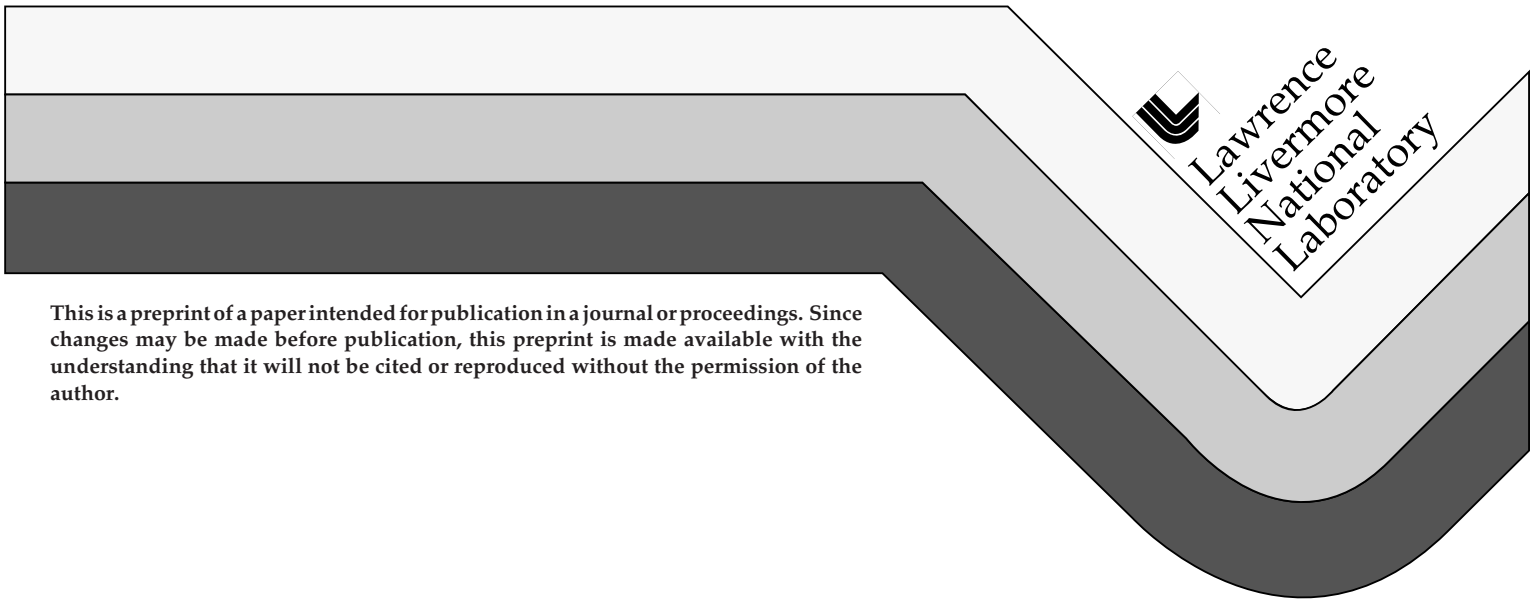
UCRL-JC-129873
PREPRINT

National Ignition Facility Integrated Computer Control System

P. J. Van Arsdall, R. C. Bettenhausen, F. W. Holloway, R. A. Saroyan, and J. P. Woodruff

This paper was prepared for and presented at the Third Annual International Conference on Solid State Lasers for Application (SSLA) to Inertial Confinement Fusion (ICF) Monterey, CA June 7-12, 1998

June 1998



National Ignition Facility Integrated Computer Control System

P.J. Van Arsdall, R.C. Bettenhausen, F.W. Holloway, R.A. Saroyan, J.P. Woodruff

Lawrence Livermore National Laboratory,
PO Box 808 MS L-493, Livermore, CA 94551-0808

Abstract

The NIF design team is developing the Integrated Computer Control System (ICCS), which is based on an object-oriented software framework applicable to event-driven control systems. The framework provides an open, extensible architecture that is sufficiently abstract to construct future mission-critical control systems. The ICCS will become operational when the first 8 out of 192 beams are activated in mid 2000.

The ICCS consists of 300 front-end processors attached to 60,000 control points coordinated by a supervisory system. Computers running either Solaris or VxWorks are networked over a hybrid configuration of switched fast Ethernet and asynchronous transfer mode (ATM). ATM carries digital motion video from sensors to operator consoles.

Supervisory software is constructed by extending the reusable framework components for each specific application. The framework incorporates services for database persistence, system configuration, graphical user interface, status monitoring, event logging, scripting language, alert management, and access control. More than twenty collaborating software applications are derived from the common framework.

The framework is interoperable among different kinds of computers and functions as a plug-in software bus by leveraging a common object request brokering architecture (CORBA). CORBA transparently distributes the software objects across the network. Because of the pivotal role played, CORBA was tested to ensure adequate performance.

Keywords: Distributed control system, CORBA, Software engineering, Framework, Simulation, Object-oriented

1 Introduction

This paper presents the architecture of the National Ignition Facility (NIF) Integrated Computer Control System (ICCS). The NIF contains 192 laser beam lines that are focused on an inertial confinement fusion (ICF) capsule at target chamber center[1]. Each beam requires alignment, diagnostics, and control of power conditioning and electro-optic subsystems. NIF will be capable of firing target shots every 8 hours, allowing time for the components to cool sufficiently to permit precise re-alignment of the laser beams onto the target. In greatly simplified form, the beam line schematic is shown in Figure 1.

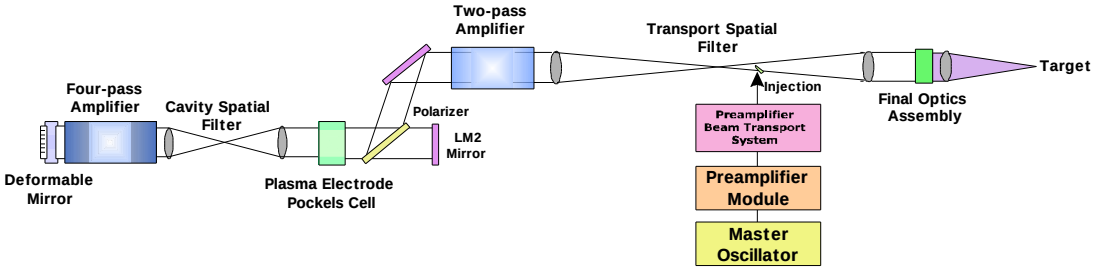


Figure 1 Simplified NIF beam line schematic

The NIF requires integration of about 60,000 atypical control points, must be highly automated and robust, and will operate around the clock. Furthermore, facilities such as the NIF represent major capital investments that will be operated, maintained, and upgraded for decades. The computers and control subsystems must be relatively easy to extend or replace periodically with newer technology.

The ICCS architecture was devised to address the general problem of providing distributed control for large scientific facilities that do not require real-time capability within the supervisory software. The ICCS architecture uses the client-server software model with event-driven communications. Some real-time control is also necessary; controls requiring

deterministic response are implemented at the edges of the architecture in front-end computer equipment. In any event, no hard real-time control is distributed over the computer network.

Over twenty distributed software applications will operate the NIF control system hardware from a central control room [Figure 2]. The software architecture is sufficiently abstract to accommodate diverse hardware and it allows the construction of all the applications from an object-oriented software framework that will be extensible and maintainable throughout the project life cycle. This framework offers interoperability among different computers and operating systems by leveraging a common object request broker architecture (CORBA). The ICCS software framework is the key to managing system complexity and, because it is fundamentally generic and extensible, it is also reusable for the construction of future projects.

Figure 2 Computer rendering of the NIF control room

A brief summary of performance and functional requirements follows [Table 1].

Computer restart	< 30 minutes
Post-shot data recovery	< 5 minutes
Respond to broad-view status updates	< 10 seconds
Respond to alerts	< 1 second
Perform automatic alignment	< 1 hour
Transfer and display digital motion video	10 frames per second
Human-in-the-loop controls response	within 100 ms

Table 1 Selected ICCS performance requirements

Summary ICCS functional requirements:

- Provide graphical operator controls and equipment status
- Maintain records of system performance and operational history
- Automate predetermined control sequences (e.g. alignment)
- Coordinate shot setup, countdown, and shot data archiving
- Incorporate safety and equipment protection interlocks

2 Control System Architecture

The ICCS is a layered architecture consisting of front-end processors (FEP) coordinated by a supervisory system [Figure 3]. Supervisory controls, which are hosted on UNIX workstations, provide centralized operator controls and status, data archiving, and integration services. FEP units are constructed from VME/VXI-bus or PCI-bus crates of embedded controllers and interfaces that attach to control points (e.g. stepping motors, photodiode sensors, and pulse power). FEP

software provides the distributed services needed to operate the control points by the supervisory system. Functions requiring real-time implementation are allocated to software within the FEP or embedded controller and do not require communication over the local area network. Precise triggering of 2000 channels of fast diagnostics and controls is handled during a 2 second shot interval by the timing system, which is capable of providing triggers to 30ps accuracy and stability. The software is distributed among the computers and provides plug-in software extensibility for attaching control points and other software services by using the CORBA protocol.

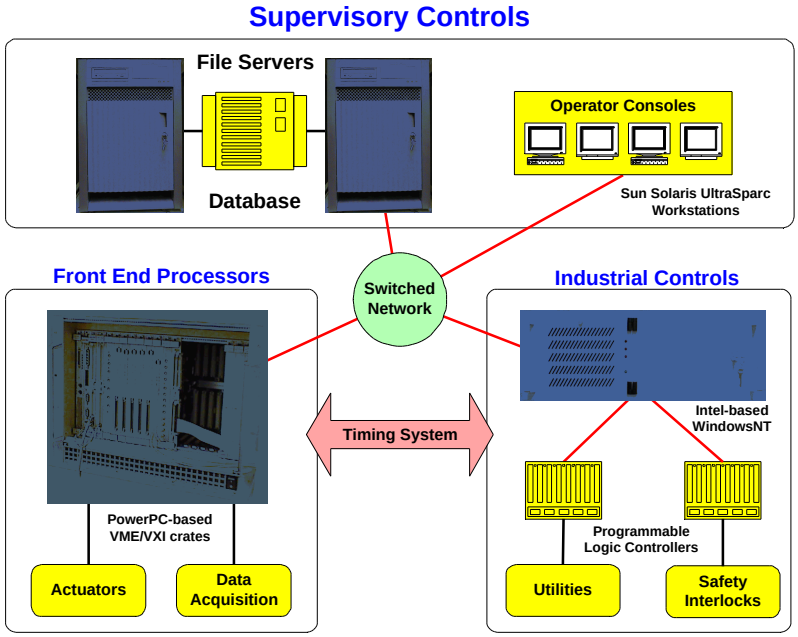


Figure 3 Integrated computer control system architecture

The operator console provides the human interface in the form of operator displays, data retrieval and processing, and coordination of control functions. Supervisory software is partitioned into several cohesive subsystems, each of which controls a primary NIF subsystem such as alignment or power conditioning. A dual server configuration provides enhanced performance with the added benefit of greater availability in the event one server fails. Several databases are incorporated to manage both experimental data and data used during operations and maintenance. The subsystems are integrated to coordinate operation of laser and target area equipment.

Front-end processors implement the distributed portion of the ICCS by interfacing to the NIF control points. The FEP software performs sequencing, data acquisition and reduction, instrumentation control, and input/output operations. The software framework includes a standard way for FEP units to be integrated into the supervisory system by providing the common distribution mechanism coupled with software patterns for hardware configuration, command, and status monitoring functions.

A distinct segment of the control system contains industrial controls for which good commercial solutions already exist that can be integrated into the framework. The segment is comprised of a network of programmable logic controllers that reside below the FEP and attach to field devices controlling vacuum systems for the target chamber and spatial filters, argon gas controls for the beam tubes, and thermal gas conditioning for amplifier cooling. This segment also monitors the independent Safety Interlock system, which monitors doors, hatches, shutters, and other sensors to establish and display the hazard levels in the facility due to high voltage, laser light, ionizing radiation, and non-breathable atmosphere. Potentially hazardous equipment is permitted to operate only when conditions are safe. Interlocks function autonomously to ensure safety without dependency on the rest of the control system.

There are eight Supervisor software applications that conduct NIF shots in collaboration with 19 kinds of front-end processor as shown in Figure 4. Seven of the subsystems are shown as vertical slices comprised of a Supervisor and associated FEP that partition the ICCS into smaller, loosely coupled systems that are easier to design, construct, operate,

and maintain. The eighth Supervisor is the Shot Director, which is responsible for conducting the shot plan, distributing the countdown clock, and coordinating the other seven.

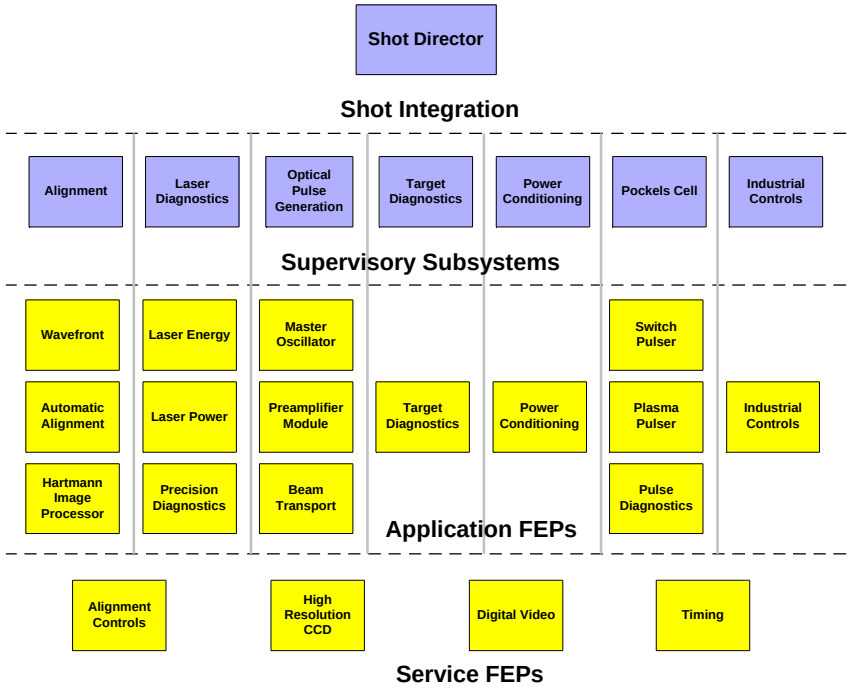


Figure 4 Software applications in the NIF control system

In the layer beneath the Shot Director are seven control subsystems. The Alignment Supervisor provides coordination and supervision of laser wavefront control and laser component manual and automatic alignment. The Laser Diagnostics Supervisor provides functions for diagnosing performance of the laser by collecting integrated, transient and image information from sensors positioned in the beams. The Optical Pulse Generation Supervisor provides temporally and spatially formatted optical pulses with the correct energetics and optical characteristics required for each of the beams. The Target Diagnostics Supervisor coordinates the collection of data from a diverse and changing set of instruments. The Power Conditioning Supervisor is responsible for high-level control and management of high voltage power supplies that fire the main laser amplifiers. The Pockels Cell Supervisor manages operation of the plasma electrode Pockels cell optical switch that facilitates multi-pass amplification within the main laser amplifiers. The Industrial Controls Supervisor provides monitoring of environmental and safety parameters as well as control of PLC subsystems for amplifier cooling, vacuum systems, argon and nitrogen gas systems, and final optics thermal control.

3 NIF computer system and network

Figure 5 shows the NIF computer system, which is comprised of 30 workstations, 300 FEP, and several hundred embedded controllers. The main control room contains seven graphics consoles, each of which houses two workstations with dual displays. The software applications are assigned to be operated from one primary console, although the software can be easily operated from adjacent consoles and remote graphics terminals located near the front-end equipment. The file servers provide disk storage and archival databases for the entire system as well as well as hosting centralized management and software object naming services necessary for coordinating the facility operation.

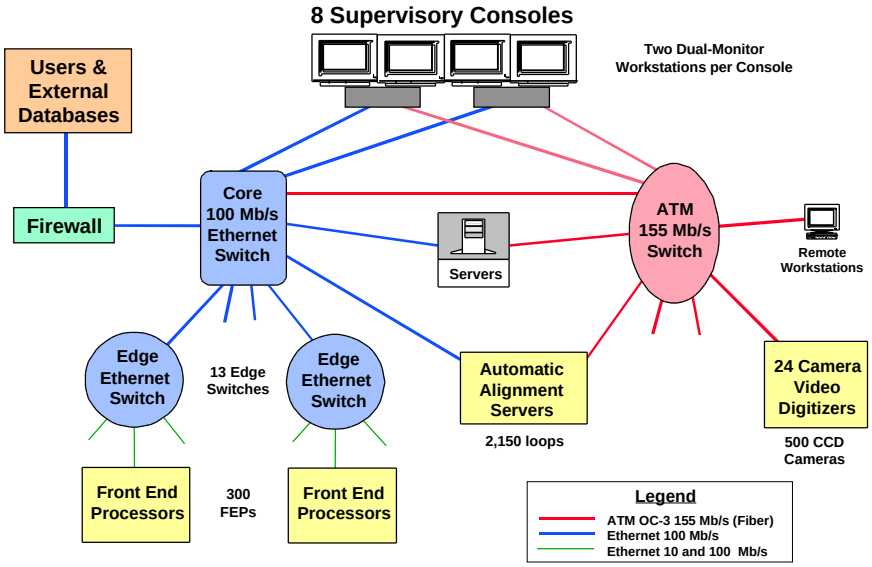


Figure 5 NIF computer system and network architecture

The network design utilizes both Ethernet and Asynchronous Transfer Mode (ATM) technologies to take advantage of the best features of each. ATM provides the connectivity for the systems requiring high-bandwidth or time-sensitive video transport which includes the video FEPs, the operator workstations, and the automatic alignment, timing, target diagnostics, and optics inspection systems. Ethernet provides connectivity for all other systems, which includes the large majority of systems. Since the ATM and Ethernet attached FEPs will communicate with the operator consoles and the file servers, the operator consoles and file servers have interfaces to both ATM and Ethernet. Although not shown in the figure, all nodes are connected to the Ethernet.

The design utilizes 155 Mb/s ATM and Ethernet at both 10 and 100 Mbit/s speeds, depending on expected traffic requirements. The operator consoles and file servers have 100 Mbit/s Ethernet connections while the FEPs have 10 or 100 Mbit/s connections, connected through Ethernet switches. Given the low cost and performance advantages of Ethernet switches relative to shared Ethernet hubs, switches with 100 Mb/s uplinks will be used.

TCP/IP is the protocol used for reliable data transport between systems, either over Ethernet or ATM. TCP provides retransmission of packets in the event that one is lost or received in error. The only traffic not using TCP will be digitized video and network triggers. Video is transferred using the ATM adaptation layer 5 (AAL5) protocol. Network triggers are broadcast to many end-nodes simultaneously using multicast protocols.

The ICCS network supports the transport of digitized motion video in addition to the more typical control, status, and shot data. The network transports video images of 640 x 480 x 8 bits/pixel at 10 frames per second between video FEPs (which capture and digitize the camera images) and operator workstations. Each uncompressed video stream requires about 25 Mbit/s of network bandwidth. Each operator workstation can display at least two video streams. Up to 24 video cameras are connected to a single video FEP, and as many as 3 simultaneous video streams are supported from a single FEP (i.e., 75 Mbit/s of bandwidth). Video compression is not currently used because of the high cost of encoding the video stream. Because the time to retransmit lost packets using TCP is excessive, it is not suitable for use with video traffic. Also, video transmissions need to be multicast, which TCP does not support.

Digitized video is sent via the ATM application programming interface (API) using the ATM Quality of Service (QoS) capabilities. The ATM API provides an efficient method of moving large, time-sensitive data streams, resulting in higher frames/sec rates with lower CPU utilization than alternative approaches, which is an important consideration for the video FEPs and console workstations. Performance testing of the prototype video distribution system indicate that 55% of the FEP CPU (300 MHz UltraSparc AXI) is used to broadcast 3 streams while 10% of the operator workstation CPU (300 MHz UltraSparc 3D Creator) is utilized for each playback stream.

Estimates of the peak traffic requirements for the various subsystems were analyzed as a basis for the network design. The expected peak traffic flows between subsystems in terms of messages per second and message size were specified. This data was combined and analyzed to determine peak throughputs into and out of each network-attached device. A discrete event simulation of the network was created in the Opnet Modeler network modeling tool to evaluate the performance of key scenarios. For example, it was shown that "network triggers" in the millisecond regime could be reliably broadcast between computers in the network via the UDP protocol. One application of network triggers is the arming of the video FEPs to enable digitizing the laser pulse during shots.

If bandwidth requirements increase in the future, the network architecture allows the integration of Gbit/s Ethernet and 622 Mb/s ATM technologies in a relatively straightforward manner.

4 ICCS Software Framework

The ICCS supervisory software framework is a collection of collaborating abstractions that are used to construct the application software. Frameworks[2] reduce the amount of coding necessary by providing pre-built components that can be extended to accommodate specific additional requirements. The framework also promotes code reuse by providing a standard model and interconnecting backplane (CORBA) that is shared from one application to the next. Components in the ICCS framework are deployed onto the file servers, workstations, and FEPs, as shown generically in Figure 6. Engineers specialize the framework for each application to handle different kinds of control points, controllers, user interfaces, and functionality. The framework concept enables the cost-effective construction of the NIF software and provides the basis for long-term maintainability and upgrades.

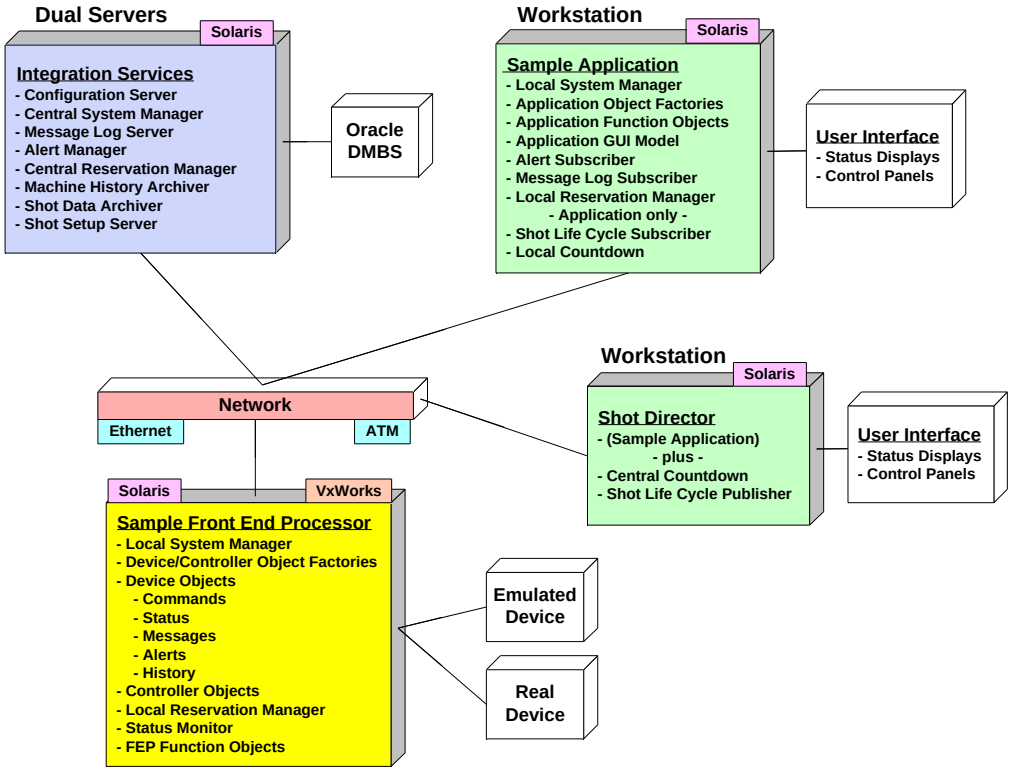


Figure 6 Deployment of ICCS framework objects into a sample application and FEP on networked computers

The following discussion introduces the framework components that form the basis of the ICCS software.

Configuration - a hierarchical organization for the static data that define the hardware control points that are accessible to the ICCS. Configuration provides a taxonomic system that is used as the key by which clients locate devices (and other

software services) on the CORBA bus. During normal operation, configuration provides to clients the CORBA references to all distributed objects. An important responsibility of configuration is the initialization of front-end processors during start-up. Configuration data are stored in the database and describe how and where the control hardware is installed in the system. Calibration data for sensors, setpoints for alignment devices, and I/O channels used by devices on interface boards are examples of static data managed by configuration. During ICCS start-up, this framework collaborates with an object factory located in the FEP. Using the data and methods stored in the configuration database, the object factory instantiates, initializes, and determines the CORBA reference for each device and controller object in the FEP.

Status Monitor - provides generalized services for broad-view operator display of device status information using the push model of event notification. The status monitor operates within the FEP observing devices and notifies other parts of the system when the status changes by a significant amount. Network messages are only generated when changes of interest occur.

Sequence Control Language - used to create custom scripting languages for the NIF applications. The service automates sequences of commands executed on the distributed control points or other software artifacts. Operators create and edit sequences by selecting icons that represent control constructs, Boolean functions, and user-supplied methods from a visual programming palette. The icons are then interconnected to program the sequence and any Boolean conditions or method arguments needed are defined to complete the sequence script.

Graphical User Interface - All human interaction with the ICCS will be via graphical user interfaces displayed upon control room consoles or on X Terminals distributed throughout the facility. The GUI is implemented as a framework in order to ensure consistency across the applications. Commercial GUI development tools are used to construct the display graphics. This framework consists of guidelines for look and feel as well as common graphical elements for beam selection, laser map, status summary, and countdown clock.

Message Log - provides event notification and archiving services to all subsystems or clients within the ICCS. A central server collects incoming messages and associated attributes from processes on the network, writes them to appropriate persistent stores, and also forwards copies to interested observers. The interested observers are primarily GUI windows on the screens of operators' consoles.

Alert System - any application encountering a situation that requires immediate attention raises an alert, which then requires interaction with an operator to proceed. The alert system records its transactions so that the data can be analyzed after the fact.

Reservation - manages access to devices by giving one client exclusive rights to control or otherwise alter the device. The framework uses a lock-and-key model. Reserved devices that are "locked" can only be manipulated if and when a client presents the "key".

System Manager - provides services essential for the integrated management of the ICCS network of hundreds of computers. This component ensures necessary processes and computers are operating and communicating. Services include parameterized system start-up, shutdown, and process watchdog monitoring.

Machine History - gathers information about the performance during operation of the NIF for analysis in order to improve efficiency and reliability. Examples of such information are installation and service of components, abnormal conditions, operating service time or usage count, periodic readings of sensors, and alignment reference images.

Generic FEP - pulls together the distributed aspects of the other frameworks (in particular the system manager, configuration, status monitor, and reservation frameworks) by adding unique classes for supporting device and controller interfacing. These classes are responsible for hooking in CORBA distribution as well as implementing the creation, initialization, and connection of device and I/O controller objects. The generic FEP also defines a common hardware basis including the target processor architecture, backplane, I/O boards, device drivers, and field bus support. The FEP application developer extends the base classes to incorporate specific functionality and state machine controls.

Shot Data Archive - The ICCS is not responsible for the permanent storage (archive) or in-depth study of shot data; it is however responsible for collecting the data from the diagnostics, making the data immediately available for "quick look" analysis, and delivering the data to an archive. The framework contains a server working in collaboration with the system manager to assure that requested shot data are delivered to a disk staging area. The archive server is responsible for building a table of contents file and then forwarding the table and all data files to the archive.

5 ICCS Software Development Environment

The ICCS incorporates Ada95, CORBA, and object-oriented techniques to enhance the openness of the architecture and portability of the software. C++ is supported for the production of graphical user interfaces and the integration of commercial software. Software development of an expected 500,000 lines of code is managed under an integrated software engineering process [Figure 7] that covers the entire life cycle of design, implementation, and maintenance. The object-oriented design is captured in the Rose design tool using the Unified Modeling Language that maintains schematic drawings of the software architecture.

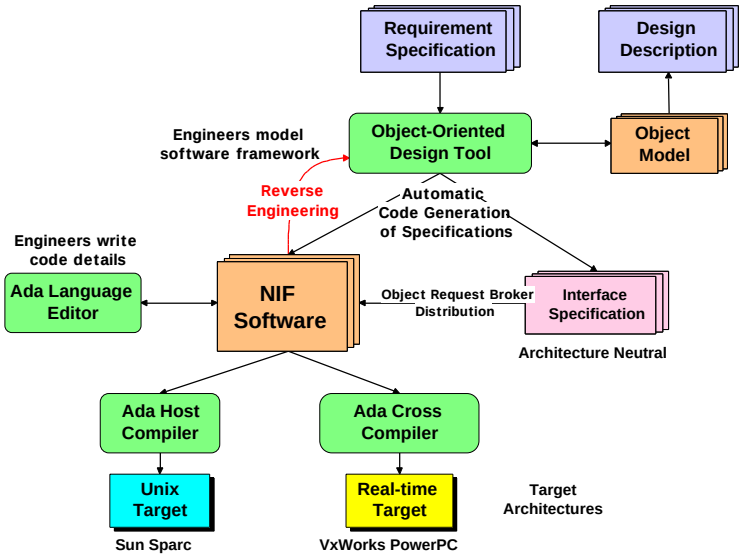


Figure 7 Flowchart of the ICCS software engineering process incorporates model-driven design techniques

Detailed requirements including use cases are analyzed by developers and result in classes defined to implement the responsibilities of the software. In general terms, the Rose tool is used to model the interfaces and interactions between major software entities. Rose automatically generates Ada95 code specifications corresponding to the class interface and type definitions. The developer fills in the detailed coding necessary to implement each class. Classes that are distributed generate IDL, which is passed through the IDL compiler to generate Ada skeleton code as before. The design description is a narrative document that explains the object-oriented model and contains other information necessary for implementation.

Ada source code can be compiled for a variety of target processors and operating systems. Current development is self-hosted to Solaris on Sparc or cross-compiled for VxWorks on PowerPC. The models, sources, binaries, and run-time images are version-controlled by the Apex configuration management system, which allows the frameworks and applications to be independently developed by different engineers, each having a protected view of the other components.

6 CORBA Distribution

Past architectural approaches to distributed controls have relied on the technique of building large application programming interface (API) libraries to give applications access to functions implemented throughout the architecture. This practice results in large numbers of interconnections that quickly increases the system complexity and make software modification much more difficult. To address this problem in the ICCS, software objects are distributed in a client-server architecture using CORBA.

CORBA is a standard developed by a consortium of major computer vendors to propel the dominance of distributed objects on local area networks and the worldwide web. The best way to think of CORBA is as the universal “software bus”. CORBA is a series of sophisticated, but standard sockets into which software objects can “plug and play” to interoperate with one another. Even when made by different vendors, at different times, the object interfaces are standard

enough to coexist and interoperate. By design, CORBA objects interact across different languages, operating systems, and networks.

At a greatly simplified level, the major parts of CORBA are shown in Figure 8. The interface types and methods provided by the Server Objects and used by the Clients are defined by an industry standard Interface Definition Language (IDL). The IDL compiler examines the interface specification and generates the necessary interface code and templates into which user-specific code is added. The code in the Client that makes use of CORBA objects is written as if the Server was locally available and directly callable -- CORBA takes care of all the rest.

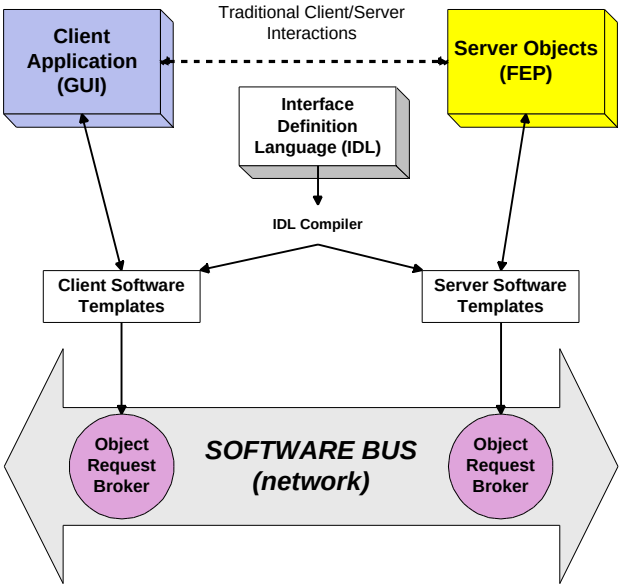


Figure 8 CORBA distribution implements a software bus on the network

Because CORBA handles the data format conversion necessary to interoperate with diverse computer systems, it is a more heavyweight protocol than previously used for control system distribution. Object request brokers have been measured and shown to perform about three times slower than point-to-point communication schemes (e.g. sockets). For this reason, we established a significant testing capability to predict the operational performance of multi-threaded CORBA (ORBexpress by Objective Interface Systems) under various deployment schemes.

Simulation results for different sizes of message streams are shown in Figure 9 for transactions between two UltraSparc machines (one was an Enterprise 3000 server and the other a 3D Creator workstation, both with 300 MHz processors). Measurements of CORBA on PowerPC/VxWorks are in progress, but performance is expected to be similar, as CPU performance is generally similar. Message rates for 100 Mb/s Ethernet are shown along with client and server CPU and network bandwidth utilization. An additional plot is shown for 10 Mb/s Ethernet, which is used to attach many of the FEPs.

We determined that most control system transactions utilize on the order of 100 byte messages. For this case, CORBA can transact at about 2700 messages per second while utilizing 80% of the client CPU and 30% of the server CPU. The reason for the wide difference in CPU utilization between client and server is not fully understood and is under investigation. For small message sizes, the CPU is the limiting resource, as the network is not heavily utilized (note that other computers are using the remaining network bandwidth). However, as messages become larger (e.g., during post-shot data retrieval) the network becomes the limiting factor. In the ICCS design, we have partitioned our subsystems such that the message rate design point will average about 500 control transactions per second. This approach provides a fivefold capacity margin to accommodate episodic bursts of message activity that are to be expected occasionally in an event driven architecture. The performance of the ICCS deployment is estimated by measuring software prototypes on our distributed computer testbed and scaled to the NIF operating regime by discrete event simulation techniques.

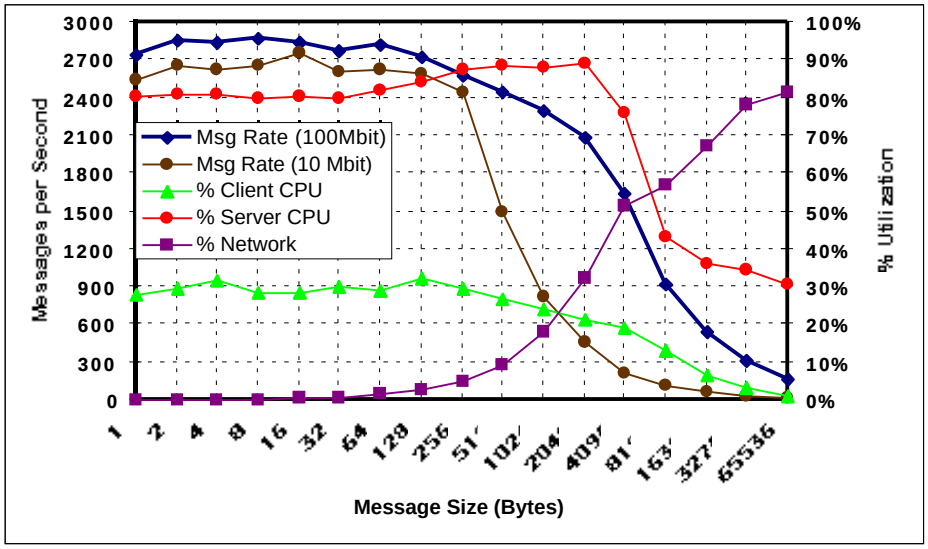


Figure 9 Performance measurements of CORBA

7 Summary

The ICCS is being developed using the iterative approach to software construction[3]. This technique is proven effective for projects whose requirements continue to evolve until late in the project development. Five iterations are planned prior to the first facility deployment of the ICCS software. Each new release will follow a plan aimed at addressing the greatest risks to the architecture while increasing the functionality delivered to the Project.

Early demonstrations have already confirmed the basic architecture and we are now constructing three prototype releases prior to delivery to NIF. The first release, which is being built during the summer of 1998, will deliver vertical slices of all applications to exercise the ICCS framework. Subsequent phases will be released to incorporate additional subsystem integration and automation during 1999 leading toward first deployment in the facility in the year 2000, when the first 8 of the 192 beams will be operated.

Construction of the ICCS incorporates many of the latest advances in distributed computer and object-oriented software technology. Primary goals of the design are to provide an open, extensible, and reliable architecture that can be maintained and upgraded for decades. Software engineering in Ada95 is a managed process utilizing model-driven design to enhance the product quality and minimize future maintenance costs. Simulation models are used to extend prototype measurements to NIF deployment scale to ensure that the fully deployed system will meet performance goals. The design permits software reuse and allows the system to be constructed within budget. As an added benefit, the framework design is sufficiently abstract to allow future control systems to take advantage of this work.

8 Acknowledgements

The authors wish to acknowledge the contributions of our colleagues without whose efforts this work would not be possible: G. Armstrong, R. Bryant, R. Carey, R. Claybourn, T. Dahlgren, F. Deadrick, R. Demaret, C. Estes, K. Fong, M. Gorvad, C. Karlsen, B. Kettering, R. Kyker, L. Lagin, G. Larkin, G. Michalak, P. McKay (Sandia National Laboratory, Albuquerque NM), M. Miller, V. Miller Kamm, C. Reynolds, R. Reed, W. Schaefer, J. Spann, E. Stout, W. Tapley, L. Van Atta, and S. West

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

9 References

1. J. A. Paisner and J. R. Murray, "The National Ignition Facility for Inertial Confinement Fusion"
2. V. Swaminathan and J. Storey, “Domain-Specific Frameworks”, Object Magazine, April 1988, pp. 53-57
3. B. Boehm, “A Spiral Model of Software Development and Enhancement”, IEEE Computer, May 1988, pp. 61-72

*Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

Technical Information Department • **Lawrence Livermore National Laboratory**
University of California • Livermore, California 94551