STATUS OF THE USE OF LARGE-SCALE CORBA-DISTRIBUTED SOFTWARE FRAMEWORK FOR NIF CONTROLS

R.W. Carey, R.C. Bettenhausen, C.M. Estes, J.M. Fisher, J.E. Krammen, L.J. Lagin, A.P. Ludwigsen, D.G. Mathisen, J.T. Matone, R.W. Patterson, C.A. Reynolds, R.J. Sanchez, E.A. Stout, J.D. Tappero, P.J. Van Arsdall
Lawrence Livermore National Laboratory, Livermore, CA 94550, USA

ABSTRACT

The Integrated Computer Control System (ICCS) for the National Ignition Facility (NIF) is based on a scalable software framework that will be distributed over some 750 Computers throughout the NIF. The framework provides templates and services at multiple levels of abstraction for the construction of software applications that communicate via CORBA (Common Object Request Broker Architecture). Object-oriented software design patterns are implemented as templates to be extended by application software. Developers extend the framework base classes to model the numerous physical control points. About 140 thousand software objects, each individually addressable through CORBA, will be active at full scale. Most of the objects have persistent state that is initialized at system start-up and stored in a database. Centralized server programs that implement events, alerts, reservations, message logging, data archive, name services, and process management provide additional framework services. A higher-level model-based, distributed shot automation framework also provides a flexible and scalable scripted framework for automatic sequencing of work-flow for control and monitoring of NIF shots. The ICCS software framework has allowed for efficient construction of a software system that supports a large number of distributed control points representing a complex control application. Status of the use of this framework during first experimental shot campaigns and initial commissioning and build-out of the laser facility is described.

INTRODUCTION

The ICCS [1] is a distributed, hierarchically organized, object-oriented control system that employs a framework [2] of reusable software to build uniform programs to satisfy numerous functional requirements. ICCS employs Ada95, Java, CORBA, and object-oriented techniques to enhance the openness of the architecture and portability of the software. Ada is generally used to implement control system semantics [3]. Java is used for the production of graphical user interfaces and the integration of commercial software, particularly the Oracle database system. CORBA provides the transparent language binding and distributed communication middleware utilizing TCP/IP transport.

This paper describes the ICCS distributed component architecture and discusses framework treatment of distribution complexity and connection management. In addition, three newly deployed frameworks are described including the diagnostic architecture framework, the shot automation framework, and the commissioning tools framework. The final section describes the evolution of the ICCS software framework based on early operational experience and refinements planned for the next major framework release.

DISTRIBUTED COMPONENT ARCHITECTURE

NIF is a largely repeating structure of 192 laser beams, each having almost identical sets of components. The NIF physical organization lends itself to a distributed, component-based communication architecture. Laser beams are aggregated into 8 beam sets called a “bundle”. There are 24 bundles in the NIF. This physical partitioning of the laser has been extended to the control system computer architecture. Control system processes and computers are organized by bundle to achieve better parallelism, performance, and reduce the impact of localized failures. This is referred to as “bundle based partitioning” and has no impact on framework or application layer software due to the location independent features of the CORBA based inter-process communication architecture. This ability to independently

organize the hardware architecture without impact to the software is one of the significant benefits of CORBA and the design of the ICCS software framework.

Following strategies of object-oriented software development, similar software components are defined as classes, and these classes are instantiated for each occurrence of a NIF component in each laser beam. Control components consist of various actuators, sensors, and instruments used to operate and diagnose each NIF laser beam and its interaction with a target. NIF physical control components are represented by named CORBA software objects in the ICCS. CORBA transparently maps the name to the software component object located in the processor that is connected to the physical hardware. This gives computer/network location transparency to the ICCS application software. The framework and application software combined have resulted in a design consisting of approximately 340 CORBA interface classes.

DISTRIBUTION COMPLEXITY

The distributed component model is an elegant and easy to understand representation of the physical NIF. The ICCS employs a layered, client-server computer architecture to control the NIF. Each NIF bundle is supported by ~25 front-end processors (FEPs) connected to various laser diagnostic and control components. An additional number of FEPs are allocated to control and monitor common components that are not bundle based, such as target area systems. A server class UNIX machine is assigned to run the supervisory and shot layer software for each bundle. The NIF currently has one bundle fully commissioned and one in various stages of activation. This computer architecture will be replicated for each NIF bundle (24x) as they are commissioned. Java GUIs are run on separate Win/PC based platforms in the control room and connect to supervisor and FEP software layers as needed. Laptops can also be connected to the network in the laser and target areas when needed. At full scale the ICCS will be distributed to over 750 computers containing over 1500 processes and 140 thousand individually addressable CORBA objects.

An analysis of CORBA object populations and TCP socket counts for each processor has guided design choices for network, memory, and CPU capacities. With the bundle based partitioning, CORBA object populations and TCP connections do not present significant risk to architectural scaling. Effective tools to organize and manage the large process population are currently being designed.

With such a large amount of distribution come challenges for building robust, resilient systems. No software application can be complete until error conditions are handled. A distributed system running on top of the TCP/IP transport layer has many more error possibilities than a monolithic, non-distributed system. The number of distributed interfaces correlates directly to system complexity. A high number of distributed interfaces allow possibly nondeterministic messaging behavior. The potential for race conditions and/or software deadlock in such an open communication environment is ever present. ICCS has developed standards for interface design that specify various communication decoupling mechanisms. These decoupling mechanisms are integral to the designs of the distributed software framework and provide examples/patterns that application interfaces can employ.

Debugging distributed systems is difficult at best. Stringent testing can expose many of distribution related software defects, and the ICCS software testing program has been largely successful at this endeavor [6]. The software testing program is a key element of the overall ICCS software engineering process and is, in general, essential for qualifying any distributed system.

CONNECTION MANAGEMENT

Highly distributed systems must deal with connection management. Individual systems might need to be restarted for any of several reasons. How clients deal with lost connections has a direct impact on system resistance to failure.

Several connection management behaviors are provided as part of the ICCS Software Framework including interface decoupling, object reconnection, subscription management, process state heartbeats, status health heartbeats, and timed invocation. This framework approach has been quite successful and covers a significant majority of cases for the detection, notification, and recovery from communication failures. This provides common

benefit to all ICCS application software. Below are brief descriptions of mechanisms employed to facilitate connection management in ICCS software. Many of these are implemented in various ICCS framework components.

Decoupled Communications – Client side

Clients can protect themselves by decoupling their communication to remote services. This technique is applicable to interface methods that return no immediate information to the client. The client leaves a message in a queue, buffer, or outgoing mailbox, and a separate task actually makes the call to the remote object. This avoids blocking in the client. It is used in the ICCS Status Monitor and Status Propagation frameworks for distributing status updates to clients. It is also used in the client side of the Alert, Event, Reservation, Archive, Subscription Management, and Message Log frameworks.

Decoupled Communications – Server side

In this technique, the method in the server has another task in the server object perform the work. Each new request can be delegated to a separate task or queue and task resource pools can be incorporated. This serves to de-couple the client from the actual work of performing the service. The Archive, Data Access, and Reservation Frameworks make use of this decoupling pattern.

Object Reconnection

This technique applies primarily to the situation where a server process is terminated, then restarted with the same objects. The first time any existing client makes a request to the new server, it will get communication failure due to the stale CORBA connection associated with the previous existence of the server process. The ICCS framework is constructed to re-establish the connection to the new server instance automatically when a new communication is initiated.

Subscription Management

The ICCS Subscription Management Framework provides subscription reconnection for all subscription services. The framework maintains process state subscriptions for all publishers that a client application utilizes. Knowledge of what processes provide publishing services is determined dynamically by the framework and requires no such knowledge in the application layers. If a publisher goes offline, the associated publisher information is marked stale in client processes. When the publisher state returns to health, the subscription management framework re-subscribes the client automatically which causes a refresh of the publisher information. The Subscription Management Framework supports automatic re-subscription for alerts, events, reservations, and status registrations.

Process Heartbeats

A typical way of monitoring the state of a remote object involves having a separate thread that periodically sends a heartbeat from the observed object to a monitoring object. The ICCS framework establishes and monitors periodic heartbeats for all application and framework server processes. This is built as part of the System Manager Framework and it is behavior that all ICCS processes inherit. Absence of a heartbeat indicates that some communication problem exists. The framework then periodically attempts to re-establish the heartbeat connection. The System Manager framework reports such error conditions on a central GUI representing the state of all ICCS processes. In addition, the System Manager Framework allows any client to subscribe to the state of any set of the process population. This feature is used extensively by the Subscription Management Framework.

Status Health Heartbeats

Early production use of ICCS uncovered failure modes where status information was not being refreshed and no indication of the stale status was provided to the operator. The ICCS Framework has since adopted a design for “status health heartbeats” to provide positive feedback for stale status. When a client registers for status updates, a status health heartbeat

is initiated between the status publisher and the subscribing status client. If the status health heartbeat does not arrive, the status is marked stale by the framework.

Timed Invocations

ICCS timed invocations use a variant of the de-coupled client side mechanism. A separate task makes the remote invocation. A different task then monitors the invoker task for some predetermined amount of time and if the invoker task does not return with a response before the timer expires, the monitoring task can proceed into error handling actions. The ICCS connection management framework has implemented timed invocation behavior as an extension to the client-side for both ORB express (Ada) and Visibroker (Java) CORBA ORBs.

DIAGNOSTIC ARCHITECTURE FRAMEWORK

The purpose of the ICCS diagnostic architecture framework is two-fold: at a "system-level", provide visibility of the network, processors, and processes; and, at a "process level", embed the ability to remotely query run-time statistics and configure objects within running ICCS framework servers and application processes. The diagnostic architecture utilizes "out-of-CORBA-band" communications, based on the User Datagram Protocol (UDP). This approach isolates the capturing of diagnostic information from the CORBA messaging behavior of the operational control system, thereby minimizing the intrusion upon the applications being diagnosed.

For the embedded process-level diagnostics and metrics, ICCS has developed framework-level components that can selectively capture performance details from specific framework objects. These framework components are embedded directly into the ICCS process architecture. This diagnostic information can be remotely activated and retrieved from runtime GUI-level views and stored for off-line analysis. The framework also supports addition of application layer diagnostic classes. Construction of new diagnostic classes at both framework and application layers is work in progress.

At the system level, ICCS is utilizing the Simple Network Management Protocol (SNMP) and SNMP processor agents. This aspect of the diagnostic architecture utilizes existing COTS products as much as possible. Specifically, the SNMP-based diagnostics provide information on the network connectivity, and individual computer resource utilization (I/O performance, memory, CPU utilization, etc.). The production environment has active SNMP agents for both UNIX and Win/PC hosts. SNMP agents for Power PC FEPs running VxWorks still need to be developed.

Collectively, the diagnostic architecture will help to collect and evaluate overall system performance, as well as provide valuable assistance in diagnosing failure-modes of the large-scale distributed component control system. The display, reporting, and analysis of SNMP events are work in progress.

SHOT AUTOMATION FRAMEWORK

Operation of the NIF for the NIF Early Light (NEL) campaigns successfully proved the NIF laser performance criteria with the first four lasers [4]. However, configuration and monitoring of the lasers for these shot campaigns was very operator intensive with human verifications of actuators and diagnostic configurations through low level device maintenance panels. This method of shot operations does not scale as the number of lasers increases, but did provided the essential requirements for concept of shot operations.

A significant requirements documentation and software design effort was initiated in November 2003 for a shot automation software layer [5]. This new layer meets the scaling requirements; adheres to the bundle partitioning; and provides a high degree of flexibility for configuring the lasers. The resulting ICCS Shot Automation Framework models (in data) the sequence and actions that operators use to conduct NIF shots. It is highly data driven with framework support for automated verification of device positions and calculation of device participation based on goals of the experiment. Using this new shot software layer, operation of the NIF is orchestrated by a workflow engine that enforces dependency ordered execution of a graphical representation of the steps in a NIF shot. It contains a Change Manger tool that

shot physicists and operators use to dynamically modify participation of laser components depending on the conditions of different diagnostics and laser segments.

The new shot layer was deployed to NIF in April 2005 and has been used successfully on the first full bundle (8 laser beams) experiments. Efforts are now underway to optimize the shot model to improve the shot rate and support additional laser beam destinations. The next shot framework release (Framework 5.0) will support increased parallelism, improve operator views of automated steps, and improve presentation of error conditions.

COMMISSIONING TOOLS FRAMEWORK

Commissioning of lasers for NIF has uncovered additional requirements for software tools that enable the efficient calibration/qualification of laser components. Such tools are essential to meet the schedule requirements for build-out of the remaining NIF lasers. As a result, a commissioning tool framework has been implemented in Java to support construction of software tools that interact with various devices in the running control system using existing CORBA interfaces. Based on requirements, collections of devices are manipulated in particular sequences to qualify them for normal operational usage. The resulting derived operational parameters are stored in a configuration database for use in integrated shot operations.

The Maintenance and Commissioning Tools (MCT) applications look and feel like normal ICCS GUIs to the end user. However, ICCS GUIs are generally stateless, depending on external servers to maintain higher-level state information. Additionally, their device control is generally limited to single commands. In contrast, MCT applications contain complex algorithms that require sending commands to a collection of devices, aggregating the resulting data, and then perform processing on the data. The MCT framework provides a mechanism for separating the displays from the algorithms via autonomous tasks. The framework facilitates proper threading of these tasks and provides callback functionality for error, status and data to the GUI display.

The initial deployment of the MCT applications contained the framework itself and three tools: Motor Backlash Measurement, Beam Rotation, and Waveplate Calibration. The Motor Backlash tool's algorithm uses image analysis to measure the slippage in the gears when changing the motor direction by tracking the movement of the center of the beam. The Beam Rotation tool also utilizes image analysis in its algorithm; it measures the horizontal and vertical rotation of the laser beam to ensure that it meets certain specifications. The Waveplate Calibration tool gathers data from photodiode energy sensors and plots the results to assist the operator in calibrating the waveplate motors. As of July 2005, a total of eight tools have been delivered to the NIF facility. Six more tools are scheduled to be delivered in the coming months.

SOFTWARE FRAMEWORK EVOLUTION

The ICCS Software Framework has had 4 major releases during the past 4+ years with many minor and service pack releases in between. The software framework was developed with an iterative process of requirements, design, and refinement. The application layers have had to adopt changes in the framework along the way. The first three framework releases were on tight 6 month development cycles culminating with the Framework 3.0 release in April 2002. Application layer software constructed with Framework 3.0 supported the NIF Early Light (NEL) activation and experimental campaigns through the summer of 2004.

The 3.0 release of the ICCS Framework focused on connection management. All of the connection management behaviors discussed in the section above were designed and developed for Framework 3.0. These connection management features represented a significant improvement in the resiliency of the ICCS to individual process failures and accomplish a large portion of the ICCS fault tolerant performance requirements.

Framework 4.0 was released for application integration in September 2003. Focused support for NEL shot campaigns delayed application layer adoption of framework 4.0 until May 2004. Framework 4.0 contained migration to new versions of most all of the commercial software tools used by the ICCS. This included major updates to configuration management systems, compilers, run-time environments, operating systems, database software, and CORBA ORBs. In addition, many refinements/enhancements were

incorporated as a result of the NEL experience. There was a substantial effort to improve the robustness of the framework servers and the framework templates used by application software. First iterations on the Shot Automation Framework and the Diagnostic Framework were incrementally added as a minor release with framework 4.1 in late 2004.

Framework 5.0 is scheduled for release to application integration in January 2006. New requirements to better support shot automation are being incorporated. Framework 5.0 will support the NIF Multi-Bundle milestone in June 2006. In general, it contains additional commissioning tools and refinements to the Alert, Reservation, and Shot Automation Frameworks. In addition, tools and techniques for improved process management, and operator session management are being incorporated in the GUI Framework.

SUMMARY

The ICCS is a distributed, hierarchically organized, object-oriented control system that employs a framework of reusable software to build uniform programs to satisfy numerous functional requirements. The framework design accommodates substantial distribution complexity and supports a significant number of control points. The software architecture and CORBA middleware allow for transparent re-factoring of the hardware architecture to meet performance requirements and minimize impact of localized hardware failures. The ICCS incorporates CORBA layer design extensions to provide robust connection management behaviors that meet fault tolerant performance requirements.

The ICCS software framework successfully supported application layer functionality in the NIF Early Light experimental campaigns that proved the NIF project performance criteria. Recent extensions to the framework include run-time diagnostics, data driven shot automation, and construction of laser commissioning tools to help meet the rapid construction schedule for the remaining lasers in NIF. The next major framework release is scheduled for January 2006. This framework and the associated application layer products position the control system for multi-bundle experiments. Construction of the remaining lasers through 2008 will lead the way for NIF's support of the National Ignition Campaign and thermonuclear ignition.

This work performed under the auspices of the U.S. DOE by LLNL under contract No. W-7405-Eng-48.

REFERENCES

[1] Arnie Heller, "Orchestrating the World's Most Powerful Laser," Science and Technology Review, Lawrence Livermore National Laboratory, UCRL-52000-05-7/8, July 2005.

[2] R. Carey, et al, "Large-Scale CORBA-Distributed Software Framework for NIF Controls", ICALEPCS 2001, October 2001.

[3] Robert W. Carey, Paul J. VanArsdall, John P. Woodruff, "The National Ignition Facility: Early Operational Experience with a large Ada Control System", ACM SIGAda Annual International Conference, UCRL-JC-150637, November 2002.

[4] P. Van Arsdall, et al, "Status of the National Ignition Facility and Control System", ICALPECS 2005, October 2005.

[5] L. Lagin, et al, "Shot Automation for the National Ignition Facility (NIF)", ICALPECS 2005, October 2005.

[6] D. Casavant, et al, "Status of Testing and Quality Assurance of the NIF Control System", ICALEPCS 2005, October 2005