ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THCPL06

SUSTAINING THE NATIONAL IGNITION FACILITY (NIF) INTEGRATED COMPUTER CONTROL SYSTEM (ICCS) OVER ITS THIRTY YEAR LIFESPAN*

B. Fishler, G. Brunton, B. Reed, M. Fedorov, P. Ludwigsen, C. Estes, M. Flegel, M. Paul, A. Barnes, E. Stout, Y. Abed, V. Miller-Kamm, R. Wilson
Lawrence Livermore National Laboratory, Livermore, CA 94550 USA

Abstract

The National Ignition Facility (NIF) is the world's largest and most energetic laser experimental facility with 192 beams capable of delivering 1.8 megajoules of 500-terawatt ultraviolet laser energy to a target. Officially commissioned as an operational facility on March 21, 2009, NIF is expected to conduct research experiments thru 2039. The 30-year lifespan of the control system presents several challenges in meeting reliability, availability, and maintainability (RAM) expectations. As NIF continues to expand on its experimental capabilities, the control system's software base of 3.5 million lines of code grows with much of the legacy software still in operational use. Supporting this software is further complicated by technology life cycles and turnover of senior experienced staff. This talk will present lessons learned and new initiatives related to technology refreshes, risk mitigation, and changes to our software development and test methodology to ensure high control system availability for supporting experiments throughout NIF's lifetime.

INTRODUCTION

The National Ignition Facility (NIF) [1] provides a scientific center for the study of inertial confinement fusion (ICF) and matter at extreme energy densities and pressures [2]. Each NIF experiment, or shot cycle, is managed by the Integrated Computer Control System (ICCS) [3], which uses a scalable software architecture running code on more than 2000 front end processors, embedded controllers and supervisory servers. The NIF control system operates laser and industrial controls hardware containing 66,000 control points (e.g. motors, calorimeters, etc) to ensure that all NIF's 192 laser pulses arrive at a target within 30 picoseconds of each other, are aligned to a pointing accuracy of less than 50 microns, and orchestrate a host of diagnostic equipment collecting experimental data in a few billionths of a second. Every NIF shot cycle [4] consists of approximately 1.6 million sequenced control point operations, such as beam path alignment, pulse shaping and diagnostic configuration and each cycle is typically conducted within 4-8 hours depending on the experiment complexity.

NIF was commissioned as an operational facility in the Spring of 2009 and is expected to remain operational until at least 2039. The control system has grown to over 3.5M lines of code. Much of it has been running on hardware that has been used for 1000s of laser shots since operations commenced. While being a fully operational experimental system, there are ongoing requests to add capabilities as requested by the NIF user community, but also from external influences that are forcing involuntary change to ensure NIF's continued operation. The object-oriented, data driven, distributed nature of ICCS helps in managing the high volume of change, but designing for change is alone insufficient to ensure the continued high reliability, availability, and maintainability (RAM) requirements while NIF maintains its goal of 400 shots per year. This paper documents the various influences that are driving change within our control system, and how we have adapted to ensure the control system's reliability and sustainability over its 30+ year lifespan.

DESIGN AND DEVELOPMENT INFLUENCES

ICCS continues to provide new capabilities to support programs internal and external to LLNL. These capabilities are critical in understanding laser and target experimental conditions, but also to support enhanced operational efficiency. With the growth of these capabilities, the number of experiment and operational scenario permutations grew exponentially.

The Department of Energy's (DOE) budget for NIF has remained flat for the past several years. Increases in employee salaries, inflation, etc. results in a net decrease in NIF's purchasing ability under a flat budget. This is forcing us to look at internal process efficiencies in order to maintain the same level of service with fewer resources.

Cybersecurity has been a priority for DOE as well as internally in the lab. Several new regulations and policies that have rolled out have required changes within the control system, primarily in the underlying frameworks. In the early days of NIF, the primary mitigation for computer security related hazards was complete isolation of the control system from external systems. However, this is no longer sufficient and to further enhance our cybersecurity position, several enhancements have been implemented.

Coding for ICCS began in 1997 [5] with much of the same code base still in active use on NIF. Control system

* LLNL-ABS-727374

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THCPL06

hardware has been in use since the NIF Early Light commissioning in 2003 of NIF's first four beamlines [6]. The technology used to develop and execute the associated software and hardware systems is reaching its end-of-life either due to product failure or the end of manufacturer support. This is forcing change to allow continued upgrades to newer versions of our current operating systems, compilers, etc., or migration to a new platform.

Retirements and attrition at LLNL have resulted in the departure of many senior experienced staff members since development of the control system commenced. This pattern is expected to continue for the duration of the project. With a thriving technology sector, there are constant challenges in retaining staff, especially with LLNL's close proximity to silicon valley. With the departure of experienced developers, the domain knowledge and intimate familiarity with the software that they designed and built is a significant loss. Our development processes needed to accommodate the loss of this knowledge. An additional issue is how to accelerate new employees to a performing level on a significantly complex system such as ICCS.

In FY16, NIF achieved a shot rate of 400 target shots per year which was again achieved in FY17. Efficiency enhancements in the control system were a significant contributing factor in achieving this higher and sustainable shot rate [7]. NIF operations has allotted quarterly facility maintenance windows ranging from a few days to three weeks to allow for major reconfiguration activities. It is within these windows that ICCS deploys our major releases with new capabilities and is allocated time for commissioning and issue resolution. With the ever-increasing demand for time on NIF, pressure exists to reduce the frequency and/or duration of these maintenance windows. This will result in demands for higher control system availability and reliability.

Figure 1 shows the Henderson & Clark [8] mapping of recent improvements resulting from the earlier mentioned influences. The changes within ICCS cover a broad spectrum of component based and system level architectural changes impacting multiple interfaces. Different strategies were necessary for dealing with both types of change.

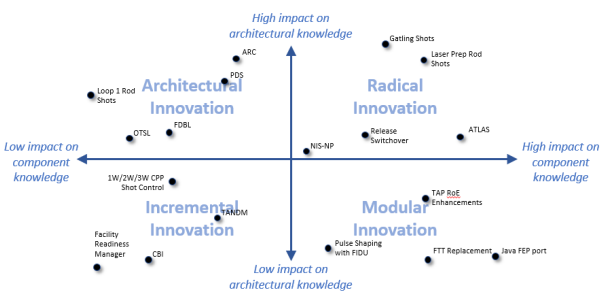

Figure 1: Henderson & Clark mapping of several recent and planned ICCS enhancements.

SYSTEMS ENGINEERING V-MODEL

Figure 2 shows the processes used by NIF for control system design and development. The initial control system development was a highly managed and successful process. As the program shifted to an operations and maintenance phase, this V-model continued to be followed however as changes continued, the activities used to link system decomposition and definition to integration/test phases gradually drifted out of date. These documents were not intrinsically updated as part of our development processes. In addition, there was a high dependency on manual processes and employee domain knowledge in the lower level of the V-model. As the number of system level capabilities grew, and the number of devices and their features expanded, the manual testing process was no longer sufficient both from an efficiency and reliability perspective.

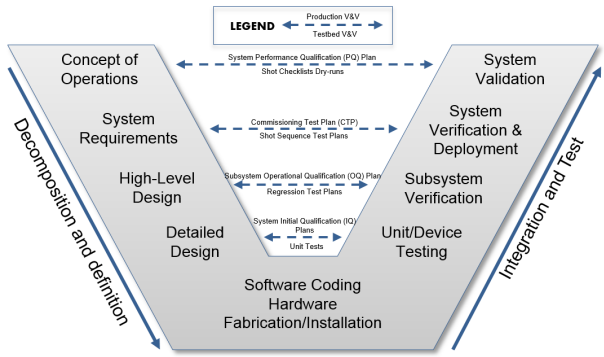

Figure 2: ICCS's adaptation of the Systems Engineering V-Model.

AUTOMATED TESTING

Component Level Testing

Earlier efforts to maintain a separate QA team to maintain and update an automated test suite [9] at the backend of the software development process was difficult due to budget constraints, increasing project scope, and schedule pressure. Lessons learned from Java porting activities [10] quickly identified that automated testing was a necessity. Relying solely on manual testing alone:

- could not provide sufficient depth in test coverage
- could not guarantee repeatable results
- was time consuming especially under a high rate of software change
- could not adequately address performance, load, and endurance testing requirements

The porting team paused on porting activities to focus on an automated test framework that could be incorporated into a continuous integration environment. A test-driven development strategy was applied which allowed the developers to initially evaluate their test cases against the Ada versions of the software. These test cases could then be leveraged on, and compared against, the completed Java code in an offline test environment using emulated and real devices.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THCPL06

The architecture of our device emulation was reworked to allow for improved offline qualification of device level software. Emulation was originally implemented at the device level of the software for the purpose of testing beamline operation and system scalability [11]. This had the disadvantage that device level code pushed to the operational environment could not be exercised in an offline test environment without having access to real hardware. Although the offline testbed had a broad selection of hardware representing NIF, there was not 100% coverage. In addition, it was not feasible to use the real hardware during simulated shot operations due to the absence of a fully configured NIF beamline and actual laser light. During the Java porting, emulation was implemented at the lower controller layer which provided a thin-client programming interface to the hardware. This would allow the device software to be fully exercised in an emulated mode, and during simulated NIF shots.

Test cases were annotated to indicate which tests were slow and fast running tests, and which could be exercised on real hardware. The test framework leveraged these annotations to determine which tests to run in the continuous integration environment (fast tests not requiring hardware). The remaining tests would be run by the developers during their own unit testing activities.

Results of the automated unit tests are compiled to show how the development Java software compared to the production Ada software. Comparisons were made not only on the individual test case pass/fail status, but also from a performance perspective. Figure 3 shows sample output from an automated test run.

		Ada		Java	
	Test Case	**Status**	**Time (s)**	**Status**	**Time (s)**
Common					
	checkEmulationStatus	PASS	3.002	PASS	2.999
	checkHealthStatus	PASS	3.003	PASS	3.004
	checkInterfaces	PASS	3.009	PASS	3.421
MotorBasicsTest					
	checkSettings	PASS	3.031	PASS	3.422
	getComponentStatus	PASS	3.033	PASS	3.011
	isMovingWhileBusy	PASS	7.433	PASS	9.012
	motorNotMovingWhileIdle	PASS	3.006	PASS	3.002
	setBacklash	PASS	3.017	PASS	3.011
	setInvalidHighBacklashFails	PASS	3.011	PASS	3.001
	setInvalidLowBacklashFails	PASS	3.021	PASS	3.022
MotorFindLimitTest					
	findLimitWhenMotorStoppedFails	PASS	7.269	PASS	15.180
	findLimitWhileBusyFails	PASS	6.587	PASS	8.104
	findLimitWithInvalidKeyFails	PASS	3.067	PASS	3.062
	findLimitWithValidKeySucceeds	PASS	28.230	PASS	30.734
	findNegativeLimitSucceeds	PASS	34.215	PASS	36.725
	findNegativeLimitWhenAtNegativeLimitSucceeds	PASS	15.614	PASS	20.240
	findPositiveLimitSucceeds	PASS	20.602	PASS	22.357
	findPositiveLimitWhenAtPositiveLimitSucceeds	PASS	15.619	PASS	20.202

Figure 3: Sample results from an early run of automated tests on a Java motor device.

Test cases were designed using black box methods where success and failure cases were identified from the device's API, as well as their add-on interfaces (e.g., setpoints, archiving).

The automated test suite will help ensure high reliability when delivering new and changed component software to the operational environment, it also provided requirements for new developers to understand how the devices operated. This helps mitigate the loss of developers responsible for the software's development and accelerate the onboarding of new staff. The development of the automated tests does not eliminate the need for manual testing. Having eyes on the system is still a necessity, and we are using the QA team to perform more exploratory testing at the component level in addition to their formal shot testing activities.

Facility Timing Transmitter (FTT) Replacement

The FTT converts the master timing clock to a standard communication frequency that synchronizes on an optical serial data stream distributed across NIF. The replacement of the FTT hardware was a recent success story within ICCS that took advantage of our automated test methodology. The FTT hardware was some of the original hardware on the project and was custom made to NIF's specifications [12]. Only a limited number of FTTs had been procured, and we were starting to see hardware failures which were draining our limited supply of spares. Since we originally contracted out and procured the FTTs, the manufacturer has been bought and sold multiple times. The new owner was not interested in engaging with us on a replacement without a substantial initial financial investment. Although we had the original FTT firmware source code in escrow, it was believed that upgrades to the deployed FTT firmware did not make it into the source code, rendering parts of it obsolete. NIF's subject matter experts on timing had retired prior to us starting discussions on an FTT replacement.

Greenfield Technologies (GFT) was selected as the new FTT hardware manufacturer. To supplement the original interface control document, the control team provided to them "ICCS-in-a-box" complete with an FTT automated test suite to thoroughly exercise the FTT interface. The system layout is shown in Figure 4. The ICCS server included the entire ICCS release and Oracle database running on a single host. The configuration was customized to allow starting a single instance of ICCS complete with a single master timing front end processor and FTT, with the necessary framework processes. Completion of factory acceptance testing depended on the successful passing of all automated tests. The activity completed with GFT successfully delivering an FTT that was a drop-in replacement for the FTT hardware in production.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THCPL06

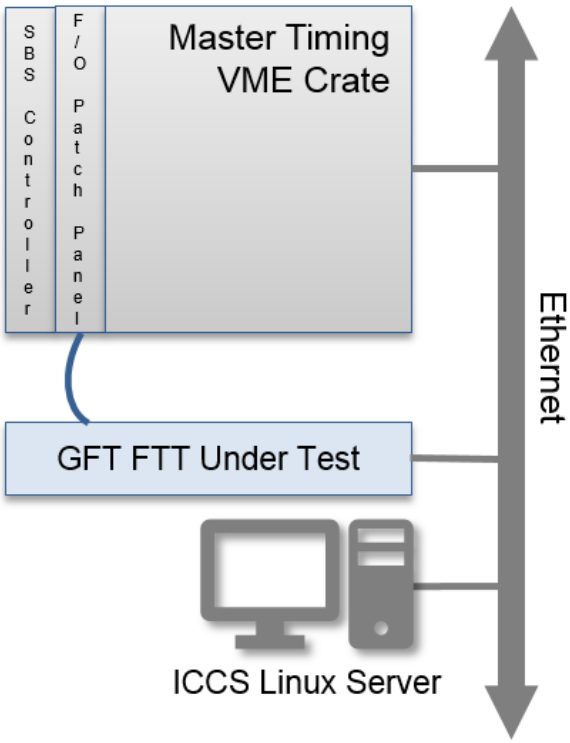

Figure 4: System configuration for GFT to qualify the FTT under development.

System Level Testing

Architectural changes, especially those that introduce new shot types or mode of operation, causes exponential increases in the number of permutations of shots that need to be run. During integration testing, developers and the integration manager run shots to qualify the software prior to the start of formal offline testing [11]. Due to the number of simulated experiments that would need to be run to fully exercise ICCS, an automated shot testing (AST) framework was needed to maintain quality and improve efficiency.

AST is part of a JavaScript and shell script set of tools that would start the ICCS system, pre-configure an emulated NIF environment, execute multiple shot cycles, and shutdown the system. Although much of the shot cycle was already automated [13] there were still multiple manual steps that required operator actions, which includes positioner alignment. AST replaces the human interaction for these manual actions through scripted actions. In addition, AST was designed to inject off-normal conditions to verify the system response. This includes facility reconfiguration events during the shot cycle, and component or subsystem level failures. The sequence of simulated experiments, configuration data, and injected off-normals and other commands are documented within an AST scheduler file that is used by a schedulding process to start the AST.

Failures for each AST run are written to a specified location. We are leveraging Splunk to allow a user to gather additional information on the shot [14]. Figure 5 is showing a sample of a Splunk dashboard with information on the AST shot cycle test execution results which includes failures, alerts, experiment to shot ID mapping, as well as performance metrics.

Like the component level automated test frameworks, AST is meant to replace much of the manual shot testing in integration conducted prior to releasing software to the formal test environment. However, we are still dependent on dedicated manual shot testing by the developers to perform deep dives into the subsystems managed by the shot cycle frameworks to ensure high quality. Future enhancements include the ability to compare the shot cycle performance within AST runs to assess the impact on the shot's critical path. We also intend on including AST in our continuous integration environment. With regular automated shot testing run during development, we hope to shorten the dedicated integration cycle to provide more time for software development during our quarterly releases.

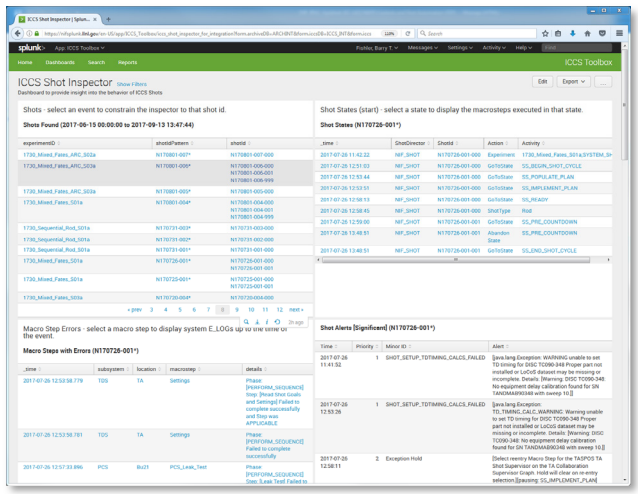

Figure 5: Splunk dashboard showing details of a simulated shot cycle.

PLATFORM AGNOSTIC SOLUTIONS

In addition to providing increased availability to developers when attempting to fill open job positions, the Java porting activity has opened the door to platform agnostic solutions. The supervisory layer of ICCS, and some components of the FEP layer can now be run on virtual machines [15] running a mix of Linux and Windows operating systems. With Java being an interpreted language, this has significantly simplified our build environment. Compiler support for multiple operating systems is no longer a necessity.

The Java based FEPs run on the highly customizable Gentoo Linux distribution. VME based single board computers (SBC) were bought in bulk to minimize manufacturing changes that are expected to occur. The selection criteria for the boards was based primarily on hardware compatibility with our existing VME chassis which includes

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THCPL06

the ability to boot diskless, and the ability to support standard open source Linux. The final selection was then based on price once the SBC was certified for operational use during an exhaustive performance evaluation and offline testing. Figure 6 shows the procurement rate for the SBCs vs. the VME conversion rate as part of our Java porting activity.

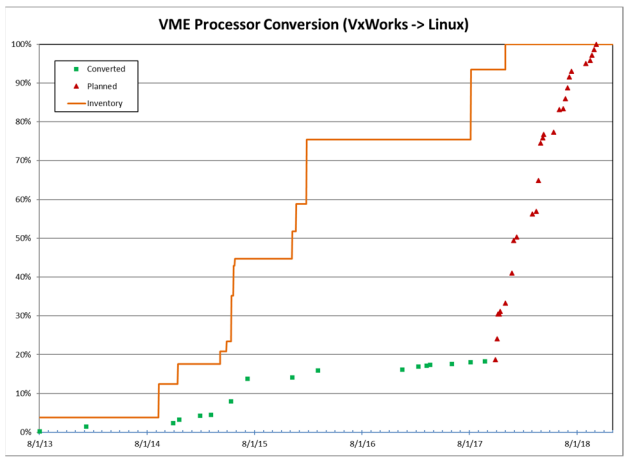

Figure 6: VxWorks/Ada FEP processor conversion to Gentoo/Java schedule and progress. The spike in completion represents the completion of the final device classes necessary to convert the Alignment Controls and Preamplifier Module FEP [16] conversions which make up the majority of the VME based processors.

CONCLUSION

Modernization of the control system technology is a necessity to sustain the facility for the next 20 years, and to attract and retain new staff. The reliance on subject matter experts must be replaced with automated verification techniques to aid in efficient regression testing and also define the correct behavior of the system. This assists in accelerating the onboarding of new development staff. The automated testing needs to address component based changes as well as system architectural changes to ensure adequate testing depth for all subsystems.

ACKNOWLEDGMENT

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

REFERENCES

[1] E. I. Moses, "Overview of the National Ignition Facility", Fusion Science and Technology 54 (2008).

[2] J. D. Lindl, *et al.*, "The US ICF Ignition Program and the Inertial Fusion Energy Program", 30th EPS Conference on Controlled Fusion and Plasma Physics, St. Petersburg, Russia, 2003.

[3] P.J. Van Arsdall, *et al.*, "National Ignition Facility Integrated Computer Control System", Third Annual International Conference on Solid State Lasers for Application (SSLA) to Inertial Confinement Fusion (ICF), Monterey, CA, 1998.

[4] L. Lagin, *et al.*, "Shot Automation for the National Ignition Facility", ICALEPCS, Geneva, Switzerland, 2005.

[5] Woodruff, J. (1997). *FY97 ICCS prototype specification*. United States. doi:10.2172/568041

[6] Van Wonterghem, B M, Burkhart, S C, Haynam, C A, Manes, K R, Marshall, C D, Murray, J E, … Wegner, P J. (2003). NIF Commissioning and Initial Performance Results. United States. Retrieved from http://www.osti.gov/scitech/servlets/purl/15007234

[7] G. Brunton *et al.*, "Status of the National Ignition Facility (NIF) Integrated Computer Control and Information Systems", ICALEPCS 2017, Barcelona, Spain, 2017.

[8] Henderson, R., & Clark, K. (1990). Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. Administrative Science Quarterly, 35(1), 9-30. doi:10.2307/2393549

[9] Zielinski, J. (2012). National Ignition Facility TestController for automated and manual testing. Fusion Engineering and Design, 87(12), 1994-1996. doi:10.1016/j.fusengdes.2012.09.016

[10] L. Lagin, *et al.*, "Status of the National Ignition Facility (NIF) Integrated Computer Control and Information Systems", ICALEPCS, San Francisco, California, USA, 2013.

[11] D. Casavant *et al.*, "Testing and Quality Assurance of the Control System During NIF Commissioning", ICALEPCS 2003, Gyeungju, Korea, 2003.

[12] G Coutts, The Design and Implementation of the Integrated Timing System to be Used in the National Ignition Facility, ADA497126, 1999.

[13] L. Lagin *et al.*, "Shot Automation for the National Ignition Facility", ICALEPCS, Geneva, Switzerland, 2005.

[14] M. Fedorov, *et al.*, "Leveraging Splunk for Control System Monitoring and Management", ICALEPCS 2017, TUCPA02, Barcelona, Spain, 2017.

[15] T. Frazier, *et al.*, "A Virtualized Computing Platform for Fusion Control Systems", ICALEPCS 2011, Grenoble, France, 2011.

[16] P. VanArsdall, NIF Integrated Computer Controls System Description, URCL-ID-140016, 1998, https://e-reports-ext.llnl.gov/pdf/239928.pdf