SOFTWARE ENGINEERING PROCESSES USED TO DEVELOP THE NIF INTEGRATED COMPUTER CONTROL SYSTEM*

A. P. Ludwigsen#, R. W. Carey, R. D. Demaret, L. J. Lagin, U. P. Reddi, P. J. Van Arsdall
LLNL, Livermore, CA 94551-0808, U.S.A.

Abstract

The National Ignition Facility (NIF) at Lawrence Livermore National Laboratory is a 192-beam laser facility for high-energy density physics experiments. NIF is operated by the Integrated Computer Control System (ICCS), which is comprised of 60,000 devices deployed on 850 computers. Software is constructed from an object-oriented framework based on CORBA distribution. ICCS is 85% complete with over 1.5 million lines of verified code now deployed online. Success of this large-scale project was keyed to early adoption of rigorous software engineering practices including architecture, code design, configuration management, product integration, and formal verification testing. Verification testing is performed in a dedicated test facility following developer integration. These processes are augmented by an overarching quality assurance program featuring assessment of quality metrics and corrective actions. Engineering processes are formally documented and releases are managed by a change control board. This talk discusses software engineering and results obtained for the NIF control system.

INTRODUCTION

NIF will be the world's largest and most energetic laser experimental system, providing a scientific center to study inertial confinement fusion (ICF) and matter at extreme energy densities and pressures. NIF's laser beams are designed to compress fusion targets to conditions required for thermonuclear burn, liberating more energy than required to initiate the fusion reactions. NIF is comprised of 24 independent bundles of 8 beams each using laser hardware containing 60,000 control and diagnostic points.

NIF is operated by the large-scale Integrated Computer Control System (ICCS) in an architecture partitioned by bundle and distributed among over 850 front-end processors and supervisory servers. The primary requirement for the control system on this facility is to automatically fire and diagnose laser shots every four hours. This process begins with reading campaign goals from the laser physics model and deriving equipment settings based on the goals. Laser alignment is then automatically performed, including wavefront correction. Equipment is then configured to the derived settings to meet the laser performance and diagnostic goals. Shot countdowns are performed to verify the derived settings by firing the pre-amplifiers and comparing the results with the laser physics model. Adjustments to the device settings are made until the laser performance criteria are met. Once settings are validated, the main amplifiers are fired in a final countdown. Shot data is archived on each of the countdowns.

The requirement to achieve efficient and reliable shot operations using a minimum control room staff dictates a highly reliable and automated control system. The ICCS meets these requirements by providing distributed computer controls throughout the facility to manage operation of the 60,000 devices.

ENGINEERING PROCESSES

A wide range of engineering processes and procedures have been implemented to ensure and maintain high quality and reliability. A standardized architecture is complemented by formal code design and engineering review. Strict release planning and configuration management ensure an on-time delivery of required functionality to meet project schedules. Formal integration of each release is conducted by the development team to demonstrate new functionality and provide an efficient environment to quickly correct any defects. Formal offline testing by an independent test team then verifies the software and associated database is ready for on-line delivery to NIF. Offline testing also specifically verifies all critical functions meet requirements. Formal training is performed to instruct control room operators on the new capabilities and obtain feedback on usability issues.

Architecture

ICCS controls are based on a segmented, partitioned and layered architecture that is data-driven, distributed and object-oriented. Segmentation into distinct control technologies reduces complexity by separating out safety interlocks and common industrial controls from the laser control system. This paper focuses on the laser control system segment. System scaling is assured by partitioning controls and software among a set of computers dedicated to controlling a single bundle of 8 beams, which is the unit-cell repeating structure of the NIF laser architecture. Controls are then replicated to implement all 24 bundles in NIF. Replication of software for the partitions is primarily accomplished by copying the processes and control database. Software layers divide control system functionality into appropriate levels ranging from close to the hardware or higher up to aggregate control of the lower layers and achieve greater integration. Layering leads to simpler and more maintainable code designs [1].

ICCS is developed using a combination of custom software and commercial off the shelf (COTS) technology, including open source technologies.

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

ludwigsen1@llnl.gov

Communications between applications is managed by CORBA protocols. Ada and Java are the primary languages used to develop the custom software, with C/C++ used for some embedded controller applications. IBM Rational, Ada Core Technology, and open source supply interactive development environments (IDEs). Research Systems International's Interactive Data Language is used for analysis of on-line data and images. XML supports workflow models and application scripts. Oracle databases manage the vast quantity of information that is used, collected, and archived by the control system.

Another tactic from the architectural toolbox is the collection of system-wide functions into servers. Functions include application startup, monitoring, and shutdown; alert and event monitoring and publication; reservation control; status monitoring and publication; and general database access. These are consolidated into server applications because they tend to be tightly coupled to the database. Rather than impact a large portion of the control system, database schema and generation changes can be accommodated more easily by modifying servers.

In the lowest layer, I/O control is managed by Front End Processors (FEP) and embedded controllers where true real-time control is needed. FEP software is divided into a public interface and a private component, respectively Device and Controller objects. Controller code is customized for the equipment and provides protocols handle register access and communications. Software devices manage the data conversion between the applicable engineering units and I/O data from one or more controllers. Higher-level clients communicate with the control points using CORBA interfaces. CORBA specifications define public interface contracts for the distributed control system, so these interfaces are held under strict configuration management.

FEP control and status information is managed and published to User Interfaces and other client applications by Status and Control Supervisors (SCS). Director objects are typically organized to mimic the modularity of laser hardware in Line Replaceable Units, and they may also be used to provide higher-level data processing capabilities. Director objects have CORBA interfaces and contain one or more Staff objects that manage data collection and processing. Staff objects frequently add value to information with condition flags for validity of the data being observed, the health of the communications connection, and other common attributes.

At the highest level of integration, shot sequencing and coordination are managed by a separate set of Shot Director and Collaboration Supervisor applications that also use the Director-Staff architecture. The data driven Shot Director application manages a software shot model to coordinate major functions along with the interactions between subsystems. Collaboration Supervisor applications coordinate sequencing of the major functions within an associated set of Shot Supervisors to control a bundle. Shot Supervisor applications manage laser shot activities on a subsystem basis by reading information from the laser physics model, defining equipment settings to be used, and configuring the equipment via Devices and Directors [2].

Software objects are uniquely identified using a string-based taxon naming convention that is easily understood and can be expanded to differentiate unique control points. Taxon identifiers are highly controlled because many external processing codes use them to reference data collected during shots, to provide device settings and calibration updates and to generally access control system information.

Since ICCS has been under development for over ten years, some applications are already being migrated to newer, cheaper or more efficient technology. Some Ada code is being migrated to Java to take advantage of the lower cost of IDEs and Java's higher portability. Automatic alignment and optics inspection image acquisition applications are being changed from more cumbersome XML scripts to use Java database-driven sequences. This is being driven mainly by the need to mediate access to common control points used in the alignment and image capture processes [3]. Database entry is migrating to a web–based form system from the present Oracle SQLForms®. This will provide easier-to-use data entry and verification that is operable on standard web browsers. Other migrations are naturally driven by hardware upgrades. The original analog imaging cameras designed into NIF were upgraded to higher resolution Firewire sensors to improve performance.

Requirements, Designs, and Reviews

One area that challenged the ICCS team was requirements specification. Requirements for FEP software, servers, and initial supervisory systems were initially documented to get started. Supervisory requirements continued to be specified after the laser hardware and control system capabilities were initially deployed and user feedback was obtained. Requirements evolve as operability enhancements are discovered during early use of the system. The ICCS architecture periodic incremental development cycles allowed the control system to adapt to these evolving requirements [Figure 1].

Well-designed abstractions in the object-oriented framework are essential tools that simplify the design of software components. Most new control applications will fit into existing design patterns, either by extension or aggregation. Occasionally, completely new designs are developed to fit unique requirements.

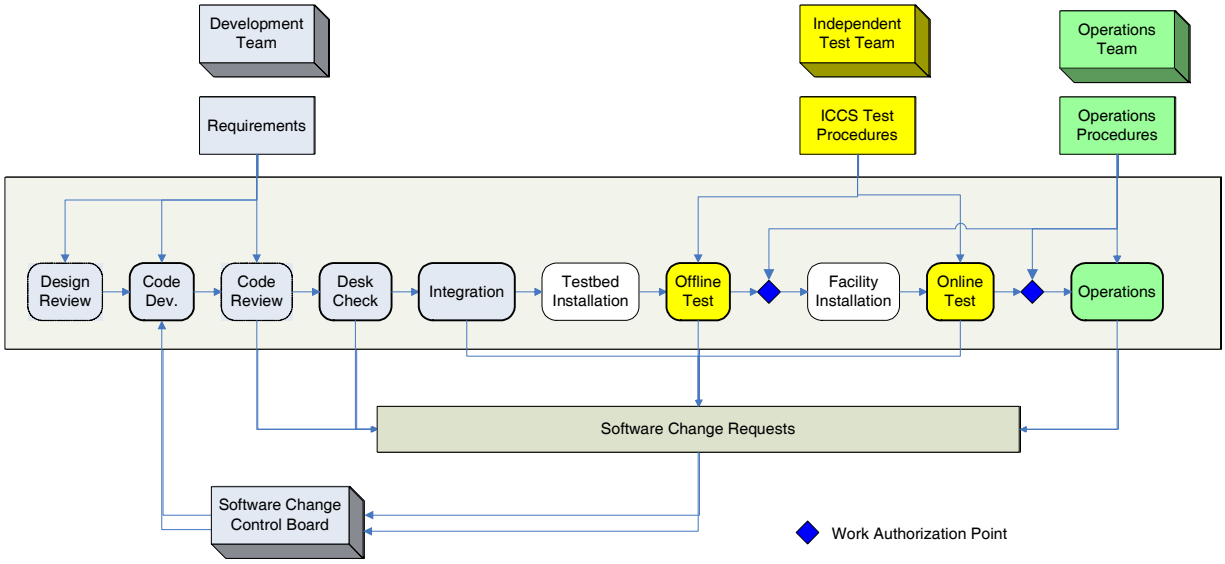

Figure 1: Software Development and Deployment Process .

A formal design review team critically reviews new application designs against the established architecture and existing requirements before approving code development. Changes to the basic architecture can disturb software in unintended ways, so the review includes looking for impacts to the database, framework, or critical interfaces. These reviews frequently result in discovery of implied framework enhancements. New requirements are also examined to determine if the solution proposed is optimal. The review team includes hardware engineers and operations representatives to ensure that requirements are well understood.

Other peer reviews also help maintain the quality of ICCS software. Any code module that undergoes significant modification requires a formal change review by a select team of software developers. This review may also be held if a particularly complex code module is affected. The review team includes developers familiar with the subject area, developers from other areas and members of the test team. A diverse review team ensures consistency with the architecture and assists development of appropriate verification tests.

All software changes are reviewed before integration by at least one other developer in a Desk Check inspection process that ensures:

- Specific code changes are correct
- Unit testing is performed
- Change documentation is complete and
- Configuration management followed.

Documented coding practices and procedures guide the reviewers during inspections. Reviews also provide an excellent forum to train new developers on the code base and inculcate accepted standards and styles.

Configuration Management

Strict configuration management of the code base ensures the product set is up-to-date and can be built without errors. It also ensures that the size of a release is manageable such that project schedules can be met. All software changes, be they problem corrections or feature enhancements, are authorized by the Software Change Control Board (SCCB) before work is started. A database-driven change tracking system maintains the inventory of the proposed and authorized changes (called Software Change Requests), as well as tracks the work status. Additionally, changes in major interfaces, object identifiers, or database schema are signed-off by permit to ensure all affected parties incorporate the change.

The content and schedule of each release is planned by management to conform to the expected commissioning effort and shot plan. Four levels of software releases help manage complexity, while minimizing response time to address operational issues that may arise. Major releases are scheduled approximately every twelve months to deliver significant feature additions. Major releases are also planned to introduce changes to the framework. Minor releases delivered three or four times per year include feature additions as well as software fixes. Service packs occur monthly to deliver software fixes that must be more coordinated. Small patches are scheduled as needed to address urgent issues.

The amount of work assigned to each release is continually monitored to determine need for schedule revision or content reduction based on available manpower, estimated workload and project deadlines. Release progress is closely monitored to permit staff reassignment whenever a particular area encounters difficulty.

A dedicated configuration management (CM) team is integral to a reliable release process. These specialists maintain the code base across multiple environments, including the challenges of mixed languages, multiple development systems, and different target architectures. The CM team has the skills and mandate to develop automated tools that ensure all releases are up-to-date with respect to changes made in interim releases such as service packs or patches. They also roll back online data and files to the development environment, where

applicable, to allow release regression and mitigate unexpected conflicts.

INTEGRATION AND TESTING

Quality control testing is performed by both the development team and the test team to ensure reliability of the deployed software. The code is unit tested by developers as they make changes. Once coding is ready, a period of integration testing brings all elements of the release together and verifies reliability, confirms new features are implemented properly, and ensures no regressions have occurred. After completing the integration phase, formal off-line testing is performed by the test team to verify the release and database configuration, and to confirm no errors were introduced during the deployment process [4].

Integration testing is performed on the entire release using a separate database instance to confirm that database updates have been assembled correctly. Most control points have been modeled by software emulators so testing closely mimics actual device behavior. Selected hardware systems have simulators in the test bed that operate like the full size hardware. These capabilities allow testing of the software release in its entirety, including shot cycle emulations that run in real-time. Daily reports of the issues identified are distributed to the development staff so that errors can be quickly addressed.

The integration period includes several testing regimes. Regression testing, both manual and automated, verifies that unmodified portions of the release will perform correctly when deployed. Software Change Request verification evaluates whether all approved software changes have been successfully incorporated into the release. Shot testing verifies the shot model properly coordinates shot activities. Mock experiments defined to exercise the automated shot setup and sequences include various beam destinations, laser configurations, power levels, and pulse shapes.

Key software and database developers perform most integration verification. The rest of the development staff, while nominally working on the follow-on release, is on call to quickly address any issues found. An “extreme programming” work process is encouraged during integration that allows defects to be fixed directly in the release code base as they are found, thus allowing the integration effort to continue. Complex defects found during integration take priority over the follow-on release except for urgent issues from the operations environment.

Formal testing is performed offline by an independent formal test team. Formal offline testing includes at least one real component of each type of control point used in NIF to verify communication paths. Signal-based and scale-model simulators are used where the use of real hardware is impractical. Auxiliary laser facilities are sometimes used to verify the software on actual NIF hardware.

Verification of device controls involved in machine protection functions is an important formal test activity. These tests confirm that devices involved in protecting the laser from potential damage are controlled during shot cycles to prevent inadvertent operation, once set to the shot configuration. Experiments designed for the final on-line test are used to verify the software will support operations when deployed to NIF.

An inventory of automated tests is being developed to efficiently leverage the knowledge of the developers and formal test team members [5]. These automated tests will decrease time spent in manual testing, while assuring test case coverage is maintained. It also allows expanded use of limited testing resources by permitting tests to be run unattended during the off-hours.

Quality Assurance

The Quality Assurance team reviews information collected by the change tracking system to provide a monthly quality metrics report. This report identifies short-term spikes in defects as well as long-term trends. The backlog of change requests is also tracked to identify areas where developer resources may need to be re-assigned when progress is slowing down. It is also used by management to support requests for additional development time or staff to meet the NIF’s operational needs.

Team Composition

The development staff features a wide range of experience. This includes GUI specialists, hardware control engineers, language experts, database specialists, and tool smiths. This mixture of skills and experience provide synergy to improve results and productivity.

PROGRESS TO DATE

Over 85% of the estimated 1.8 million lines of code for ICCS have been deployed to NIF for routine commissioning and shot operations. Quality control processes in place have been very effective, consistently finding 90% of defects before the code was delivered. NIF recently completed commissioning a full laser bay of 96 beams and is presently activating bundles in the second laser bay [6]. ICCS is routinely used on hundreds of shots and the control system performance easily meets the four-hour shot sequence requirement.

A Precision Diagnostic System (PDS) is in routine use to evaluate laser operating scenarios at the equivalent of target chamber center. This system provides laser diagnostic measurements for laser scientists and engineers to characterize performance and improve the operating models and procedures used by ICCS to perform shots. PDS results have improved pulse generation techniques that have already been incorporated.

FUTURE PLANS

The next area of development is the extension of ICCS to support target shot operations for the National Ignition Campaign. Automated control of target diagnostics, alignment to target chamber center, cryogenic targets and

an advanced radiographic capability are in the early stages of development.

Thirty types of target diagnostic instrument systems are scheduled for the National Ignition Campaign. These include hard and soft X-ray spectrometers and imagers, optical diagnostics and neutron diagnostics. Many diagnostics were developed and used during the NIF Early Light campaign in 2003-2004. Information learned from this work is being incorporated into target diagnostic capability. This includes a re-factoring of the software in this area to increase modularity and code reuse.

Cryogenic targets are key to ignition experiments scheduled to begin in 2010. The on-line system will control and characterize Deuterium-Tritium fuel ice layer formation in ignition targets using cryogenic temperature controls and X-ray image diagnostics. The ice formation process is expected to take several hours, which will drive changes to the shot control software for synchronizing the shot cycle and cryogenic systems when the target is ready.

The Advanced Radiographic Capability (ARC) will provide extremely short laser pulses on a few beam lines for back-lighted diagnostics and fast igniter applications. This requires some beams to be easily converted from normal operation to ARC mode. Pulse injection for these special beams will also need to be selectable. Additionally, new alignment techniques will be needed.

SUMMARY

Rigorous application of software engineering practices can deliver reliable large-scale control systems for complex physics machines. Principles of architecture, code design, configuration management, product integration, and formal verification testing all contribute to successful deployment of integrated products that meet both customer expectations and aggressive project schedules. Management practices guide the development process at all times. Quality assurance provides measurements that lead to timely corrective actions.

A well-defined and open architecture is essential to efficient software development, long-term maintainability, and consistent results. It simplifies code development by reducing software complexity through reusable designs, frameworks, and coding templates.

Code design must be managed to control complexity. Reviews incorporated into the design process ensure requirements are reasonable and the design will meet requirements. Inspections included as a routine part of development help verify the design has been implemented correctly and also ensure accepted coding practices and standards are used.

Strong configuration management is critical to success as well. Incremental releases with active management and quality controls can effectively mitigate the interrelated challenges of technical risk, schedule pressure and available manpower. A software change tracking system greatly assists in this process.

Formal product integration helps achieve reliable control system delivery. The integration effort should be extensive and include verification of specific changes and assurance that regressions have not occurred. Testing should exercise software at the highest level of integration to confirm that all code elements work together as planned.

Both offline and online formal independent testing is incorporated into the deployment process. The test team should be familiar with common operator issues as well as have knowledge of software development concerns. This testing validates required functionality and critical control execution. It can identify usability issues where software performs functions correctly but is difficult for operators.

An experienced development staff with a wide range of skills will improve productivity and results. A dedicated configuration management team should build and confirm software release content.

Finally, management and quality assurance practices should augment the development process. These include keeping the staff informed of the release schedule and its progress. Ongoing review of data from the change tracking system identifies issues and highlights trends that may need increased attention.

REFERENCES

[1] R. Carey, et al, “Status of the Use of Large-Scale CORBA-Distributed Software Framework for NIF Controls”, ICALEPCS 2005, Geneva, Switzerland, October 2005

[2] L. Lagin, et al, “Shot Automation for the National Ignition Facility”, ICALEPCS 2005, Geneva, Switzerland, October 2005

[3] K. Wilhelmsen, et al, “Automatic Alignment Systems for the National Ignition Facility”, ICALEPCS 2007, Knoxville, October 2007

[4] D. Casavant, et al, “Testing and Quality Assurance of the Control System during NIF Commissioning”, ICALEPCS 2003, Gyeongju, Korea, October 2003

[5] J. Zielinski, “NIF ICCS Testcontroller for Automated and Manual Testing”, ICALEPCS 2007, Knoxville, October 2007

[6] L. Lagin, et al, “Status of the NIF Integrated Computer Control System”, Sixth IAEA-TM on Control, Data Acquisition, and Remote Participation for Fusion Research, Inuyama, Japan, June 2007