware abstractions, recipes, user interfaces, applications (that pull everything together), etc.;
- **Components** codify regularly occurring capabilities such as configuration, logging, mail, communication, database, device models (e.g. actuators, motors, regulators, cameras, sensors), etc.

Layering

Layers provide a narrow and well-defined interface to layers below it [7], with each layer defining a progressively more abstract machine and permits retargeting[8]. There are seven layers to the architecture (see Figure 1) consisting of:

1. **Support** – basic classes and utilities;
2. **Frameworks** – basic abstractions and components;
3. **Framework Services** – generalized services;
4. **Controllers** – interfaces to external systems;
5. **Devices** – device behaviour and commonly used devices;
6. **Application Support** – glue that holds the entire system together; and

7. Systems – the unique requirements of each application.

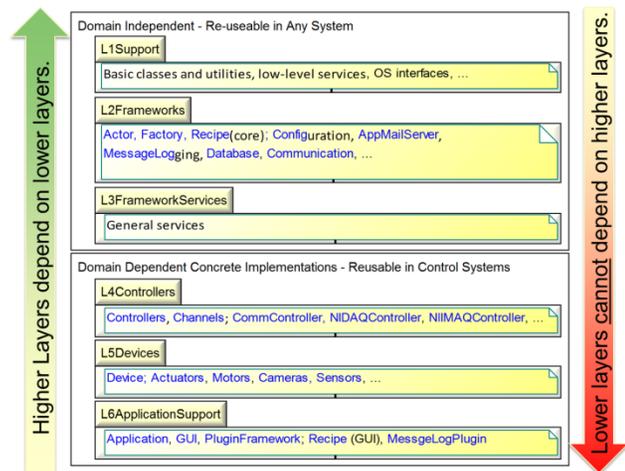


Figure 1: Layering

Abstractions

Abstractions describe recurring themes, but lack the concrete details to stand on their own. There are three key abstractions at the centre of the Framework.

Actors represent resources that have state. The resource can be a single device or a supervised collection of devices; it can change value and/or state, and publish this to interested subscribers. Figure 2 is a high level class diagram of the actor design.

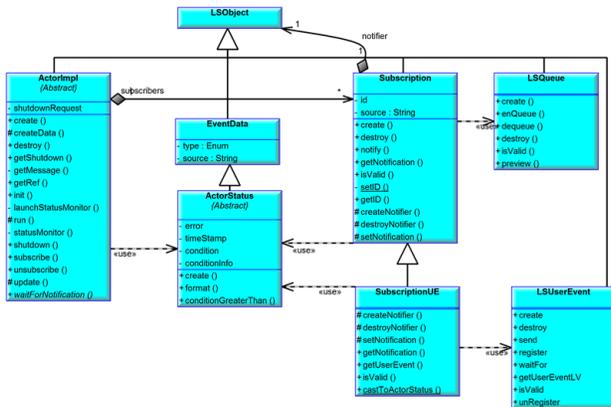


Figure 2: Actor Class Diagram

Hardware Abstractions insulate device implementations from the actual hardware: the Device models the behaviour of the hardware; the Controller implements the concrete interface to the hardware; and the Channel adapts the Device to the Controller. The same Device implementation can be used with different Controllers (e.g., Aerotech, Newport, Wago, etc.). Figure 3 is a high level class diagram of the hardware abstraction design.

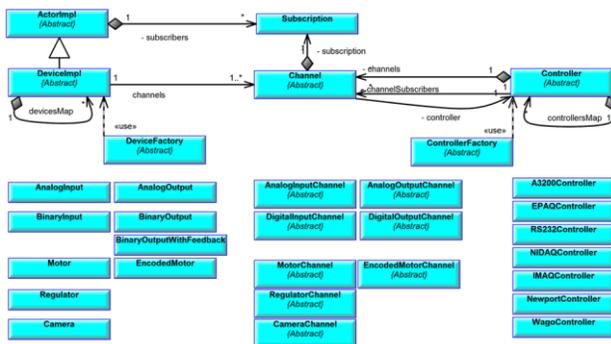


Figure 3: Hardware Abstraction Class Diagram

Application and User Interfaces encapsulate the rules for building the program and allowing the program to interact with operators, testers, and developers. With a graphical user interface (GUI) Framework, user interfaces implemented to provide Device control (see above) can be reused within the same application as well as in other applications. Figure 4 is a high level class diagram of the application and GUI design.

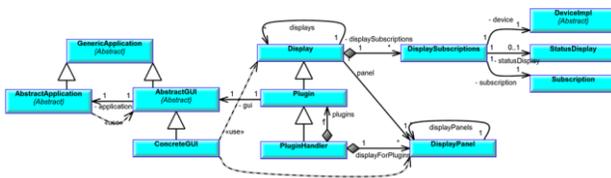


Figure 4: Application and GUI Class Diagram

EXPERIENCES

What Comments May Be Encountered Along the Way

LabVIEW applications are ‘sub-standard’ and are unstable for production. LabVIEW is a programming language. Good software engineering practices, a good design methodology, and trained engineers are what make quality systems that meet DOE Order 414.1D[9].

Why is it taking so long? Early systems are on the hook for creating the Framework. Once the base Framework is in place, migration to a more agile development process allows the delivery of manual control of the machine, followed incrementally by more complex solutions to meet needs and expectations.

You implemented what I asked for, but that’s not what I want! Customers often don’t know what they want until they see what they are getting. Working with customers to develop the user interfaces helps to understand the requirements and expectations, and helps the customers buy into the system being built.

Individuals had their own software ‘toolbox’. Standalone developers tend to have their own collections of software tools that they upgrade and fix each time they are reused, but the changes are rarely applied to previous systems. Usually only the developer understands their own tools. A common shared and configured toolbox allows these tools to be tracked and understood amongst many developers, and many systems.

How To Do This

“good engineering practices”. A team trained in software engineering skills – project planning, requirements analysis, object oriented design and programming, code reviews, independent test, configuration management (*Jira* change management[10][11] and *AccuRev* source code control[12]) – is essential. These skills are used to create plans, estimates, and schedules that are tracked and communicated with management and customers. The focus of the development effort should be on the systems, with reuse in mind. Items identified for reuse are refactored into the Common Framework when needed and/or mature for reuse.

Track How It’s Going

Collecting metrics is important to measure the effects. Table 2 illustrates our metrics order of the earliest (CFTA Mapping) to most recent (TH ARC) systems, and correlates the effort (in days) with the artefacts of the systems – number of classes and control points (devices and controllers) – required to implement each system.

Table 2: Development Metrics

		Effort Total	Effort per Class	Effort per Control Point	Control Points
1	CFTA Mapping	585	4.1	45.0	13
2	CFTA Cleaning	199	2.4	8.3	24
3	GDS Etch	486	3.0	5.3	91
4	FICS2	321	2.5	14.6	22
5	CFTA Leaktest	98	1.2	9.8	10
6	PR Coater	109	0.6	0.9	119
7	SolGel Coater	109	0.6	0.9	123
8	TH ARC	249	2.1	2.1	120

As the systems are built, each contributes something to the Framework. Earlier systems are taxed with more significant contributions than later systems. At first, customers were uneasy as they felt their systems were being unfairly taxed and taking significantly longer than they expected.

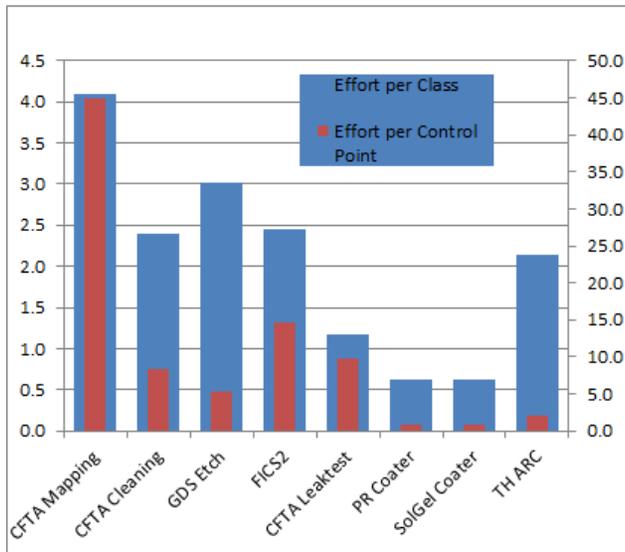


Figure 5: Development Trends

However, they were pleased with the quality – “These are some of the most stable systems we have seen.” Customers of later systems were also pleased that “the time required to generate a product of the same complexity [and quality] was significantly reduced”. National Instruments is taking a keen interest in this process and the results.

WHAT NEXT

The metrics indicate that the approach is successful. The next steps are to:

Continually Improve – more agile development, subdivide the Framework into packages, encourage developers to enhance their skills with training and certification.

Rapid Prototyping – develop a process that allows prototypes to be deployed that are needed for proof of concepts for machines and machine processes.

REFERENCES

- [1] G Larkin, “Using LabVIEW in a Critical Laser Application for the National Ignition Facility at Lawrence Livermore National Laboratory”, National Instruments Case Studies, c2010.
- [2] SH Baxamusa, “A Solvent Cleaning Process for the Outer Surfaces of Plastic ICF Capsules”, 20th Target Fabrication Meeting, 2012.
- [3] NA Antipa, “Automated ICF Capsule Characterization Using Confocal Surface Profilometry”, Fusion Science and Technology 63 (2), 151-159, March 2013.
- [4] LM Kegelmeyer, “3D Surface Mapping of Capsule Fill-Tube Assemblies used in Laser-Driven Fusion Targets”, European Society for Precision Engineering and Nanotechnology, February 2011.
- [5] NA Antipa, “The Capsule-Fill-Tube-Assembly Mapping System”, 20th Target Fabrication Meeting, 2012.
- [6] M Flegel and G Larkin, “Mitigation of Optic Flaws for NIF Laser”, NIWeek 2011 – Big Physics Symposium, August 2011.
- [7] Phillippe Kruchten, Christopher J. Thompson, “An object-oriented, distributed architecture for large-scale Ada systems”, TRI-Ada '94 Proceedings of the conference on TRI-Ada '94, 262-271, 1994.
- [8] John P. Woodruff, “The National Ignition Facility Integrated Computer Control System”, Presented at Stanford Linear Accelerator Centre, April 2000.
- [9] US Department of Energy, “DOE 414.1D, Quality Assurance”, April 2011.
- [10] Atlassian, “Jira”.
- [11] J Fisher, "Utilizing Atlassian JIRA for Large-Scale Software Development Management", 14th International Conference on Accelerator & Large Experimental Physics Control Systems (ICALEPCS), October 2013.
- [12] AccuRev, “AccuRev”.