Industry:
Government/Public Sector

Organization:
Lawrence Livermore Labs

Description:
Lawrence Livermore Labs ensures national security and applies science and technology to important problems of our time.

Business Problem:
To build the largest Inertial Confinement Fusion Laser facility, the LLNL team needed a development toolkit proven capable of managing large, object-oriented projects.

Rational Solution:
Rational Apex, Rational Rose

Key Benefits:

- Allowed leveraging of in-house Ada experience
- Enabled the development of robust, testable code segments with confidence
- Enhanced workers ability to work independently, then integrate well

High Technology Key to Building and Operating World's Largest Laser Fusion Facility

Inertial Confinement Fusion (ICF) involves focusing powerful laser beams on a tiny, BB-sized capsule containing deuterium and tritium (heavy isotopes of hydrogen) to trigger a fusion reaction through compression and heating. The fuel in the capsule ignites and burns like a miniature star, all within billionths of a second. During this time, detailed measurements can be made of the temperatures, pressures and many other physical phenomena that occur at these extreme conditions — providing scientists with a wealth of valuable information.

Experiments at extreme temperatures (100,000,000° C) and densities (20 times the density of lead) will provide the data needed to understand the effects of aging on nuclear weapons, helping the US Department of Energy (DOE) monitor and maintain the safety and reliability of the US stockpile in the wake of the ban on nuclear testing. Experiments in fusion energy are also hoped to provide the foundation for a new, environmentally attractive energy source. And the ability to re-create conditions existing inside the sun and stars will significantly impact the science of astrophysics and high energy density physics.

DOE Building World's Largest ICF Laser Facility

The potential of Inertial Confinement Fusion (ICF) is so great that the DOE approved and funded the construction of the National Ignition Facility (NIF), a football-stadium-sized research facility for the primary purpose of demonstrating fusion ignition. Located at the Lawrence Livermore National Laboratory (LLNL) in Livermore, California and involving the participation of four national science labs the NIF will cost $1.2 billion and take over seven years to build. It is expected to become fully operational some time in 2003.

The NIF will contain the world's largest laser, an array of 192 powerful beams that will run the length of the facility and be used to start the fusion process. The NIF laser will produce 2 megajoules of energy in a 25 nanosecond pulse and reach a power of approximately 500 trillion watts for 4 nanoseconds at a time — more than ten times the power and three times the duration of the Nova laser, the one used at present by LLNL for ICF research.

Each shot of the NIF, which will generate about 400 MB of data, will require aligning all components of the laser so that all 192 beams propagate down 600-foot paths, through their amplifiers, and into the target chamber within 50 microns of their assigned spot inside the centimeter-scale target. Additionally, all beams must arrive at the center simultaneously within 30 picoseconds. The alignment process involves approximately 9500 stepper motors, every one contributing in some way to the position of the beam. It will be carried out by an automated system that analyzes some 3000 distinct images, measures image offsets and commands motion of the stepper motors, making operator intervention necessary only when a control loop fails to stabilize.

LLNL is using Rational Rose/Ada to design and Rational Apex to develop the software for the NIF's ICCS, one of the largest distributed control systems for experimental equipment. This distributed control system for system operation consists of 354 computers: 35 UltraSPARC computers running 8 supervisory applications plus 264 PowerPC Vx Works computers and 55 UltraSPARC computers running 14 front-end processor applications.

LLNL Addresses Massive Challenges with Advanced Concepts, Technology

The NIF presented significant technical challenges to LLNL, even with the team's prior experience building the Nova laser in the early '80s and other large-scale scientific facilities in the 1970s:

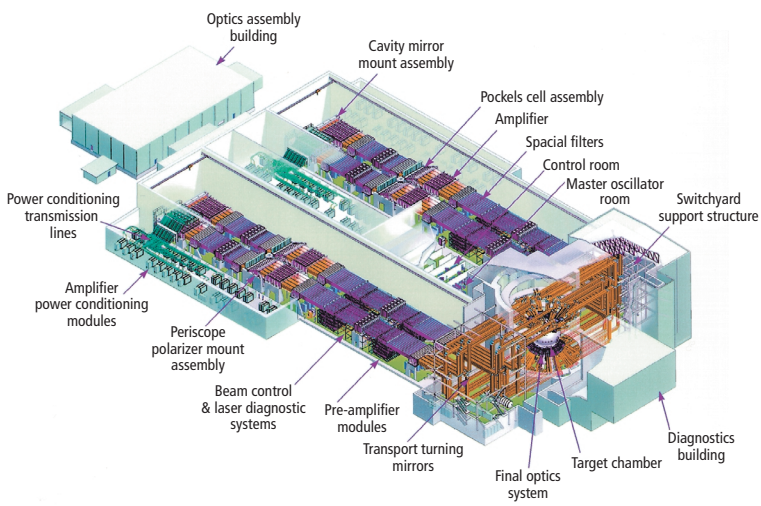
- The ICCS must provide the controls for a dozen engineers to operate a coordinated machine composed of 50,000 distinct control points
- The system must operate in a very large, highly distributed environment

"We set out to write down major structural components in Rose, capture interfaces between them as Ada specs, then put those specs under control in Rational Apex."

— John Woodruff

- Applications must support and be maintainable over the anticipated 30-year life of the NIF while allowing for the evolution of the experimental facility such as changes in control and diagnostic procedure or the addition of new computer technology
- The system must be highly automated and robust, able to run full-time (24 x 7 except for scheduled maintenance), with an allowed downtime of 7.5 days per year for unscheduled maintenance
- The system architecture must be flexible enough to absorb significant changes in requirements late in project construction
- The system must be delivered on time and on budget

Based on past experience, LLNL decided to address infrastructure risks by incorporating modularity, segmentation and open-systems standards. This allows components and subsystems to be replaced at designated interface points if necessary. Interoperability among computers and operating systems is being addressed by leveraging the international standard Common Object Request Broker Architecture (CORBA). And to address the significant cost, reliability and maintenance issues of a control system of this magnitude, a model-based, object-oriented approach to the design and development of the ICCS software was chosen.



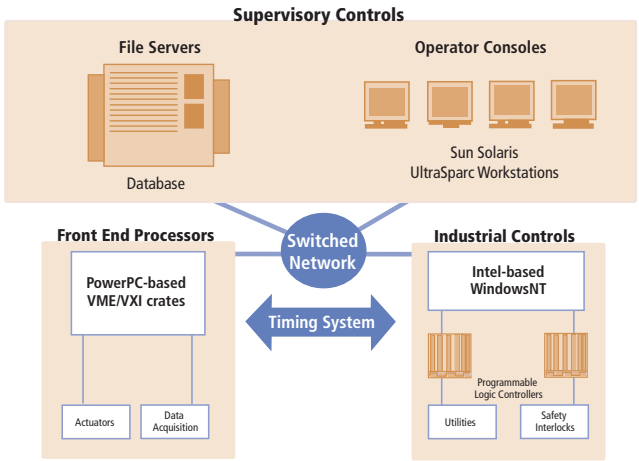
"The object-oriented paradigm has two important properties, encapsulation and inheritance, that increase the reliability and lower the cost of software products," explained John Woodruff, lead software architect at LLNL. "The NIF design exploits these properties." By taking a modular approach to NIF software development, individual programmers can perform their duties without interfering with their colleagues' work. The team considers this feature of the NIF strategy a significant improvement over the Nova experience and expects it to contribute to ease of integration when the system is delivered.

Object-oriented modules also significantly ease the effort and cost of maintenance on a system of this magnitude. "Experience has shown that software systems developed from functional specifications tend to be difficult to modify after they are deployed," explained Woodruff. "This is because the feature that needs to be changed is usually not localized in a single module; instead, the functional modules depend on data that might be stored in data structures that are not themselves well modularized. With our approach, modules in the finished software product encapsulate both the data and the functional behavior of the object; the software system is easier to maintain because maintenance operations tend to touch a smaller collection of modules."

Team Chooses Ada 95, Rational for Its Development Toolkit

The LLNL team needed tools proven capable of managing large object-oriented projects. An Ada shop since building the Nova laser, the arrival of Ada 95 with object-oriented features made it the obvious choice of programming languages for the NIF's control system software development. The team selected Rational Rose for its modeling ability to design and generate Ada components, along with Rational Apex for its Ada 95 compiler, runtime capabilities and its configuration management and version control (CMVC) system. ORBexpress2 provides the CORBA middleware.

"We feel strongly that a carefully engineered programming language will let us build robust, testable pieces that we can use and be confident of when it comes time to integrate them," said Woodruff. "Using other paradigms doesn't give you tight



specification to do integration. We've demonstrated for ourselves, at least through prototype, that by using configuration control, modeling, specifications and interface design, we can enable different workers to work independently, then integrate well."

"We set out to write down major structural components in Rose, capture interfaces between them as Ada specs, then put those specs under control in Rational Apex. This is important as more people begin to add code."

Engineers Adopt New Approach with NIF: Start With an Architecture

"The challenges of the NIF are significantly different from those of the Nova laser, built when there were no GUI development tools, no fast networks, less capable computers, and when the concept of software architecture hadn't caught the attention of engineers," noted Paul Van Arsdall, Deputy Systems Engineer for NIF Controls, who was instrumental in the Nova controls project. "The primary focus at that time was on the computers and their capabilities. We had to make sure that engineers could run the job on the computers they had. We had to build a network. This time, we started with an architecture."

The primary goal of the ICCS design was to provide an open, extensible and reliable architecture for the NIF that can be maintained and upgraded for decades, permits software reuse across multiple applications, and allows the system to be constructed within budget. Engineers also hoped to develop an architecture abstract enough to be reusable in similar projects in the future. To accomplish this, they observed the architecture of other control systems, seeking as many abstractions as possible that could be used for an event-driven, distributed control system like the NIF. They proceeded to identify the general functions found in similar systems, and to provide solutions independent of individual pieces of the laser. The product architecture is a loose coupling of these solutions. The reusability goal will be met if future products can apply these solutions.

The Solution: A Two-Layered Model

The ICCS architecture was created with two layers: a control system consisting of front-end processors (FEPs), and a supervisory system.

Most FEP's run under VxWorks on PowerPC processors. The supervisory system layer is hosted on Workstations running Solaris UNIX. Communication between and within layers is handled using CORBA messaging.

The supervisory system layer provides operator controls and status, data retrieval, processing and archiving, and integration services. It is partitioned into several cohesive subsystems, or "frameworks," representing the specific abstract components that solve different parts of the control problem:

Configuration – a hierarchical organization for the static data that defines the hardware control points accessible to the ICCS, also responsible for initializing the FEPs during startup

Status Monitor – provides generalized services for broad-view operator display of device status information using the push model of event notification

Sequence Control Language – creates custom scripting languages for the NIF application

"We've generated several classes of device that are quite different from each other, all built on the same architecture."

— Rita Bettenhausen

Graphical User Interface – enables all human interaction with the ICCS, via GUIs displayed upon control room consoles or on terminals distributed throughout the facility

Message Log – provides event notification and archiving services to all subsystems or clients within the ICCS

Reservations – manages access to devices by giving one client exclusive rights to control or otherwise alter the device

System Manager – provides services essential for the integrated management of the ICCS network of hundreds of computers; ensures that necessary processes and computers are operating and communicating

Machine History Archive – collects data about the performance and operation of the NIF, originating within the ICCS, for analysis of the NIF operation in order to improve efficiency and reliability

Generic Front End Processor – pulls together the distributed aspects of the other frameworks by adding unique classes for supporting device and controller interfacing; also defines a common hardware basis

Shot Data Archive – collects the 400 MB data generated from the diagnostics, makes it immediately available for "quick look" analysis, and delivers it to an archive.

The subsystems are integrated to allow operators to coordinate operation of laser and target area equipment from a centralized control room. This capability is critical since most operations require coordination of multiple components that may be hundreds of feet from each other and served by different front-end processors. For example, an operator can command one front-end processor to move the beam path on a laser beam, and command another processor to manipulate a video camera so he can monitor the process — all with a few clicks of a mouse from a single control panel.

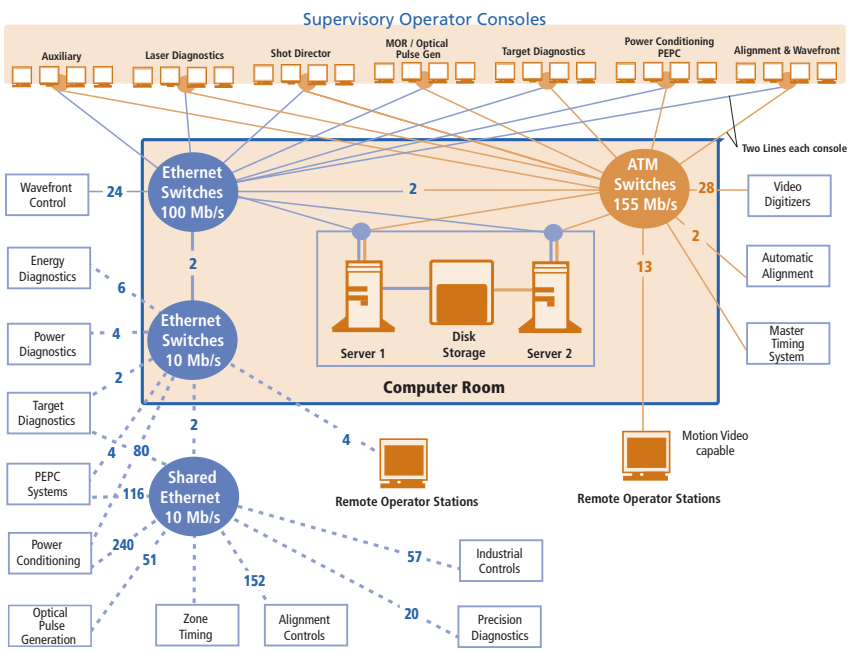
The front-end processor software performs sequencing, data acquisition and reduction, and coordinated device control. ORBexpress distribution control is used to integrate the FEP units into a single cohesive system.

The framework strategy is reusable for any highly flexible, event-driven control application. "Our goal in using frameworks was to have a software substrate independent of the NIF so that we can offer it to another experimental facility and they could particularize it to their devices to do what they want," said Woodruff. "We knew that the laser we're building now is not the laser that will run for 30 years. This framework will last, even though the hardware will evolve. As long as we have the tools to generate code based on standards such as CORBA for interoperability, SQL for the database, and GUI tools, we can use this over time and for different facilities."

Software is being built from the bottom of the framework up, starting with those elements least likely to change and allowing for expected changes late in the schedule to the thin GUI layer at the top of the control system.

Software Development Process Yields Operational Prototype

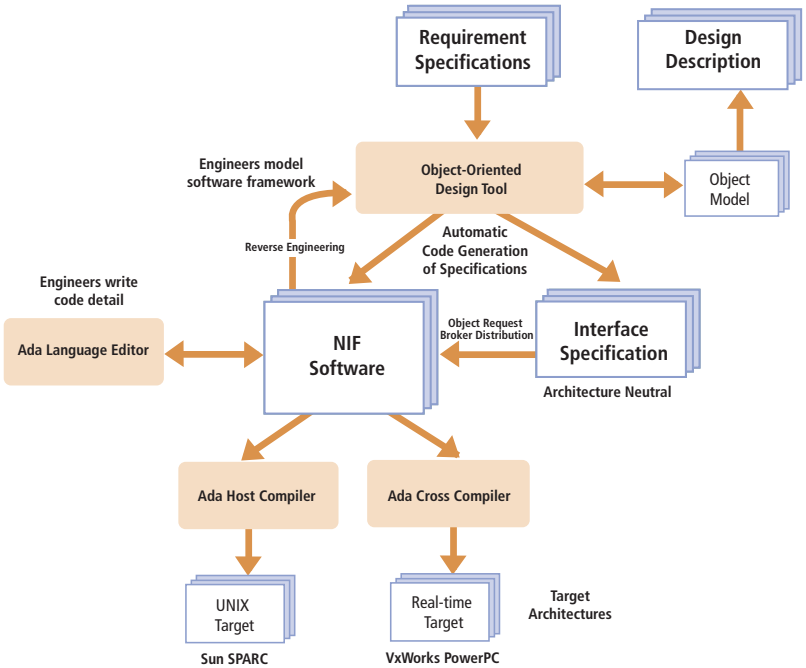
For each of the major supervisory control applications, (e.g., Alignment, Power Conditioning and Laser Beam Diagnostic), LLNL developers use Rational Rose to define classes that implement the software requirements. Important software use



cases are expressed in terms of object scenario diagrams. In general terms, Rational Rose is used to model the interfaces and interactions between the major software entities in the design. After the modeling cycle is complete, the Rational Rose tool generates the Ada specifications corresponding to the class interface and type definitions. The developer fills in the detailed coding necessary to implement internal details of each class. Classes that are distributed generate interface definition language (IDL), which is passed through the IDL compiler to generate Ada code.

The Ada source code then takes one of two paths depending on the target processor architecture. For Solaris, the source is compiled, linked, debugged and tested on the workstation using the Apex Ada 95 compiler. For PowerPC or other VxWorks targets, the source is compiled and debugged under the Rational Apex Embedded cross-compilation system. In either case, the models, sources, binaries and run-time images are managed by the Apex configuration management system, which provides full capability for version control. Using Apex subsystem facilities, the frameworks can be independently developed by different engineers, each of which has a protected view of the other components.

"Our first real triumph was building the front-end processor interface and modeling the things that go into it to make a generic front-end processor," said Rita Bettenhausen, the front-end processor engineer who devised this generic solution that will be used system wide. "We can punch out instances of classes from the model, generate it into code, and end up with code that fulfills its function. This is important, with 14 different kinds of front-end processors, all with different properties. For example, we've modeled the stepper motor (device), captured the abstraction of the device and how it acts, extended it with subclasses of the device, and shown how to generate code for the new class, such as a new kind of motor. An engineer can then go into the model, find the class she or he wants, and specialize it. We've generated several classes of device that are quite different from each other, all built on the same architecture. We're satisfied that all you've got to do is define the particulars of the device you want, and you get a running front-end processor from it."



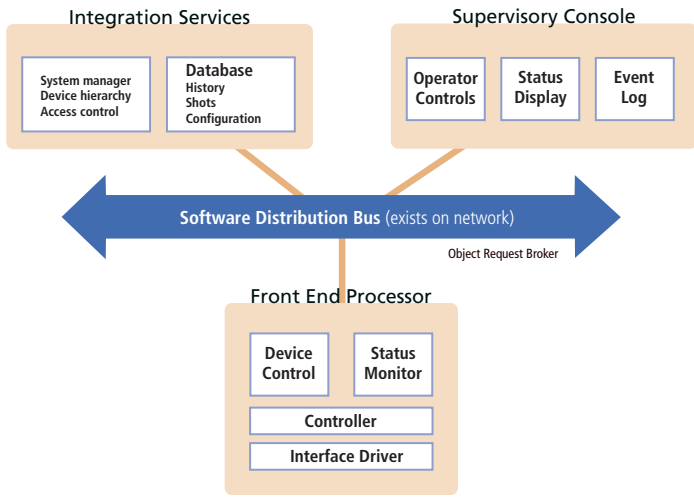
Leading Edge Technology Venture Paying Off

Construction of the ICCS incorporates many of the latest advances in computer and software technology — and represented some significant risk taking early in the project. In fact, the team needed to convince management of the credibility of their approach. "We adopted a fairly aggressive position of the software we were going to use, of adopting object-oriented technology, and especially the use of CORBA," explained Woodruff. "We have aggressively relied on tools and abstractions and notions of reusability because we expect to complete the system with a highly focused small team of developers. By building frameworks that we can use everywhere, we can solve each problem once and make an effective system solution without needing to write 400,000 lines of detailed code. If our abstractions are complete and right, they are the answer to development cost problems."

Although final code and algorithmic efficiency will not be realized for three years, the team has already successfully demonstrated distribution over ORBexpress, reusable frameworks, control of three of the planned 14 front-end processors, Oracle8 DBMS, and several architectural concepts that will be reused for the entire system.

"Modules in the finished software product encapsulate both the data and the functional behavior of the object; the system is easier to maintain because maintenance operations tend to touch a smaller collection of modules."

— John Woodruff



"We have the tools and will stress them — Rational Rose to generate and maintain consistent models and to generate the Ada source code, and Rational Apex to hold that source code and compile," Woodruff said. "We'll also rely on the infrastructure (ORBexpress, the database and GUI tools) to work at the scale of this industrial-strength programming. Our team of ten software engineers will be able to implement the entire supervisory control layer on project schedule. But we not only have a schedule, we have a fair pile of code and we must be able to integrate all the applications. We've built facilities like this before and we have a reputation to protect, and a reputation for doing things on budget. With this project, we want to maintain our reputation and do "big science" that works — and for the amount of money that we've been allotted. So far, we've done well."

The analysis and design of the control applications with Rational Rose, which included two prototype iterations, was completed in two years. Software development will be iterative over a three-year period, with each application being developed independently over a six- to eight-month period and then demonstrated. When the first eight laser beams are ready to be tested, the software components will be integrated, rendering 1/24th of the NIF laser fully operational and usable for physics experiments. Remaining bundles of the laser will be installed monthly over a two-year period, with the facility scheduled to be completed at the end of 2003.

Future Significance of NIF is Global

Successful implementation of the NIF will enable physicists at LLNL to make significant contributions for decades to national security, energy and science. It will also increase the high-technology capabilities and international competitiveness of a number of industrial partners including firms specializing in precision optics, laser and electro-optics technologies, high-speed instrumentation, micro-fabrication and advanced imaging. And it is spawning numerous spin-off technologies including advanced manufacturing techniques, extreme ultraviolet lithography for making more powerful computer chips, and micropower impulse radar for a variety of near-term commercial applications.

About Rational Software Corporation:

Rational Software Corporation® (Nasdaq: RATL), the e-development company, helps organizations develop and deploy software for e-business, infrastructure, and devices and embedded systems through a combination of tools, services and software engineering best practices. Rational's e-development solution helps organizations overcome the software development paradox by accelerating time to market while improving quality. Rational's integrated solution simplifies the process of acquiring, deploying and supporting a comprehensive software development platform, reducing total cost of ownership. IDC has recognized Rational as the revenue leader in multiple application development and deployment markets for four years in a row. Founded in 1981, Rational, one of the world's largest software companies, had revenues of $754 million in its twelve months ended December, 2000 and employs more than 4,000 people around the world. Rational is a component of the Nasdaq-100 Index®. Additional information is available on the Internet at www.rational.com

Rational®
the e-development company™

Dual Headquarters:
Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Toll-free: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com

International Locations:
www.rational.com/worldwide

CS137 Rev.02/01 *Subject to change without notice.*