Final Report
Reporting Period: 01-June-95 to 31-December-97

LEGOS: Object-Based Software Components for Mission-Critical Systems

Sponsored by:
DOD Technology Reutilization Program
Issued by Department of Energy, Grant DE-FG03-95SF20715

Joint Venture Lead:
I-Kinetics, Inc.
Phone: 781.270.1300
Seventeen New England Executive Park
Burlington MA 01803

Transforming Legacy to ComponentWare

Table of Contents

DISCLAIMER

Acknowledgments

The I-Kinetics team wishes to thank the DOD TRP program for the funding that has helped support the research and development of this effort. In our opinion, the program fulfills a vital need for small-business funding of critical high-risk R&D. This type of long-term development capital for high-risk technology R&D is essentially not available from the private sector.

This document represents the Final Technical Report for Grant DE-FG03-95SF20715. Portions of this document were adapted from previous I-Kinetics work-in-progress papers, specifications and technical descriptions.

A substantial amount of the contract effort utilized the intellectual property base of I-Kinetics ComponentWare®, DataBroker®, and ComponentFactory® products and technology. These are commercial products that were developed previous to, or during the period of this contract.

1. Summary

An estimated 85% of the installed base of software is a custom application with a production quantity of one. In practice, almost 100% of military software systems are custom software [IK89, IK90, IK92, IK95a,b]. Paradoxically, the marginal costs of producing additional units are near zero. So why hasn't the software market, a market with high design costs and low productions costs evolved like other similar custom widget industries, such as automobiles and hardware chips?

The military software industry seems immune to market pressures that have motivated a multilevel supply chain structure in other widget industries: design cost recovery, improve quality through specialization, and enable rapid assembly from purchased components. The primary barrier is technical: we (as an industry) do not know how to independently produce reusable software components (component-ware). Some of the symptoms of this are:

- systems are designed and built from scratch as monolithic structures without benefit of purchased components or identifiable reusable parts.
- Each vertical-market supplier and system integrator must possess every horizontal software technology specialty in-house.
- Despite a high degree of shared low-level functionality across broad classes of applications, a viable (pervasive) component-ware industry does not exist to exploit it.

The primary goal of the ComponentWare Consortium (CWC) technology plan was to overcome barriers to building and deploying mission-critical information systems by using verified, reusable software components (ComponentWare®). The adoption of the ComponentWare infrastructure is predicated upon a critical mass of the leading platform vendors' inevitable adoption of adopting emerging, object-based, distributed computing frameworks - initially CORBA and COM/OLE.

Object-based distributed computing frameworks, such as CORBA and COM/OLE, are generally classified as distributed object management systems (DOMS). The CWC technology plan leverages emerging DOMS and advances them by developing software components and services required for demanding, domain-specific information and data management systems. The CWC currently recognizes three levels of participation. *Strategic Partners*, such as NAVSEA, Air Force, Army, Pratt&Whitney, supplied CWC with system integration projects in such areas as: space mission telemetry data management and analysis, turbine engine design and engineering, logistics management and capital market risk management. *Technology Providers* created the components and the underlying infrastructure. *Solution Providers* will plan and implement component-based solutions.

Each CWC Technology Provider and Solution Provider contributed and integrate key component software technologies. IONA, the current CORBA market leader, extended their CORBA product, Orbix(R), with high-performance transaction management, object groups and fault-tolerant services. NetLinks Technologies built a new generation of ORBitize™, a comprehensive DOMS Interactive Development Environment. Heuristicrats Research researched a range of security and data analysis Components. BBN Inc., researched supplying collaborative planning components as well as high-speed Internet connectivity services. I-Kinetics advanced research and development of their component software such as the DataBroker®, ObjectPump® and Automated Component Generator (ComponentFactory®). The ObjectPump transforms at run-time a legacy application or data source into a Component. The Automated Component Generator (ACG) is specifically targeted at existing software that has little or no DOMS capabilities. The ACG automates the generation of a complete Component from existing code.

The CWC technology plan also includes a ComponentWare system engineering methodology called the Information Factory Methodology. This methodology continued to be developed and was matured into a commercial offering call the ComponentFirst™ methodology and Virtual Application Warehouse reference architecture. Rather than utilizing a single monolithic application or database, the and Virtual Application is assembled on demand from a montage of legacy applications and data sources that have been encapsulated as components.

The key success criteria of the Virtual Application Warehouse and ComponentFirst Methodology is how well it supports new applications and changing user requirements and achieves reliability through the reuse of verified components. Data analysis and modeling systems are ideal domains to challenge these innovations. Application data is evolving beyond traditional numerical data to encompass a wide array of graphical, audio, signal, and image data types. A key capability is the run-time (dynamic) generation of components for legacy data and applications. Another is compile-time (static) generation components for legacy codes and applications.

Each CWC Strategic Partner utilized the Virtual Application Warehouse and ComponentFirst Methodology in deploying their systems over a minimum period of two years. These systems have demanding requirements: adapt quickly to new usage profiles, require a minimum of staff training, operate reliably in diverse environments. Each Strategic Partner domain presented a wide range of test and validation environments for driving successive generations of ComponentWare technology. Strategic Partners found that that component software resulted in significant reductions in the cost of system testing and requirements validation for their demanding systems.

The ComponentFirst Methodology made significant progress an approach towards answering how to manage the development process when component reuse is important.

- The key technology of the ComponentFactory is the ability to assemble new applications on demand from legacy applications and data sources. Once the

ComponentFactory is in-place, custom systems can be assembled from components located anywhere on the Internet.

- There is a big gap between announcements of UML OO CASE tools features and actual implementations of those features. The good news is that the UML CASE tools are fairly mature in their support of UML. However, the much harder problem, tool interoperability through standardization on one repository schema (metadata) model is yet to be delivered by the market.

The long-range goal of this work is to build and deploy military systems from verified, reusable architectures. The promise of component-based applications is to enable developers to "snap together" new applications by mixing and matching prefabricated software components.

A key result of this effort is the concept of reusable software architectures. Software architectures enable component reuse. A programmer cannot develop a component to meet reuse requirements unless those requirements are defined. A software architecture, which describes the mechanisms by which components interact and gives common patterns of component interaction, provides such requirements.

A second important contribution is the notion that a software architecture is something that can be captured in a formal language and reused across multiple applications. The formalization and reuse of software architectures provide major cost and schedule improvements. The Unified Modeling Language (UML) is fast becoming the industry standard for object-oriented analysis and design notation for object-based systems. In this effort, UML was used to prototype the reference architecture of military weapon system.

However, the lack of a standard real-time distributed object operating system, lack of a standard Computer-Aided Software Environment (CASE) tool notation and lack of a standard CASE tool repository has limited the realization of component software.

Our approach to fulfilling this need is the software component factory innovation. The factory approach takes advantage of emerging standards such as UML, CORBA, Java and the Internet. The key technical innovation of the software component factory is the ability to assemble and test new system configurations as well as assemble new tools on demand from existing tools and architecture design repositories.

The key approach was to transform legacy data and applications into components. These components could then be combined in a domain specific basis to military software system architecture design efforts

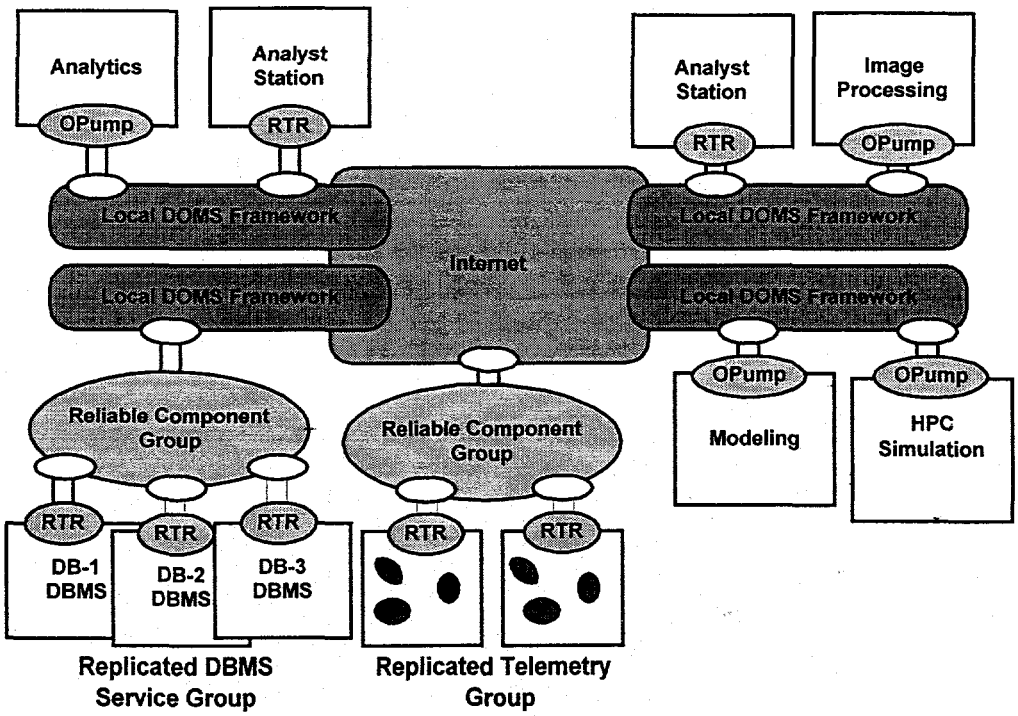


Figure E.1 **The Virtual Application Warehouse** enables information systems to increase their performance and functionality by dynamically integrating other application and data components. The end-user (analyst) can access data and applications either locally or remotely through the Internet. The Virtual Application Warehouse runtime environment consists of specialized components, ObjectPump's Reliable Component Groups and the distributed object management system (DOMS) run-time environment. An ObjectPump (OPump) is a general purpose adapter for creating components dynamically by encapsulation of legacy software code. Both Read-to-Run Components and the ObjectPump act as mediators between the DOMS services and a native application module's specific control and data protocols. A Reliable Component Group is implemented by a group of replicated components. In this scenario, a reliable database service is implemented using three DBMS Components and a reliable telemetry service is implemented using redundant telemetry feeds. Five different geographically distributed sites are shown connected by both wide area and local area routers. Each site has its own local DOMS framework, applications and services.

2. Technical Results

2.1 Component Software Technology RoadMap

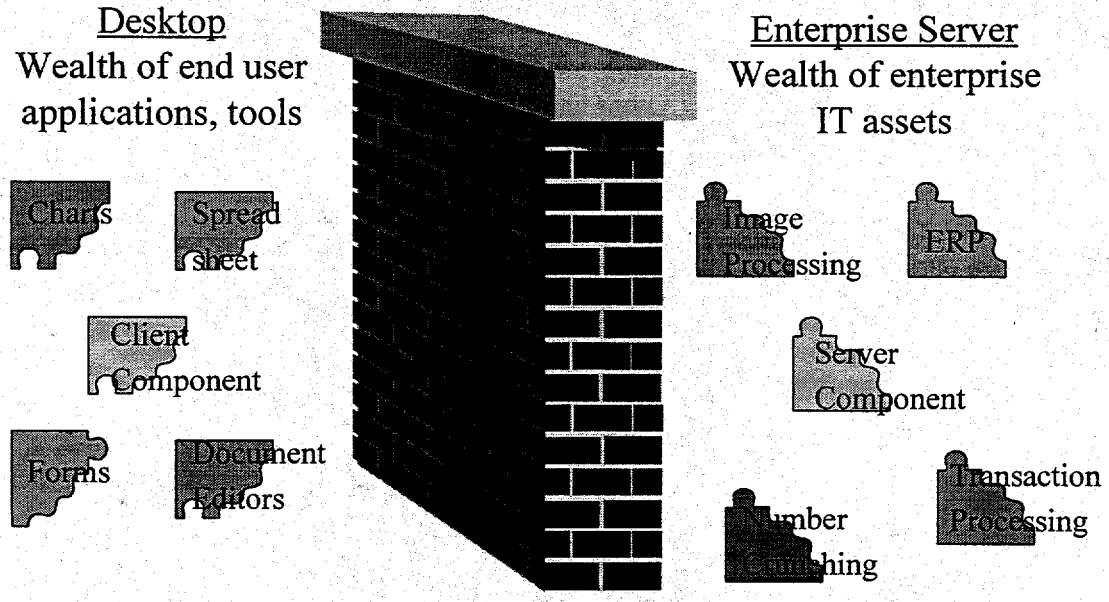
The Need for Component Software

Increasing change in the enterprise systems results in increasing re-integration of current capability with new capability. The enterprise can not eliminate integration. The information technologist has an unquenchable need to integrate faster, better and cheaper end-user personal computing with enterprise computing.

- The complexity of applications and data are increasing rapidly.
- Increasing availability and computing power requires distributed systems.
- The enterprise can not throw valuable systems away. New systems must evolve by adding new capabilities into the existing mix of legacy systems.

The integration of different applications on different platforms is very difficult. While applications offer integration interfaces, these application interfaces and data formats are naturally designed for a particular "sandbox" – a specific need with a particular initial set of operational requirements. Each application's sandbox introduces integration barriers of:

- languages,
- operating systems,
- network protocols,
- proprietary and standard application programming interfaces,
- application development tools.



Different languages, operating systems, network protocols, development tools, and interfaces create the "Barrier" to the integration of enterprise applications with desktop applications.

The result is many different interfaces resulting in many "square peg in round hole" integration barriers. Enterprise systems appear as valuable but inflexible applications that do not work with "new" systems. The major defining behavior of a "legacy system" is that cost of ownership and putting achieved quality at risk inhibit evolution of functionality and capability. Some of the "unfavorable" characteristics of the enterprise legacy investment are:

- Implemented in languages and OS that are no longer broadly supported
- Difficult to hire, retain and train staff to support
- Designed for standalone operation – no support for a network-based distributed system.
- Not available for latest and greatest OS, language or desktop or server applications.

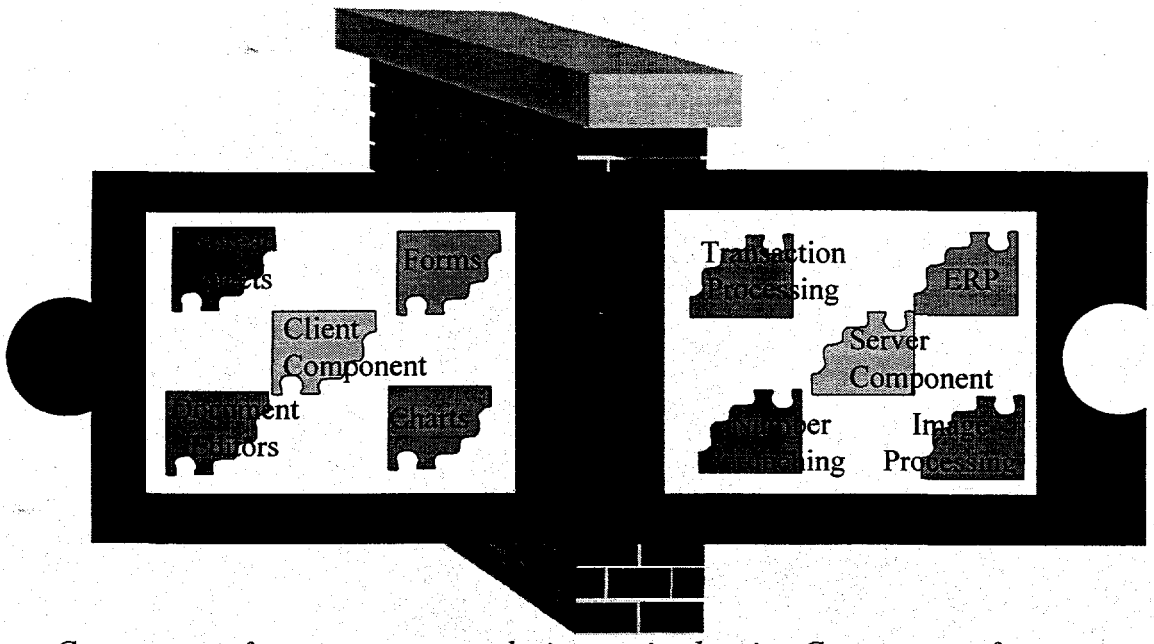
Component Software: "Faster, Better, Cheaper" Integration

The enterprise needs new capabilities for existing production applications at a faster rate and at a lower cost. New applications must integrate with existing applications into a pervasive infrastructure used both internally and outside the organization by its partners, customers, and suppliers. To meet these demands, enterprise systems are rapidly shifting to a new class of application – component-based applications.

The promise of component-based applications is to enable developers to "snap together" applications by mixing and matching prefabricated "plug&play" software components. This promise has been realized for the desktop application, the leading examples being drag&drop grids, charts, figures, images, forms and the "office suites" of spreadsheets, word processing and presentation.

Applications "age well" when built using components. The developer maintains and improves the application by replacing or upgrading components. Examples are:

- supporting more users accessing a critical data source by adding another database server;
- improve manufacturing floor scheduling by upgrading to a better non-linear optimization component;
- upgrading an sales order component with high-performance data access and multi-threading.



Component software crosses over the integration barrier. Component software is universally accessible and interoperates across boundaries of operating systems, networks, languages, development tools, and interface styles. Developers "snap together" applications by mixing and matching prefabricated software components.

However, enterprise component-based applications have yet to emerge. Enterprise application components place additional demands on the enterprise information system infrastructure. Enterprise application components are more difficult than desktop application components. Enterprise components have two additional challenges: they must be able to perform under high-user load and they must integrate with the existing enterprise information system investment.

What are the Basic Capabilities of Enterprise Component Software?

I-Kinetics has identified six basic capabilities required by enterprise application component software:

1. **Environment Independence**: Components can interoperate across languages, operating systems, networks, development or production environments.
2. **Location Transparency**: You can transparently access a component from within a single process, multiple process running within the same machine, or multiple processes running across networks and operating systems. From an architecture or implementation view, you are one level above transport, server location, process activation, datatype representation on different platforms.
3. **Interface separate from Implementation**: The interface is a contract that specifies a component's functionality. The interface exposes no implementation or platform or environment dependencies.
4. **Self-describing Interface**: A component provides the ability to describe the interface specification. At a minimum the specification must include method and property specifications. This is an essential feature for component software to be able to bind together at run-time – an essential capability for reusability.

5. **Platform Binary Independence:** The ability to install and use a component out of the box on any platform.

6. **Universal Application Component Framework:** A component must be able to integrate with one or more components to form a new well-behaved and well-defined application.

However, component software needs a ubiquitous set of rules of engagement across different component interaction boundaries – a distributed object management framework.

We live in fortunate times, because there is a widely adopted and deployed specification, CORBA®, that supplies the infrastructure capabilities on which we can build component software.

CORBA Supplies the First Four Capabilities, but what is CORBA?

In the fall of 1990, OMG first published the Object Management Architecture Guide (OMA Guide). The Common Object Request Broker Architecture (CORBA) 1.0 specification soon followed in 1992. The OMG is currently composed of over 700+ members and as an organizational body that owns and advances the OMA, CORBA 2.0 and CORBAservices specifications. CORBA has quickly become pervasively available on every leading desktop and server platform in an extraordinarily short period of time with the commitment of such industry leaders as Borland, IBM, IONA, Novell, Netscape, Oracle, and Sun.

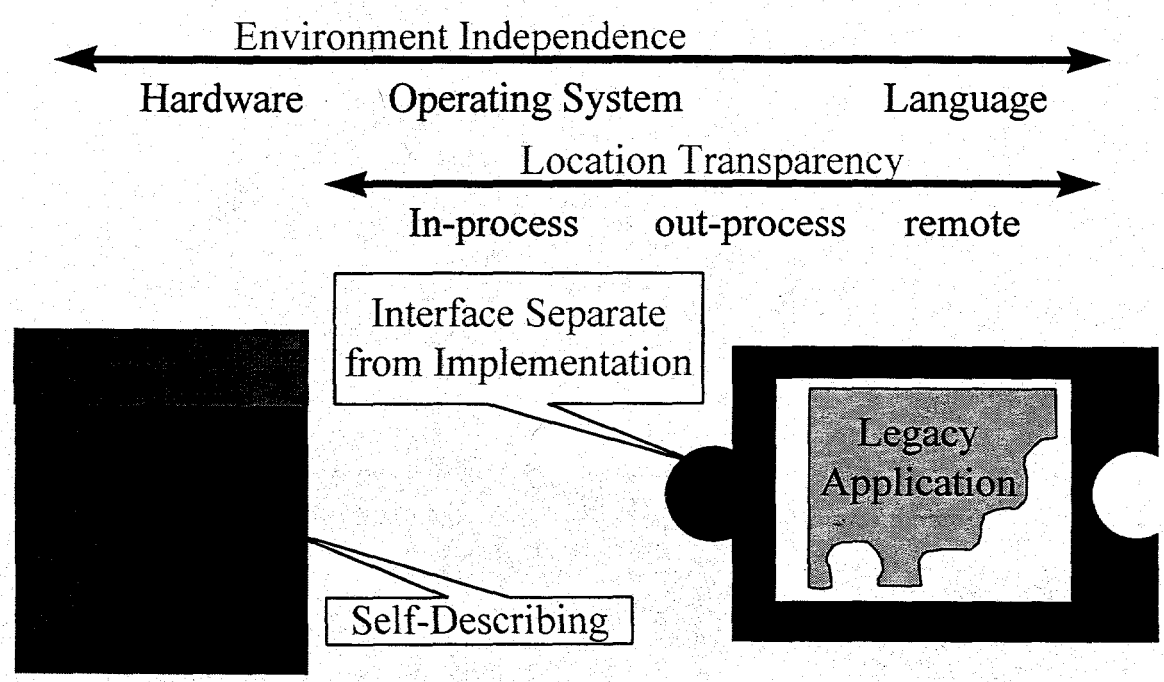
CORBA represents a paradigm shift in middleware – a pervasive, industry-wide agreement on distributed object interaction protocols and framework services. A client/server application migrated to CORBA becomes an assembly of collaborating components. CORBA's distributed object technology allows us to put together complex client server systems by assembling and extending objects. Objects may be modified without disturbing the rest of the objects in the system or how they interact. This is the fundamental value of object-based distributed computing the flexibility to execute the component in its "native" habitat, while accessing it from a one where it does not currently reside.

A key part of the CORBA specification is the Interface Definition Language (IDL). IDL is the CORBA standard language for defining an object's interface. CORBA uses IDL to specify an object's boundaries. Some key features of IDL are:

- provides operation system and programming language independent interfaces to all the services and clients that reside on a CORBA distributed object bus.
- every CORBA object must have a IDL specification and the component's IDL must be available from the Interface Repository.
- all CORBAservices and CORBAfacilities interfaces are defined in IDL.
- designed to be development language independent. IDL-specified object methods are written in and invoked from any language that provides CORBA bindings –currently Java, C, C++, SmallTalk and ADA are available. Limited implementations for COBOL are currently in the definition process.

- all clients of an object have a consistent view of that object, regardless of the environment of the client or the server object, or the location of client or the server object.

CORBA is a distributed object management framework for component software. The CORBA framework represents a ubiquitous set of inter-component communication protocols and services. CORBA satisfies the first 4 capabilities on which we can build component software. CORBA-based component software is universally accessible and interoperates across boundaries of operating systems, networks, languages, development tools, and interface styles.



The Common Object Request Broker Architecture (CORBA) supplies the first 4 requirements for component software: (1) Environment Independence, (2) Location Transparency, (3) Interface Separate from Implementation, (4) Self-describing.

Platform Binary Independence – The Fifth Component Software Capability

Initially, we defined Plug&Play platform binary independence as the ability to install and use a component out of the box on any platform. Component software has achieved a great amount of Plug&Play by incorporating CORBA run-time services. However, we defined our "six" component software capabilities so that there would be "NO BARRIERS" to integrating legacy systems with new enterprise capabilities. Let's raise the stakes by examining two critical needs for platform binary independence, that if satisfied, will enable universal Plug&Play for our component software.

(1) Pervasive inter-object communication protocol. CORBA is available for the desktop (Windows, Macintosh) and enterprise systems (OS/390, OS/400, UNIX, VAX/VMS). However, Plug&Play requires NO BARRIERS. There will always be new

platforms and new operating systems. An ubiquitous inter-object communication protocol, the core capability of a CORBA framework, is needed that can be easily ported to any new environment that comes along.

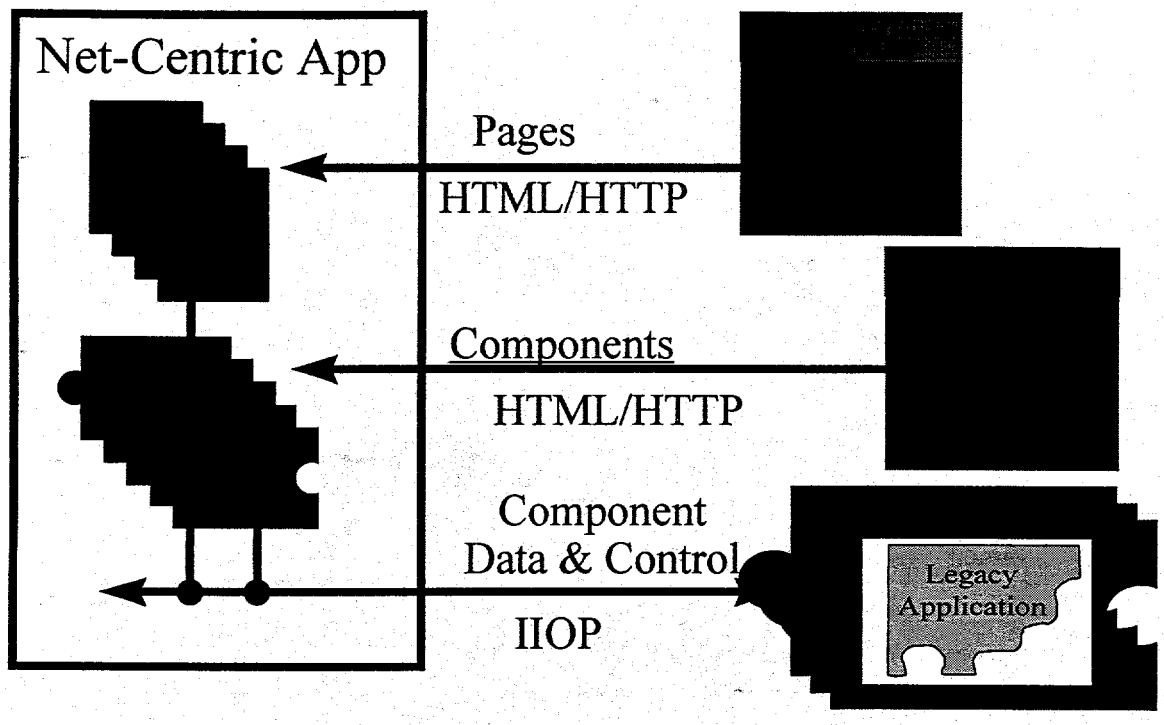
Solution: The Internet Inter-ORB Protocol (IIOP) is a defacto standard for ubiquitous inter-object communication protocol using TCP/IP. IIOP was originally motivated to standardize interoperability between different CORBA vendor implementations. IIOP has been rapidly adopted because it defines a standard set of message contents, formats, and semantics for transmitting requests and replies between objects. Given the specification of an object's methods and data structures, it manages all the packing and unpacking of the method call and object data structures. IIOP is a separable CORBA run-time service and is fast, reliable, and has a very small footprint.

Leading CORBA vendors, such as IBM, IONA, Sun and Borland have shipping IIOP implementations. By providing free versions of Java IIOP implementations, Sun, Netscape and IONA have taken a leadership role in providing a high performance, open standard, inter-object communication protocol for the Intranet and Internet distributed application.

(2) Platform Binary Independence. The ability to access a legacy application on its native platform is only the first level of Plug&Play capability. Full Plug&Play is realized when the component can be developed once and then installed on any platform. Additionally, platform binary independence results in the ability to relocate a component at run-time from one host to another. Performance, reliability, and replication are all factors that motivate component relocation in a large distributed system. Migrating components requires that the executable binary be machine independent.

Solution: Binary independence is achievable with a pervasively available language and virtual machine, such as Java.

- **Java the Language**: Java is based on a simplified set of the C++, the dominant object-oriented language. Java has refined C++ with the pivotal additions of platform independence and an integrated security subsystem. Unlike C++, Java supports only single inheritance. Java has features to deal with naming, security, garbage collection and performance.
- **Java the Platform**: But Java isn't just a language. It's also a portable operating system. The Java virtual machine enables a language to be loaded and interpreted or compiled. The Java virtual machine insulates an implementation from the native operating system and hardware.. The Java Platform consists of Java Core Classes, the Java Virtual Machine and the Porting Interface. A Java program can run on any computer that supports the Java runtime, with the downloaded code being checked automatically by the Java security subsystem. Java enables pervasive binary independence as every major operating system for the desktop (MS Windows, Macintosh, UNIX), Windows NT and UNIX server and the mainframe has or will have the Java virtual machine as part of the basic operating system configuration. The JavaOS was designed for use in devices such as network computers and smart cards by porting the JavaOS to microprocessors (JavaChips).



The hyper-rapid deployment of the net-centric application within the enterprise is driven by the ease with which multi-tier enterprise information systems are built using the open-standards of HTML, IIOP, CORBA and Java. HTML pages or Java components can be loaded into the Java-enabled desktop browser using HTTP. Any enterprise application component can be accessed using CORBA IIOP from Web pages, Java, or Web Servers.

Just some of the platform binary independent capabilities we realize from CORBA, IIOP and Java are:

- The high costs and risks of proprietary middleware and network protocols are eliminated. The use of open standards, such as IIOP, CORBA and Java protect against being locked into a single platform or vendor solution. CORBA and IIOP are pervasively available on every leading desktop and server platform. Continual support and advancement of IIOP is insured by industry-wide adoption. Key areas of advancement in the IIOP specification are firewall security, compression, and network management.
- CORBA and Java enable enterprise-wide system component reusability, increasing the organization's ability to assemble new applications rapidly from existing services. You can use existing infrastructure investments in desktop computers, servers, mainframes, databases, applications, and networks.
- You can manage distribution of applications components from a central, verified source rather than requiring costly updates to desktops or highly mobile portables.

Replacing Middleware with Component Software

> "..every piece of the software developer food chain is up for grabs. The whole paradigm for developing software needs to be rebuilt... client/server applications of the early '90s are now legacy." Ann Winblad, Partner, Hummer Winblad Venture Capital. As quoted in Computer Reseller News, April, 1997.

The emergence Java and CORBA delivers a solid infrastructure foundation on which enterprise component software can arise. Substantial return on investment (ROI) within months of deployment of intranets have drastically altered enterprise Information Technology strategies and application vendor product plans -- globally. 95% of the Fortune 1000 have deployed at least one Intranet. 40% had a budget of $1 Million or greater for intranet projects in 1997. The significant trends that will increasingly impact and shape the world's enterprise software market for the next 10 years are:

- **De-facto Standardization of Middleware:** Middleware can be loosely described as the software to glue a new application with the existing installed base of software. The client/server era saw the birth of a vast range of different types of middleware. Standard middleware will catalyze enterprise IT. The use of standard middleware eliminates the high costs and risk of being locked into a single platform or vendor solution. The Intranet platform and software vendors are rapidly converging on two middleware choices: OMG's CORBA and Microsoft's ActiveX/COM. The Common Object Request Broker Architecture ("CORBA") has quickly become pervasively available on every leading desktop and server platform. When asked in 1995, 14% of a group of Fortune 1000 IT managers surveyed indicated that they would deploy CORBA in 24 months. One year later, in 1996, this had grown to 40%. In 1997, Java and CORBA became intimately associated as the solution for building networked business applications and commercial services. Estimates vary from 60% (Gartner'97) to 85% (Zona'97) on the Fortune 1000 who have started funded Java/CORBA projects. Starting in 1998 and beyond it will become increasingly

difficult to sell applications that are not CORBA and/or COM capable. *All current client-server middleware will be replaced.*

- **New Legacy Investment:** As enterprise information technology (IT) moves to standardized middleware (CORBA/COM) and the Internet, they must bring their installed base of production applications and data. Applications achieve "legacy" status if they are in production and can not be pulled back because of user demand. New systems must evolve by integrating with the existing mix of Windows NT, UNIX and mainframe legacy systems. *The world's enterprise multi-$Trillion IT investment must be transformed to distributed objects to remain competitive.*

Competitive organizations are developing a new class of distributed applications using the Internet. The driving forces are: vertical integration of services across vendors to be more competitive, extending new services and support to customers to be more competitive, and the extension of new business processes across the corporation. Some example applications are: customer service, interactive on-line banking, global securities trading applications, real-time transportation and tracking systems and telecommunication services management.

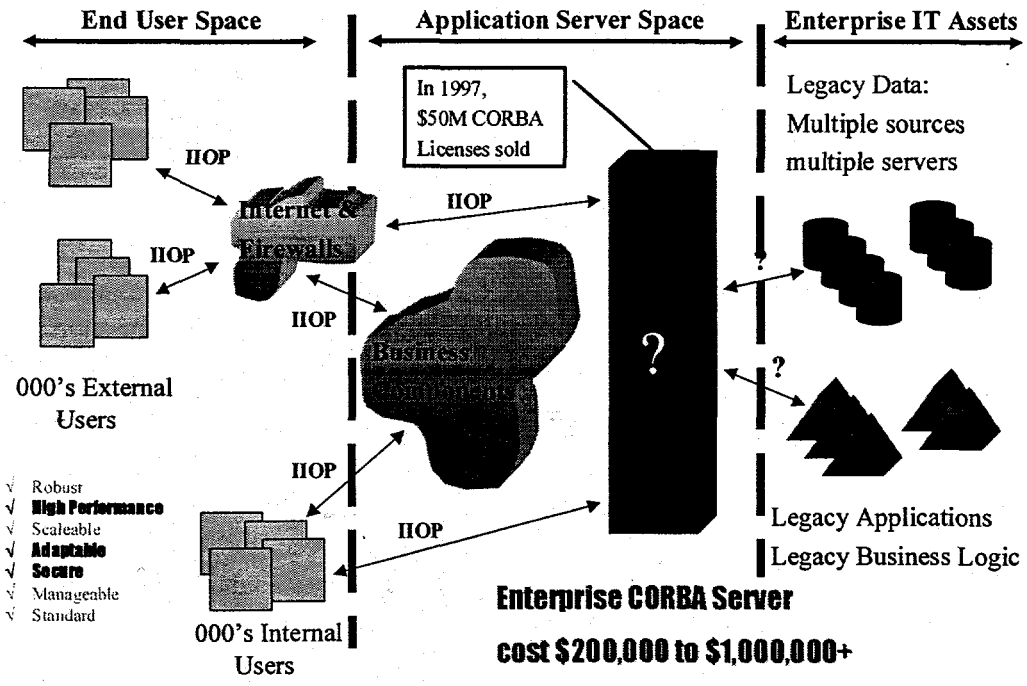
Over the years, companies have acquired a vast amount of in house developed applications that represent user, departmental, and enterprise business knowledge. Information Technology (IT) staff have developed a skill set in building these systems. The existing investment in applications and IT staff represent a vast resource of business and knowledge capital that an enterprise must reuse.

However, the sizable investment required to migrate existing IT to a distributed infrastructure of standard middleware and components is a formidable barrier. The most difficult to migrate of the IT investment is the "In-House" developed legacy application and legacy data. These applications and data are characterized by a high cost of ownership and a limited, even brittle ability to incorporate distributed computing capabilities.

A CORBA developer license consists of a CORBA compiler and run-time libraries. A CORBA developer manually codes CORBA servers and clients. The development of CORBA clients is relatively easy. The development of CORBA servers is very difficult. Developing enterprise quality CORBA servers is very very difficult and expensive. Transforming a legacy application into a CORBA server is the most expensive and risky development of them all. *Enterprise server applications must have "mainframe" capabilities: 7x24 reliability, high-performance, scalability, security, and be manageable. The lack of any these enterprise capabilities will prevent production rollout (i.e. project failure).*

Early success stories show that the cost of developing a CORBA server starts at $200,000 (mostly developer labor cost). Examples include wrapping a relational DBMS call interface, such as Oracle, a math library or a 3270 terminal emulation application. First generation CORBA servers usually achieve high-performance and extensibility (because they have the source code). Adding 7x24 robustness, scalability, security and

manageability escalates cost rapidly. For example, second generation CORBA servers in production at Swiss Bank, Wells Fargo, Boeing, Chevron, Fidelity, and Bell South had initial development costs of $1,000,000+ per CORBA server wrapper.

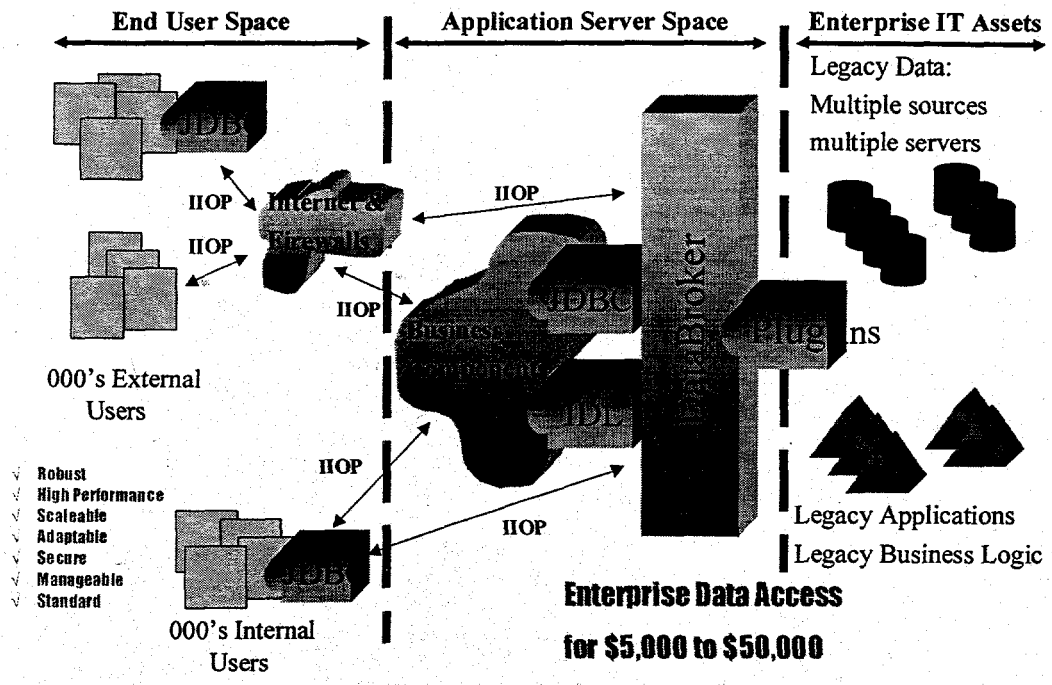


Transforming legacy applications into an enterprise capable CORBA server is very risky and expensive.

In 1994, I-Kinetics began development of the DataBroker -- the component solution to the (at the time predicted) high cost of developing CORBA servers for enterprise data access. With the November, 1997 release of DataBroker 5.0 Enterprise Edition, the enterprise buyer can transform, with an installation, existing relational databases, such as Oracle and Sybase, into enterprise CORBA servers for a starting cost of $5,000. Customers include Army, AT&T, Sprint, Bell Atlantic, Hughs, Navy, Pratt & Whitney, Siemens AG, and TRW.

OPENjdbc is the JDBC driver component of DataBroker and is a pure Java client of the DataBroker IDL. Like DataBroker, OPENjdbc was CORBA based from the beginning. Rather than using a proprietary protocol, OPENjdbc would use CORBA IIOP to talk to DataBroker and thus leverage all of the benefits of CORBA. Among other things, this validated that CORBA is an excellent technology for rapid development, partly because it allows for separation of implementation issues into discrete distributed objects that can then be developed in parallel. It was also because of the way that we leveraged CORBA in every possible way inside both DataBroker and OPENjdbc.

Today we are realizing the fruits of investing in CORBA earlier with such features as a complete component-based architecture, multi-threading, load-balancing and the ability to quickly add CORBAservices such as Naming, Security and Events.



DataBroker delivers enterprise data access for CORBA at a starting cost of $5,000.

DataBroker meets all of the requirements for enterprise data access.

- **Robust:** DataBroker is a rigorously designed and engineered 7x24 solution, based on the industry-leading ORB. DataBroker isolates dependencies on third-party technology so that failure of one data source can't affect another. DataBroker allows the ORB to be used to implement high-availability configurations.
- **High performance:** DataBroker is designed for high performance, both in simple usage scenarios and in complex heavily loaded enterprise systems. DataBroker uses the ORB to optimize the transfer of data and to implement more sophisticated performance enhancements such as caching.
- **Scalable:** DataBroker uses a sophisticated per-session multi-threading implementation that scales and is capable of taking full advantage of the capacity of SMP hardware. DataBroker allows load balancing to be implemented using the ORB across multiple DataBroker components on multiple platforms. DataBroker enables connection pooling and query caching to be directly implemented using the capabilities of CORBA.
- **Adaptable:** DataBroker has a component-based architecture, which allows plug-in components to be specialized to suit data sources and to be implemented on the appropriate platform and in the necessary language. DataBroker clients may be written in Java using OPENjdbc on any platform that supports Java, or in any language supported by CORBA on any platform supported by CORBA. DataBroker allows for native drivers for multiple de facto APIs for client-side development. DataBroker supports the use of native drivers for de facto APIs as gateways to different data sources on the server.

- **Secure:** DataBroker can be used in conjunction with a full spectrum of standard security solutions supported by the ORB. This includes HTTP tunneling, secure IIOP proxies for authentication, and secure IIOP for wire encryption over SSL.
- **Manageable:** Because DataBroker is a CORBA-based component, it can be deployed and managed using standard ORB vendor management tools as well as de facto systems management tools based on SNMP. The component-based architecture of DataBroker maps effortlessly into network management.
- **Standard:** DataBroker fully exploits both de facto and formal standards, as detailed earlier.

Universal Application Component Frameworks

> "By 1999, component software will be the dominant method of new application development", Roy Schulte, Gartner Group, 1997

Enterprise component software answers to the economic need to buy rather than build fundamental services, such as data access and transactions. DataBroker is a front-runner of the type of enterprise components that you will be able to buy for building your distributed applications. It is sufficiently compelling that it is replacing the practice of using proprietary middleware to integrate your new systems with your legacy systems. However, enterprise components has even more to offer -- the universal application component framework.

Component frameworks will complete the promise of being able to *assemble* different components to build your back-end enterprise business applications. Component frameworks specify how two or more components can be integrated together.

For example, a component service such as DataBroker can be used by tools that understand the JDBC interface. These are tools that enable you to visually specify the mapping between the database relational model to your Java objects. In this manner, tools such as JavaBlend, PowerJ, or JBuilder can automatically generate SQL and Java code that directly binds to the DataBroker JDBC interface. But what if your business logic calls for both efficiently accessing large amounts of data, analyzing the data and publishing the results to thousands of subscribers? For this business component, you would want to combine JDBC with a messaging service such as JMI or CORBAevents. To accomplish this you would need a tool that fully supports both JDBC and JMI and has the design patterns to combine the two different services.

But it is very difficult for tools to support all the different component services you might want to use. Instead, what if components could describe their interfaces to a tool? More importantly, what if a component could describe to a tool how its interface should be used and also what should be forbidden usage? Then the tool could be able to support any component, because the component would educate the tool on how it is used. For example, Visual Basic displays a OLE component's properties in the property sheet. A OLE component knows how to describe all its properties to Visual Basic because both support the OLE component framework.

The role of the component framework is to specify a standard model of how components describe themselves. As a result, you are able to use visual tools to assemble different components together to build new components. Let's take a look at a successful desktop component framework, OLE/COM and then two complementary component frameworks for the enterprise system: Enterprise JavaBeans and CORBABeans.

OLE/COM has achieved success as a desktop component framework because of its ability to "drag and drop" document components such as charts, figures, sound, and spreadsheet tables between word-processing and spreadsheet desktop applications. ActiveX is the successor to OLE.

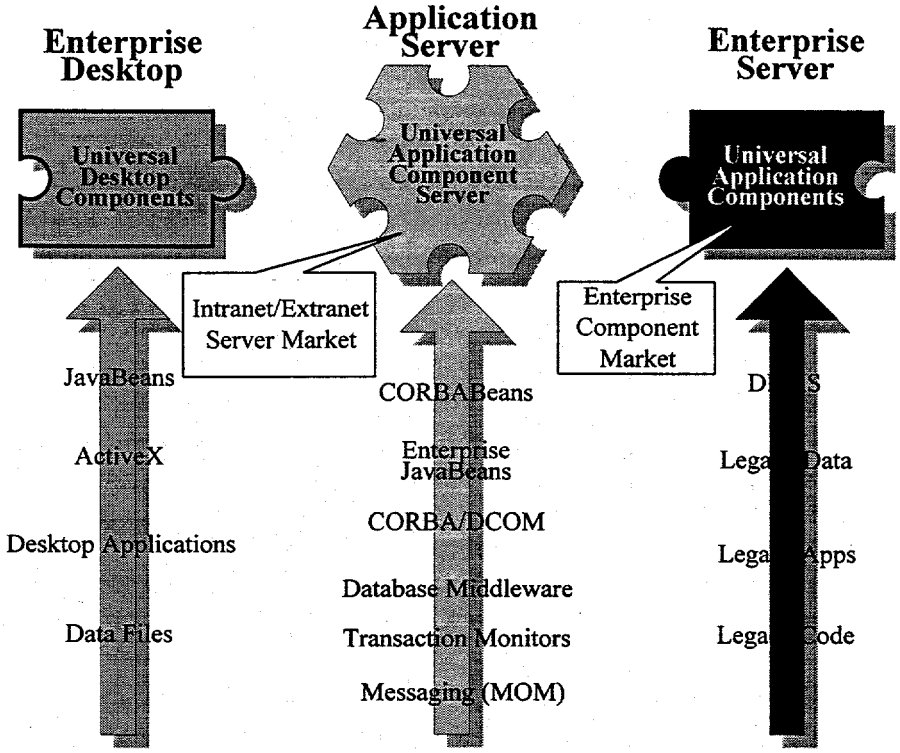
JavaSoft's JavaBeans is a "Pure Java" component framework that is competing with ActiveX. Fortunately, JavaBeans has been designed to co-exist with ActiveX while taking advantage of the "Write Once, Run Anywhere" benefits of Java. Any ActiveX component is JavaBeans capable and JavaBeans components are "out of the box" ActiveX compliant. A wide selection of tools and applications from Borland (JBuilder), Microsoft (Visual J++), Netscape(Communicator), Powersoft (PowerJ), IBM (VisualEdge and CBToolkit), Sun (Java Workshop/Studio) and Symantec(Café) can seamlessly incorporate JavaBeans or ActiveX components.

Wide adoption of ActiveX and JavaBeans component frameworks and the rapid merger of the Web browser and desktop applications is giving birth to a vigorous first generation of ***Universal Desktop Components.***

The "plug&play" complement to the Universal Desktop Component is the Universal Application Component. However, the equivalent component framework for enterprise applications is needed: a universal component framework and universal middleware. We already have the universal middleware -- CORBA.

The leading candidates for enterprise application component frameworks are Enterprise JavaBeans (EJB) and CORBABeans. Given that CORBABeans was born from the JavaBeans effort, you can expect them to only differ by their approach to object interface specification. EJB will be a "pure Java" component model while CORBABeans will be a mapping of EJB from the Java object specification to the language-neutral IDL object interface specification model. EJB will in turn drawn from CORBA all of its strengths as a distributed object foundation.

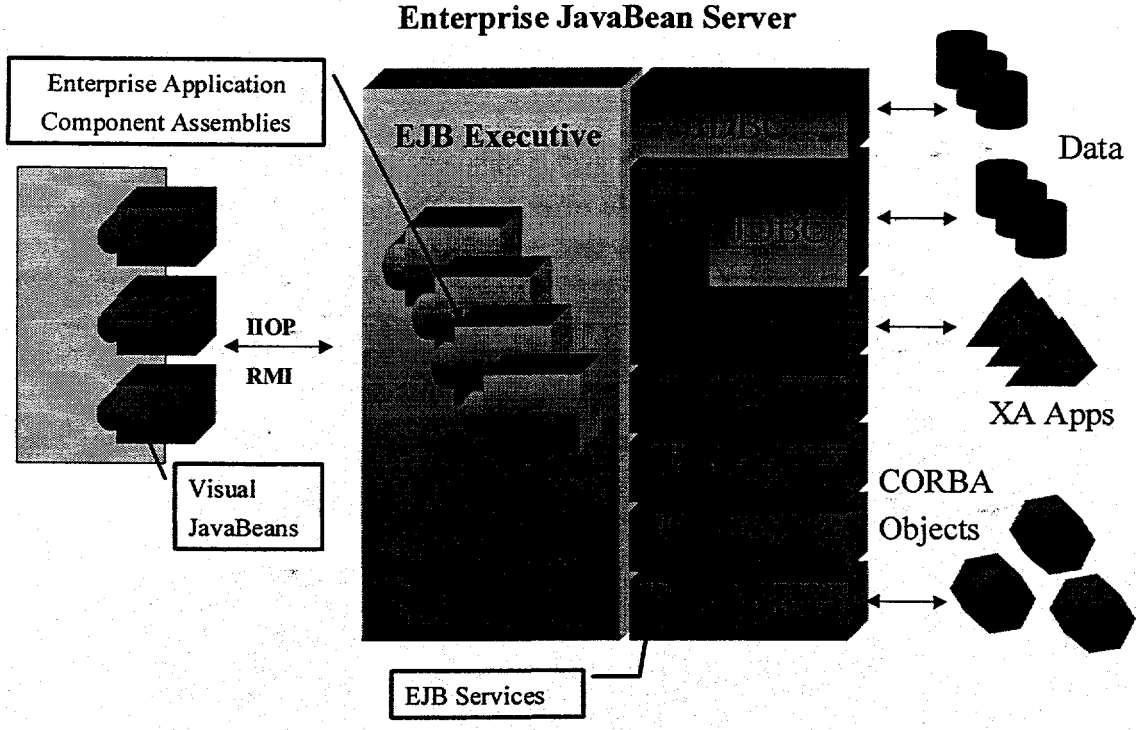
The next generation of Universal Application Servers, such as Netscape Application Server, IBM Component Broker, IONA OTM and Oracle Application Server are incorporating Enterprise JavaBeans and CORBABeans creating Universal Application Component frameworks. *Together, the consolidation of the application server, CORBA, Bean component infrastructure and middleware is the Universal Application Component Server by which the Universal Desktop Component plug-and-plays with the Universal Application Component.* Finally, our sixth component software capability is realized: an enterprise application component is able to integrate with one or more components to form a new enterprise application.



The Universal Distributed Component Infrastructure

The Enterprise JavaBeans framework provides a standard set of application programing interfaces (APIs) which access a core set of underlying enterprise-class infrastructure services. These infrastructure services include:

- JDBC: The Java Data Base Connectivity (JDBC) API provides a uniform interface to a wide range of relational databases.
- JTS: The Java Transaction Service (JTS) provides an API for connecting transaction resources on different machines for a transaction. The JTS API is a Java binding to the OMG CORBA Object Transaction Services (OTS). Like OTS, JTS is based on the X/Open distributed transaction processing model.
- JMI: The Java Messaging Service (JMS) provides a uniform API for access to existing enterprise message queue systems, such as IBM MQSeries and BEA's MessageQ and publish/subscribe, such as Tibco's Rendezvous.
- JNDI: The Java Naming and Directory Interface (JNDI) provides a unified interface to multiple naming and directory services, such as LDAP, DNS and NIS.
- JMAPI: The Java Management API (JMAPI) is a set of APIs for development of distributed system, network and application management for the enterprise.
- JavaIDL provides standards-based interoperability and connectivity with CORBA. JavaIDL includes the API specification for the JavaIDL language Mapping, idltojava, a tool that generates stub code, and a CORBA runtime.



The Enterprise JavaBean Component Framework

In the next section, we will examine how in the very near future you will build applications using components such as DataBroker, a universal data access component, and component frameworks such as Enterprise JavaBeans and CORBABeans.

Universal Data Access Component

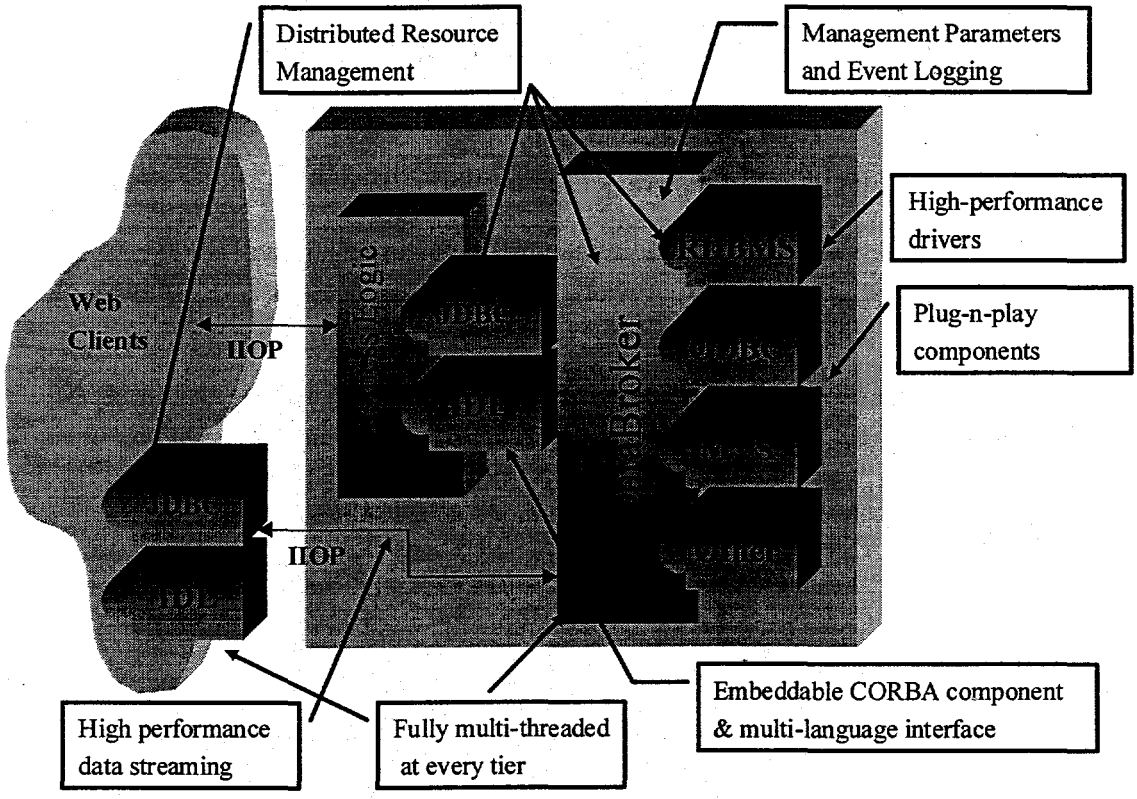
Today, DataBroker is deployed either as a discrete off-the-shelf data access server or as embedded component in an application or application server developed by an enterprise system developer. Because it is based on CORBA, DataBroker is inherently language and platform independent. If a CORBA 2.0 compliant ORB is available, DataBroker can be used. DataBroker uses the latest shipping versions of Orbix and VisiBroker and interoperates fully via CORBA 2.0 IIOP with the various other ORBs that are bundled by infrastructure vendors.

DataBroker is uniquely positioned as the universal data access component for emerging component frameworks such as Enterprise JavaBeans (EJB), CORBABeans and ActiveX/COM.

- DataBroker is a set of components and as such is already a component Container as defined by EJB and CORBABeans. The DataBroker components that are visible are the plug-ins for accessing Sybase, Oracle, Informix, ODBC and mainframe data sources such as DB2 and IMS. The DataBroker manages the lifecycle of it's components. It supplies a set of factories for itself and all of its components. For example, it launches a Sybase plug-in when required. The DataBroker responds to events such as connection request by managing a pool of connections to each plug-in. Also, the DataBroker provides event management such as event logging and automatic recovery from dead sessions and other

"dirty" client exceptions. DataBroker can detect and recover critical connections, memory and other resources "lost" from a client failure during a session. Your enterprise servers are able to run non-stop for months because they are kept from being "polluted" by "zombie sessions" and "lost memory". This built-in recoverability helps maintain the integrity of your databases and servers and networks across system failures.

- DataBroker is uniquely distinguished in offering a CORBA data access object model for relational databases with CORBA IDL that is based on the JDBC specification. Since JDBC is part of the Enterprise JavaBean specification, CORBABeans will likely use the JDBC interface specification for the data access object model. By also offering an OLE DB interface, DataBroker can offer an "universal" data access service for the combined Java, CORBA and Microsoft ActiveX/COM platform markets.
- By supporting the JDBC interface, DataBroker already supports EJB and CORBABean interface introspection and metadata. For example, metadata for the Interface Repository, Introspector, BeanInfo, FeatureDescriptor, PropertyDescriptor, MethodDescriptor are already well specified in IDL.



The DataBroker is actually a set of components such as the OPENjdbc driver, the various native data source plugins for Sybase, Oracle, Informix, ODBC, MVS, and the CWAdmin and Lifecycle Manager components.

- EJB and CORBABeans specify how a component is delivered, distributed and installed. A component must include a deployment descriptor. This is a file that contains the textual description of the deployment properties for a component type. DataBroker already supports a file based deployment descriptor and also a set of deployment properties accessible through its CWAdmin component. Properties available through the

CWAdmin include licensing, database support, inter-component version compatibility, transaction support, and security support. As visual tools emerge for EJB and CORBABeans, you will be able to administrate and manage DataBroker as well as other enterprise components.

Migrating Legacy IT to Component Software

DataBroker is part of the I-Kinetics vision for enterprise component software, called ComponentWare®. In 1998 and beyond, lured by an abundance of success stories of early adopter competitive advantages, the Fortune 1000 will migrate their IT assets to a distributed object architecture. Each enterprise will have a critical need to buy rather than build middle tier application server solutions and reuse rather than re-engineer their existing information technology investment. Gartner Group (1997) estimates that during the next several years at least 60% of new application development will be based on assemblies of components.

A significant trend in the enterprise component market is the adoption (or at least the announcement that they are adopting) component architectures by key business application vendors. Baan, JD Edwards, Lawson, Oracle, PeopleSoft and SAP are among some of the "packaged enterprise application" vendors that have begun breaking their monolithic applications into more flexible assemblies of components.

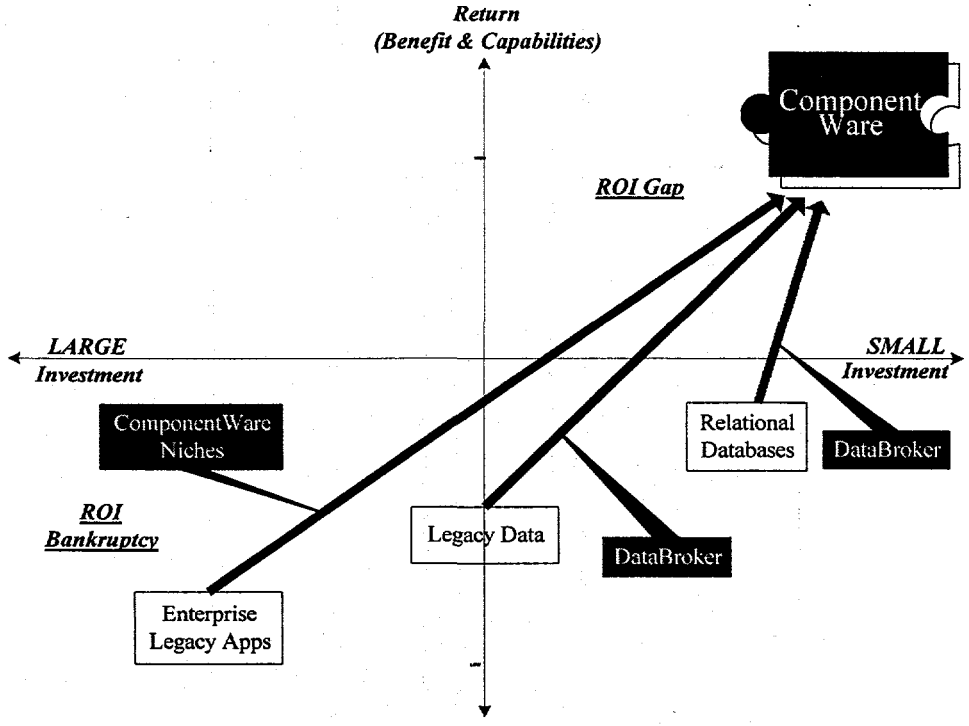
The major need of enterprise IT is the ability to assemble best-of-breed functionality. Components can enable the mix and match of a variety of third-party packages that supplement an enterprise business application. However, the cost of migrating legacy IT to distributed objects at an affordable risk and cost has resulted in the enterprise component Return-On-Investment (ROI) Gap.

I-Kinetics is filling the enterprise component ROI Gap with DataBroker and other forthcoming ComponetWare products. A key technological capability is that ComponentWare transforms an installed base of legacy applications into components, with or without the cooperation of the vendor.

- DataBroker® transforms existing databases, such as Oracle, Sybase, existing middleware, such as ODBC and mainframe middleware and applications, such as DB2, CICS, IMS and ADABAS into fully functional, CORBA-compliant enterprise application components. DataBroker is the first in the ComponentWare® product line of plug-and-play enterprise application components.
- I-Kinetics will continue to extend the ComponentWare product line by expanding the range and quality of legacy data and applications supported. The next major target is transformation of enterprise resource planning (ERP) applications such as SAP, Baan, and PeopleSoft, sales force automation (SFA), such as Sibel Systems, Concentra, Trilogy and customer interaction and support (CIS), such as Remedy, Scopus and Vantive into components. The major need is for the enterprise to build "Relationship Management" applications that provide an integrated view of prospects and customers across field sales, telesales, telemarketing, marketing and customer

services and support. Much of the integration will be combining data from different applications through standard interfaces such as JDBC and OLE DB.

- I-Kinetics Professional Services works with clients to migrate their enterprise IT to CORBA for a competitive advantage. Leadership in technology, the Component-First™ methodology and training for building and managing very large component-based distributed system is a key asset to our customers in assuring success.



ComponentWare lowers the risk and cost of integrating legacy Enterprise IT assets. It transforms legacy into enterprise components, increasing value through increased access and functional capabilities. Higher return for less investment results in enterprise components that fill the ROI Gap.

2.2. Military System Software Needs Analysis

The Reference Architecture stratey was driven by the following military software system needs:

- **Military Systems that change as rapidly as mission (Faster, Better, Cheaper)**

A system framework must be architecture-independent, and broad enough to address both large and small domain concerns. It must address significant integration and concurrency control problems. As the scale of the system grows larger, crossing more functional boundaries, it must provide very reliable services, with little or no downtime. The framework must provide shared access to other components, sub-systems and systems without introducing single-points of failure.

- **Leverage (reuse) current investment in legacy systems**

Large-scale integrated military software systems are likely to evolve into very large distributed systems composed of semi-autonomous subsystems. These different subsystems will have been developed and maintained by different mission-oriented

groups and contractors. These subsystems will have different conceptual frameworks, addressing different domains. Also, these subsystems are likely to maintain and manage their own internal data representations, and workflow methodology. Each subsystem will be integrated into an expanding system or mega-system by exporting a well-defined subset (the interface) to the framework.

- **Reliable & Scaleable**

Cost and reliability concerns dictate that integration into a component framework introduce little or no intrusive modifications to existing systems. Integration must enable in-place system elements to be maintained and enhanced independently of each other. Moreover, a framework should facilitate portability, to enable migration of system elements to higher performance platforms. Technology transfer and management risks should also be recognized, by reducing adjustments to training and operational procedures, and stand-downs for system replacement and validation.

These needs resulted in the following identified requirements of the reference architecture methodology:

- Enable new systems to be assembled on demand from a collection of components.
- Base the component framework on emerging technology and standards for object-based frameworks.
- Enable the reuse of analysis, design and quality assurance as well as code reuse.
- Enable a legacy application to be transformed into a component, while leaving the legacy code undisturbed.
- Enable commercial "off-the-shelf" (COTS) applications to be used as components.
- Enable arbitrarily complex systems to be assembled from either individual components or applications composed of individual components.

All new technology is to based on de-facto standards: TCP/IP, UNIX, C, C++, Ada, CORBA

2.3 Virtual Application Warehouse

ComponentWare promises to do for information systems what chip standardization did for computer manufacturing. The ComponentWare promise is to package applications as standard, re-usable software components. The foundation for ComponentWare is a critical mass of the leading platform vendors adopting object-based distributed computing frameworks and standards – initially CORBA.

The Virtual Application Warehouse (VAW) represents the ability to assemble new applications on demand from legacy applications and data sources. The Virtual Application Warehouse is based on ComponentWare technology. Once the VAW is in-place, users are able to assemble custom applications from components located anywhere on the network while developers continue to add components to the VAW. The result is both the ability to adapt quickly to different organizational demands and achieve reliability through the reuse of verified legacy code and data repackaged as components and managed in the Virtual Application Warehouse. Reusable components result in significant reductions in the cost of mission critical information system development, test and maintenance.

The key technical innovation of the VAW is the dynamic (run-time) integration of application and data components. The prior state-of-the-art, static (compile-time) integration, requires that changes in one application be propagated to all other dependent application interfaces and the application recompiled. Typical examples are adding a new command or data message. Dynamic integration enables applications to be assembled at run-time from reusable components rather than built as static, monolithic code configurations.

The ObjectPump extends dynamic integration technology one step further by enabling the dynamic synthesis of a component adapter for a legacy application. The ObjectPump encapsulates a legacy application or data source, transforming it into a component at run-time, while leaving the legacy application undisturbed.

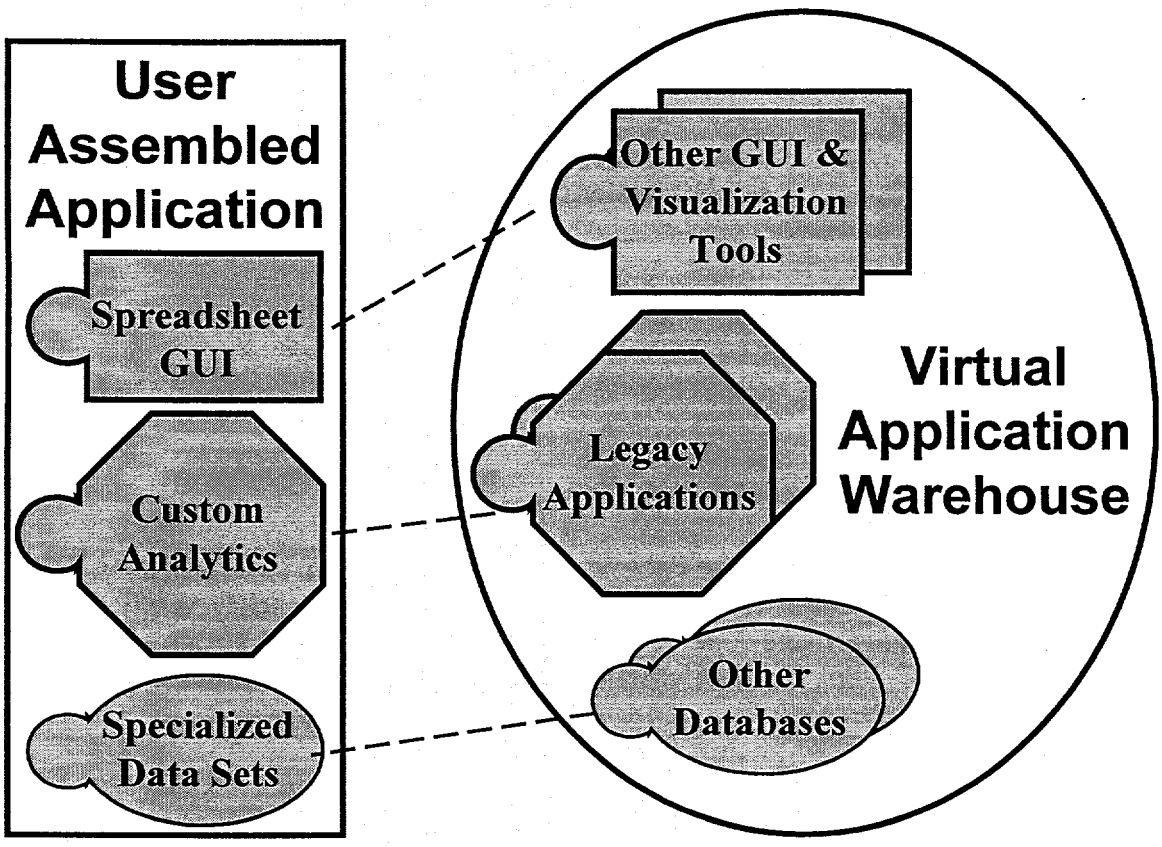


Figure 1. The Virtual Application Warehouse enables specialized applications to be assembled from a montage of components distributed over a network of machines. Any specific application configuration lives only as long as it is active. However, this range is very wide. Producing a report from a database may result in an application lifetime measured in minutes. An interactive analysis application may live for hours; planning and scheduling applications for weeks to months; monitoring applications until its subject (deep space robot, hospital patient, chip fabrication run) is no longer of interest. For

mission-critical systems, a VAW realizes immediate significant cost reductions. Rather than building redundant testbed systems, virtual testbeds can be assembled on demand from verified production system components without reducing the reliability of production systems. New system capabilities can be verified by testing new components in a virtual testbed which shares most or all of the production system.

1.1 Overview of the Need and Emerging Technology for ComponentWare

An estimated 85% of the installed base of software is a custom application with a production quantity of one. In practice, almost 100% of mission-critical systems are custom software [IK89, IK90, IK93a, IK93b, IK94]. Paradoxically, the marginal costs of producing additional units are near zero. So why hasn't the software market, a market with high design costs and low productions costs evolved like other similar custom widget industries, such as automobiles and hardware chips?

The software industry seems immune to market pressures that have motivated a multilevel supply chain structure in other widget industries: design cost recovery, improve quality through specialization, and enable rapid assembly from purchased components. The primary barrier is technical: we (as an industry) do not know how to independently produce reusable software components (ComponentWare). Some of the symptoms of this are:

- Systems are designed and built from scratch as monolithic structures without benefit of purchased components or identifiable reusable parts.

- Each vertical-market supplier and system integrator must possess every horizontal software technology specialty in-house.

- Despite a high degree of shared functionality across broad classes of applications, a viable (pervasive) component industry has not emerged.

The long-range goal of ComponentWare is to overcome barriers to building and deploying systems from verified, reusable software components. I-Kinetics, by itself, could not accomplish this goal. Rather, this ambitious R&D is based on and leverages the following:

1. A viable ComponentWare technology infrastructure is starting to emerge industry-wide. The basis of this infrastructure is a critical mass of the leading platform vendors adopting an object-based distributed computing framework. The current framework contenders, such as Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) [OMG91]. OpenDoc and Microsoft Object Linking and Embedding (OLE), as a group, have almost achieved the required critical mass. The OMG CORBA is an industry-wide (Apple, DEC, HP, IBM, Novell, Sun, others) initiative in the specification of object-based distributed computing [OMG91,OMG92]. Once a object-based framework is widely available, the ComponentWare infrastructure will begin to emerge, followed by widespread commerce in software components. Significant reductions in the cost of system testing and requirements validation will drive the development of the initial base of software components.

2. The ComponentWare Consortium (CWC), comprising seven leading companies in object-based distributed computing technology, has been awarded TRP funding

to greater facilitate software reuse in the 90's. The Technology Reinvestment Project (TRP) is a federal initiative to integrate the commercial and defense sectors through cooperative R&D and commercialization of critical high-technology. I-Kinetics will manage the CWC, whose current membership includes BBN, Heuristicrats Research, IONA Technologies, NetLinks Technology, SunSoft, and Pratt & Whitney. Total research and development by members will exceed $30,000,000 over the next two years. The goal of CWC is to package data and applications as standard, re-usable software components. We will do this by taking advantage of emerging object-based distributed computing frameworks such as the OMG's CORBA, OpenDoc or Microsoft's OLE/COM. CWC will submit their developments to the OMG for possible adoption as industry standards. CWC will also be working with the two other TRP funded consortia, whose members include Andersen Consulting and IBM to advance ComponentWare and framework interoperability.

Each CWC member will be contributing and integrating key ComponentWare technologies. IONA, the current CORBA market leader, will extend Orbix® with object groups and fault-tolerant services. BBN, I-Kinetics and HRI will be contributing jointly developed WWW-CORBA gateway technology. For workflow, SunSoft will continue to enhance its recently announced "publish and subscribe" and Distributed Objects Everywhere (DOE) technology. "Publish and subscribe" allows groups of users to implement complex workflow on demand. NetLinks will develop a new generation of developer tools for inclusion in ORBitize™. Heuristicrats Research will develop a range of security and data analysis components. Another key technology milestone for the CWC is "Plug&Play" components. Standard components for Sybase, Oracle, Informix and Ingres are already available for I-Kinetics ComponentWare™ product line. The next releases of the I-Kinetics ComponentWare will incorporate the latest consortium member contributions in security, fault-tolerance and workflow services.

The Requirements of a Virtual Application Warehouse

There are three basic requirements of ComponentWare and the Virtual Application Warehouse:

1. Creating components;
2. Assembling components together to create applications.
3. Assembling complex applications from component-based application assemblies.

This third requirement defines general composability in which arbitrarily complex and large applications can be assembled from either individual components or applications composed of individual components. This is a requirement that will drive the functionality of successive generations of ComponentWare technology and methodology.

For these requirements, the Virtual Application Warehouse offers an comprehensive methodology for applying ComponentWare to large mission-critical distributed information systems. The following subjects motivate the VAW approach:

- A definition of ComponentWare and how it results in different reusability dynamics than object-oriented development.
- The research and analysis of the needs and functionality required of reliable and scaleable mission-critical information systems. The result of this needs analysis is the specification and design of the Virtual Application Warehouse architecture. Dynamic (run-time) integration of components is the key common technical capability of the VAW.
- Several different component classes are identified and specified for the Virtual Application Warehouse. From this followed the research and analysis of component assembly methodologies. The basis groundwork for deploying arbitrarily complex systems from component assemblies is established.
- The study of different methods for transforming legacy applications into components. This work motivates the two complementary approaches to component generation: (1) compile-time tool-based generation of a component interface and (2) run-time, dynamic synthesis of a component interface.
- The ObjectPump extends dynamic integration technology one step further by enabling the dynamic synthesis of a component adapter for a legacy application. The ObjectPump encapsulates a legacy application, transforming it into a component at run-time, while leaving the legacy application undisturbed.

Each of these is detailed in the following sections.

Definition of and the Reusability Dynamics of ComponentWare

In the object-oriented (OO) community, the term component has increasingly become the general term that labels all kinds of objects and class libraries that are designed for reuse. We have adopted the definition of component as

any object-based module of code that is designed to facilitate code reuse

There are two areas of object technology that represent the emerging object-based reuse technology and methodology disciplines: (1) object-oriented programming languages and class libraries; (2) object-based component software (ComponentWare). The issues and goals of object technology differ for the two areas, although some of the basic concepts are the same. In fact, the term object technology is hard to precisely define because it is so often applied to each area without taking the differences into account.

Object-oriented programming languages and development tools are useful for organizing function and data structures in the form of source code modules. The most common object-oriented method for packaging and organizing these source code modules is the class library (C++, SmallTalk, Lisp). These language-based object definitions (class libraries) facilitate the rapid coding of applications and can be shared and reused in a

specific programming language. For example, a GUI class library would include classes such as Window, Button, RadioDial, ListBox, ScrollableListBox, etc. A data structure class library would include classes such as List, Array, Heap, Hash, Btree, Bag, etc.

However, object-oriented programming languages and class libraries do not provide a means for separate applications to be integrated with other custom applications or packaged software. Nor do OO-progamming languages enable developers or users to assemble applications from components. They do not fulfill the need for diverse objects, supplied by any company, written in any programming language, to freely interact – an essential requirement for achieving the promise of a component industry.

Application components (ComponentWare) are relatively large object-based components that are designed to facilitate functionality reuse at the application level. ComponentWare defines a binary object interface that is independent of programming language. It also defines a mechanism to ensure that connections between components are valid, even as components are individually upgraded or replaced. Object-oriented programming languages, on the other hand, define a programming language (source code) standard, making their objects dependent on both language and implementation. These languages assume that when an object is upgraded, dependent applications can be recompiled and re-distributed simultaneously. This is unrealistic in a distributed system, where components are typically supplied by different companies and different programming teams.

Programming classes facilitate code reuse. Most of the labor and time involved in developing applications is consumed in analysis, design and verification of an application relatively little (10-20%) is involved in actually writing code. Thus programming classes maximum increase in productivity through reduction in the overall effort is at most 20-30%.

Application components facilitate functionality reuse at the application design level. They are used in the analysis and design phase of the effort, while lowering considerably the quality assurance cost, and thus begin increasing productivity well before any code is written. Additionally, unlike class-library based components that were developed from the ground up to help programmers, application components are designed and created in a top-down manner. Indeed, to the extent that a design depends on pre-existing application components, the need for new code is reduced to that required to "glue" the components together to form a new application.

Additionally, ComponentWare technology enables legacy applications to be transformed into components. Achieving legacy code reusability by transforming legacy applications "in-the-whole" is significantly more cost effective than tearing apart and rebuilding legacy code into class libraries.

In summary, the major reuse dynamic of class-libraries is code reuse. The major reuse dynamic of ComponentWare is analysis, design and quality assurance reuse. Now consider the impact on productivity when a legacy application can be turned into a component and that legacy application can be used in the design and verification phase of a new application, the impact on productivity is even more significant. Reuse based ComponentWare, reduces analysis and design, as well as coding, resulting in a reduction of effort of 50-400%. Future gains based on assembling applications from legacy applications can be expected as high as a factor of 10 to a 100 (1,000% to 10,000%).

Needs Analysis of Mission Critical Information Systems

The design of the Virtual Application Warehouse is driven by the following needs of large mission-critical information systems:

- Systems that change as rapidly as organization (Faster, Better, Cheaper)
- Leverage (reuse) current investment in legacy systems
- Reliable & Scaleable

The needs motivate the following identified requirements of the VAW:

- Enable new applications to be assembled on demand from a collection of components.
- Base the component framework on emerging technology and standards for object-based frameworks.
- Enable the reuse of analysis, design and quality assurance as well as code reuse.

- Design tools and methodology for
    - rapidly migrating legacy systems into components;
    - managing components.
- Enable a legacy application to be transformed into a component, while leaving the legacy code undisturbed.
- Enable commercial "of-the-shelf" (COTS) applications to be used as components.
- Enable arbitrarily complex systems to be assembled from either individual components or applications composed of individual components.
- All new technology is to based on de-facto standards: TCP/IP, Unix, Windows, C, C++, CORBA, OLE/COM, etc.

These requirements resulted in the functional specification of the Virtual Application Warehouse:

- Support CORBA and OLE as the object-based frameworks. A critical mass of the UNIX platform vendors (ATT,DEC,HP,IBM,SGI,SUN) are supporting CORBA, while Microsoft OLE has approximately 80% of the desktop PC market.
- Implement component application assemble with dynamic (run-time) integration of components.
- Implement dynamic integration with the exchange of data specification (metadata) between components.
- Support three major methods for transforming a legacy application into a VAW component.
    1. Re-engineer the legacy application into a component or set of components;
    2. Statically encapsulate a legacy application by developing a custom component adapter;
    3. Dynamically encapsulate a legacy application, by dynamically synthesizing a component adapter.
- Support common control protocols for all components.
- Define tools for aiding in the capture and management of component specifications.

Three different classes of components are identified for the VAW that fulfill the functional specification of the VAW. Each component class offers a subset of functionality required by the VAW.

1. **Application Components**

A class of components that offer a standard method of accessing a wide range of applications and services, from spreadsheets to files, to databases, to legacy applications.

2. **Framework Components**

A class of components that bridge different frameworks such as OLE and CORBA. Offers interoperability between differ CORBA ORBs as well as "Network OLE" capability.

3. **Tool Components**

A class of Components for component development and management, such as the ObjectPump, performance monitoring and diagnosis, and system management.

At least one Application Component and one Framework Component is required in the assembly of a new application. The following figure shows the basic VAW component assembly configuration for the UNIX environment using CORBA as the common framework.

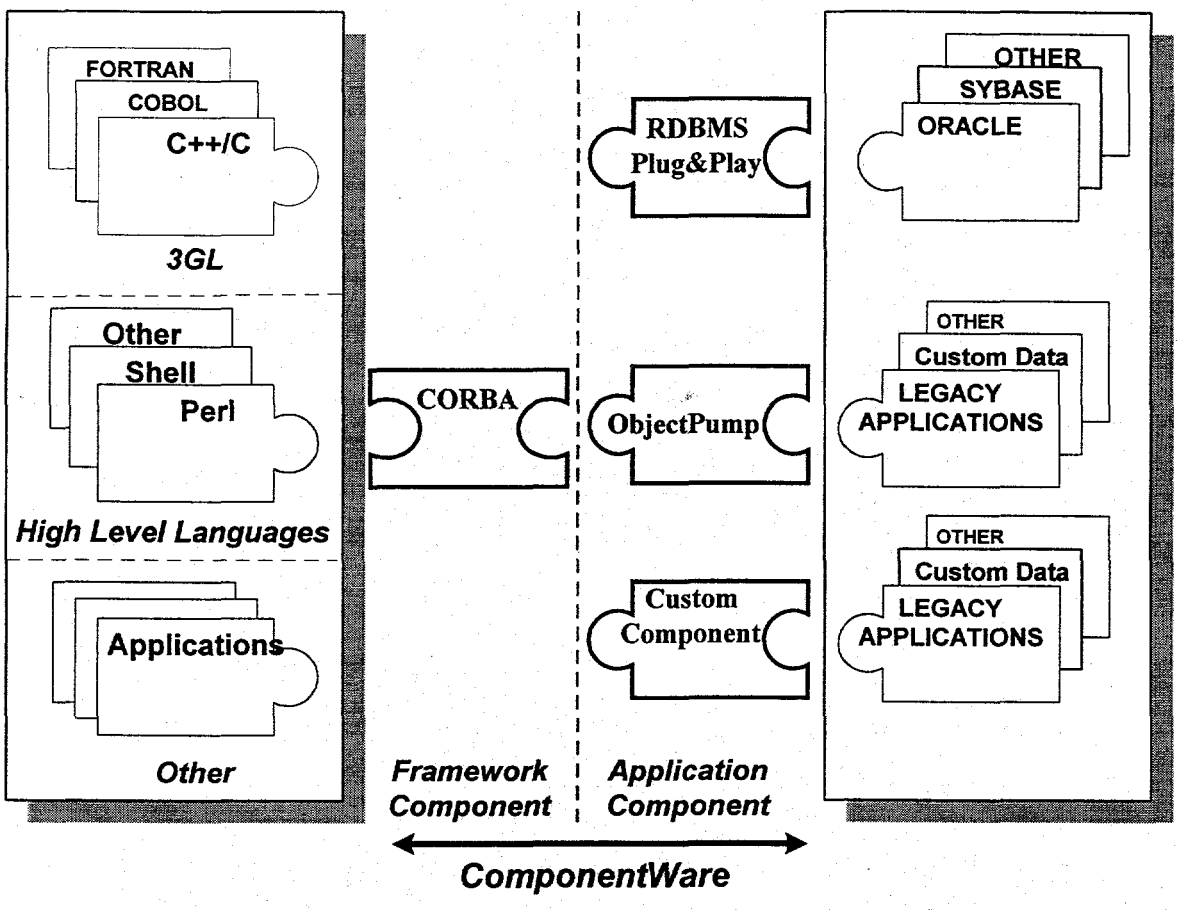


Figure 2. VAW Component-based application configuration assembly using a CORBA framework. This VAW component assembly configuration demonstrates how any Perl , C,C++, FORTRAN and other applications can dynamically integrate with VAW components through a CORBA Framework Component. The VAW is populated with Plug&Play components such as Sybase or Oracle and ObjectPump component adapters and custom component adapters to legacy applications and data.

A key innovation which demonstrates the power of ComponentWare is that two different Framework Components can be used in an assembly to implement inter-framework integration. The following figure shows the basic VAW component assembly configuration for a mixed Windows-UNIX system using CORBA and OLE as the common framework.

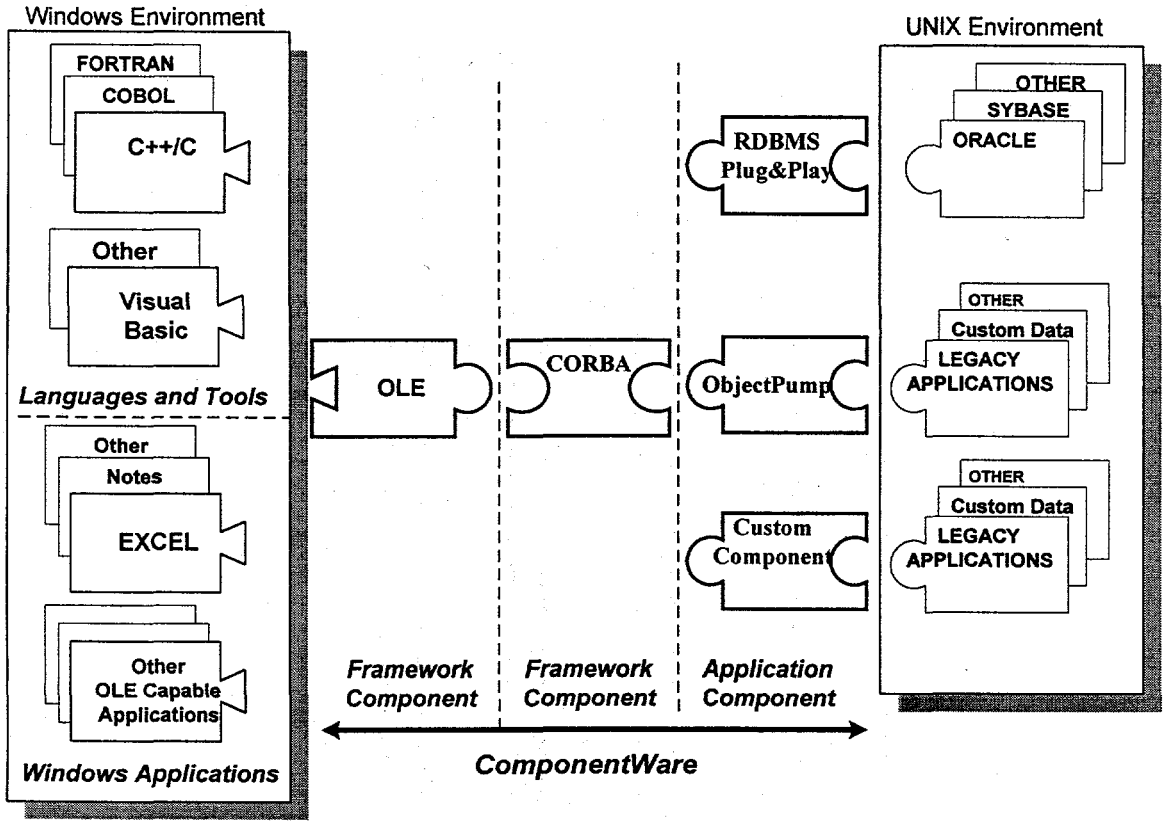


Figure 3. Windows-Unix VAW Component-based application configuration assembly. This configuration demonstrates how PC Windows (desktop) application, such as Excel, or a Windows development tool, such as Visual Basic (VB) can dynamically integrate with any VAW component by assembling a chain of Framework Components on demand. For example, Excel connects into the CORBA framework through the OLE Framework component. The OLE Framework Component connects to the Orbix Framework component. The OLE and Orbix Framework Components together result in interoperability between Windows framework (OLE) and the Unix framework (CORBA). From Excel's point of view, the CORBA-based Application Components appear as OLE Automation services.

Methods for Migrating Legacy Applications to a VAW

There are three basic strategies (methods) for migrating legacy applications to a object-based framework.

(1) Re-Architecture, Re-Implement

Re-implementing legacy software for CORBA requires re-designing it as if it were a new application. This approach is the most costly as well as the most disruptive to the current installed system. However, it has the greatest potential to realizing full distributed computing. Complete design walkthrough and restructuring of the application architecture allows the application to be broken up into sharable components.

This method and the Custom Component Adapter Retrofit method are the current state-of-the-art methods available using CORBA technology. Although one can imagine tools and methodologies for partially automating this

(2) Custom Component Adapter Retrofit

If the application has a suitably modular approach, with a well defined interface, then an adapter can be added. The adapter enables the framework and application to interoperate. It translates between the application's data and control interface and that required by the framework. The most benefit for the least cost is realized when the application code does not require changes to its internal data structure or internal logic flow.

This is current state-of-the-art using current CORBA and ComponentWare technology. I-Kinetics uses this approach for RDBMS and other applications which have a large installed base. When this approach is used the result is high-performance Components whose interface is static but which can dynamically assembled with other Components. I-Kinetics calls these types "Plug&Play" Application Components as the custom applications they encapsulate can be migrated into a CORBA framework by installing the specific "Plug&Play" Component for that application.

(3) Non-perturbative Encapsulation

This strategy is the only choice if the application code can not be modified. "No-touch" applications are quite frequent. If the complete set of code is no longer available, or if the application has a large quality-assurance investment the result is "No-Touch" applications. Also this approach is used by "outsiders" who do not have the privilege of perturbing an application. This case describes most production systems.

This is the ObjectPump solution. The ObjectPump encapsulates a legacy application or data source, transforming it into a component at run-time, while leaving the legacy application undisturbed. When this approach is used the result is rapidly developed, low-cost Component interface encapsulation of legacy applications.

"Plug&Play" Components

"Plug&Play" Components are custom component adapters. They are targeted "best-of-breed" commercial applications such as RDBMSs. For example, the RDBMS

"Plug&Play" ComponentWare gives you an off-the self component adapter which can be installed and immediately used. Current "Plug&Play" Components available include:

- Informix RDBMS Component
- Oracle RDBMS Component
- Sybase RDBMS Component
- ORBPerl: Perl Component
- Common Component Interface (CCI) Library for 3GL languages: C,C++,Fortran.

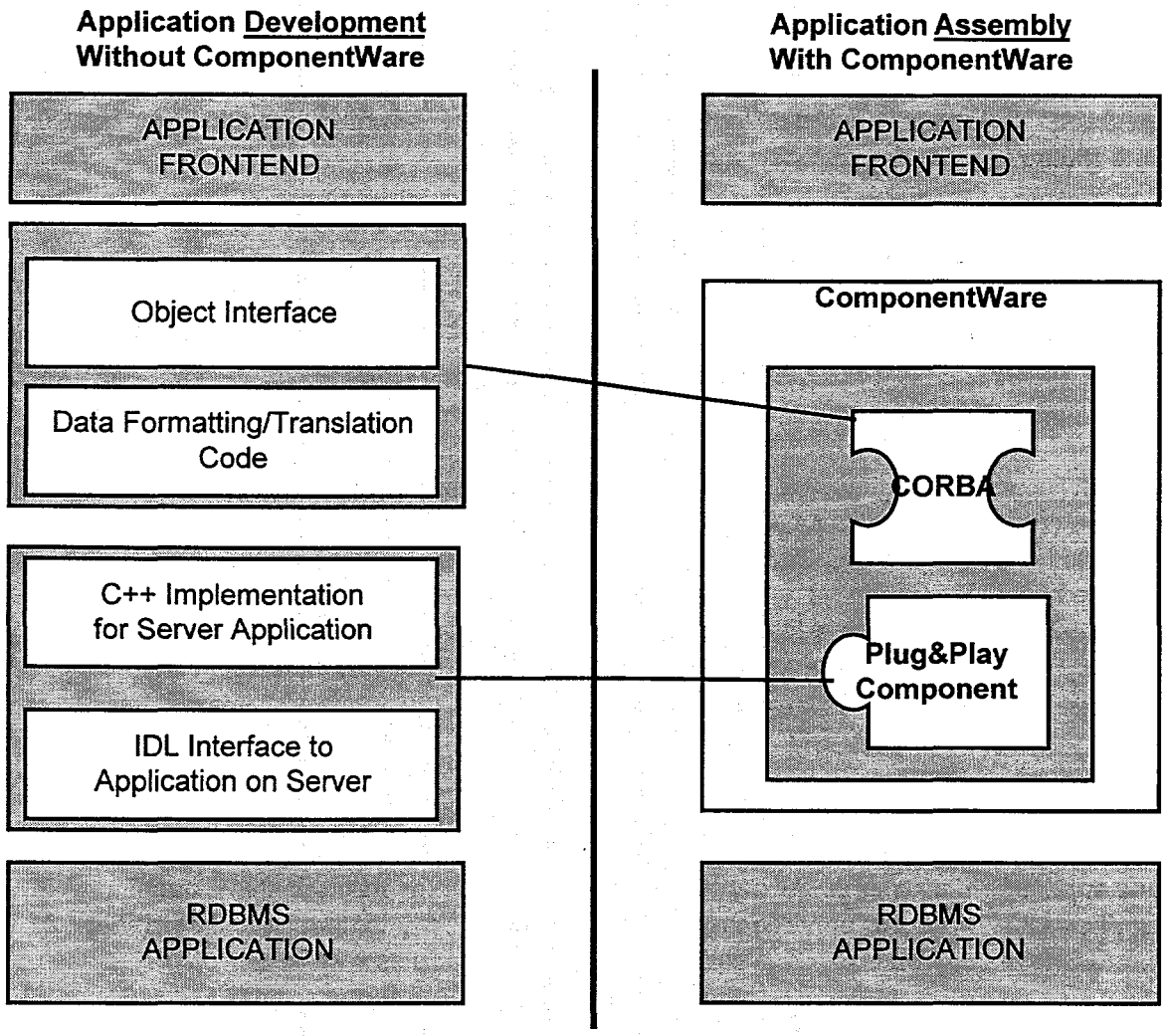


Figure 4. Plug&Play ComponentWare decreases development time by installing pre-built component adapters for RDBMS applications.

ObjectPump

The ObjectPump is a universal component adapter for legacy data sources and applications. The ObjectPump enables the VAW to dynamically encapsulate an application, transforming it into a component, by dynamically synthesizing a component adapter.

The ObjectPump addresses a key limitation of current object-based distributed system frameworks: static component interfaces and data models. Static (compile-time) integration of applications requires that changes in one application specification must be propagated to all other applications. The size of a distributed system is limited because the cost of development and maintenance grows disproportionately faster than the size of the system.

We argue that dynamic component adapter synthesis is an essential requirement to achieving the benefits of ComponentWare. For typical system integration projects, that do not use an object-based methodology, 60%-80% of the integration code involves the transformation of data between legacy code modules [IK91, IK94]. Data specification is even higher for object-based methodologies, because of encapsulation disciplines, approximately 80%-95% component adapter is data transformation [IK93a,IK93b, IK94]. Unfortunately, this causes the component adapter to be very sensitive to changes in the encapsulated legacy application. Maintenance or enhancement requires changing the component adapter. Typical examples are adding a new function or data structure.

The ObjectPump integrates with other objects by exchanging both interface and data format specifications at run-time. New component adapters, specified by the exchanged metadata, can be created (synthesized) at run-time. The ObjectPump acts as mediator between the target data source or application and the Framework Component.

The ObjectPump's dynamic interface capability is based on a model of data model independence. Data model independence, sometimes referred to as data transparency, is where a component can receive data from a foreign component without having to support the foreign component's data model.

There are two major methods to achieving data model independence. Both of these methods require that the data model specification (metadata) as well as the data be exchanged.

1. The data producer returns both the metadata and the data. This requires the receiving object to transform the data into the required form.
2. The data specification is sent by the client object The producer object transforms any returned data into the form specified.

In following Figure, a conceptual diagram of the ObjectPump architecture is shown. The ObjectPump is organized as three major layered modules: (1) CORBA and Native Platform Services module, (2) Automation Interface Manager, and (3) Data Interface Manager.

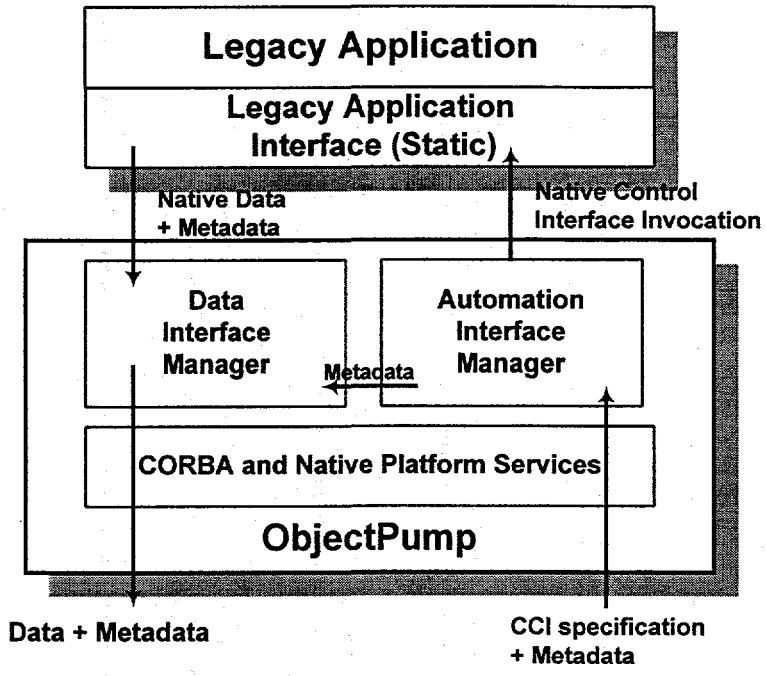


Figure 5. The ObjectPump Architecture

The CORBA and Native Platform Services module manages the interaction of the ObjectPump with CORBA services and the native platform services. The Automation Interface Manager and Data Interface Manager integrate directly to the encapsulated application. The Automation Interface Manager translates the Common Component Interface (CCI) specification into the application's native control commands or call interface. The Data Interface Manager translates the encapsulated application's native data and metadata into the specified data and metadata defined for this particular component.

VAW Toolset for Component Definition and Management

The ideal goal of total automation of a component from legacy code is not achievable with current technology. Because the design and internal use of the legacy code is underspecified, capturing the required control and data interface specification of a legacy code hunk is a human-level problem solving cognitive skill. However, a partial solution of this task, tools to aid the developer in the specification of a component of legacy code, will have significant impact.

Each tool in the VAW toolset is a member of the Tool Component class and as such will be components will all the capabilities of ComponentWare. Like their sibling the ObjectPump, each one can be combined with another component or components to create a new tool or application. The VAW tool set consists of:

1. Component Control Interface Definition Wizard (CIDW)
2. Component Metadata Interface Definition Wizard (MIDW)
3. Component Environment Definition Wizard (EDW)
4. User Definition Wizard (UDW)
5. Component Interface Definition Repository (CIDR)
6. Virtual Application Warehouse (VAF)

The VAW Toolset is specifically targeted at legacy applications and data sources that have no data model independence capabilities. The technical approach and functional specification of the VAW Toolsets are:

1. **Component Control Interface Definition Wizard (CIDW)**

Using the CIDW, the developer interactively identifies which control functions to execute to generate the necessary data. Initially target executables will be DLL's, scripts, Shared Archive libraries, and command line executables. Parameters in the command execution can also be specified. The specified parameters can be marked as public or private. Public parameters will allow users to input "arguments" to the parameters. Private parameters are specified by the developer and cannot be accessed by the end user. In addition, developers can access other components in the Component Interface Definition Repository enabling the creation of complex components composed from other component definitions.

2. **Component Metadata Interface Definition Wizard (MIDW)**

With the MIDW, the data format, type and structures of the component are specified. This metadata interface definition is managed separately from the control interface definition. This enables the developer to separate data from control. For primitive data types, this step is necessary because of the different data type implementations on a wide range of hardware. For example, choices have to made on whether to export a 16-bit integer as a 16-bit, 32-bit or 64-bit integer. For complex data types, such as relational tuples, matrices, lists, etc., the metadata must be captured because the development languages typically do not support these complex data types.

3. **Component Environment Definition Wizard (EDW)**

The EDW enables the developer to specify and manage platform environment configuration and run-time behavior data. The EDW enables the capture and management of such information as the network address of the encapsulated legacy application or data source, executable or file name, physical machine dependent path, and any start up parameters. By separating this information from the metadata and control interface definition, component become machine and location independent. For example, the ObjectPump uses the Environment Definition to locate the executable or file and access it.

4. **User Definition Wizard (UDW)**

The UDW manages one or more end user descriptions of a component. Whereas the Environment Definition is used for programmatic access, the User Definition is used to facilitate end user access and localized customization. The main goal is to present a description of the component that the end user can understand. An example would be naming a component "Patient Monitor Bed 6" or "June Sales Report". End users seeing this name would know immediately what the component was for. Information stored in the User Directory Specification is Description, Alias, Owner, Long Description, and Usage Explanation. Users can then browse this information to help them utilize the component to their best advantage.

5. **Virtual Application Warehouse (VAF)**

The Virtual Application Warehouse will be able to access component definitions in the CIDR. The user will be able to assemble components interactively, invoke them, and capture the resulting output. The Virtual Application Warehouse is designed to be the general container for a entire range of planned ComponentWare addon functionality. These add-ons include: ability to access component run-time performance data and the ability to schedule components for execution based on time or events. For UNIX, the Virtual Application Warehouse GUI will be implemented using Tcl/Tk, a GUI scripting language.

6. **Component Interface Definition Repository (CIDR)**

The CIDR will manage and store the definitions from the VAW Component Definition Wizards: MIDW, CIDW, EDW, and UDW. A developer will be able to specify the control and metadata definition to one or more legacy applications, files or data sources and store it as a component definition.

The ObjectPump ability to dynamically synthesis a component adapter is accomplished with the Component Interface Definition (CID). The ObjectPump processes the required transformation between two different data interfaces based on the CID. For example a "Telemetry Output" CID would enable the ObjectPump to encapsulate at runtime a telemetry feed. ObjectPump supplies automatic transformation and formatting of exchanged data. For the Virtual Application Warehouse, the ObjectPump will act as mediator between the target data source or application and the data object's data interface.

The ObjectPump interacts with the CIDR to synthesis the legacy application adapter. The Component Control Interface to Definition specifies the input of data, the extraction of data, and executing commands using the legacy application's native interface. Example commands are UNIX shell scripts, existing FORTRAN, C, or C++ applications, or system libraries.

The ObjectPump transforms the data input to or output from the encapsulated application as specified by the Component Metadata Interface Definition. Once the data is produced, the Metadata Interface Definition specifies the data type, format and structure the output data is to be transformed to. This provides data type and model consistency among a wide variety of applications. Since the Metadata and Control Interface Definitions are separate, a component is application independent. A different Control Interface Definition can be combined with another Metadata Interface Definition to define a new component. The result is reusability of Component Interface Definitions. This allows staff to change or introduce new applications without disturbing the legacy application or a production system composed of many components.

In addition the CID Repository also provides Environment and User Definition for a component. Examples of Environment Definition information are location of binaries and host operating environment. This information will be used for executing command sets and locating binary files. User Definitions will be used to store 1 to N user specific descriptions of a component. Examples of User Definitions are "Telemetry Output" or "Science Package Command Set". This information helps the user determine the use and relationship of the component in a large mission operations system.

In following figure a conceptual diagram of the ObjectPump using the Component Interface Definition is shown.

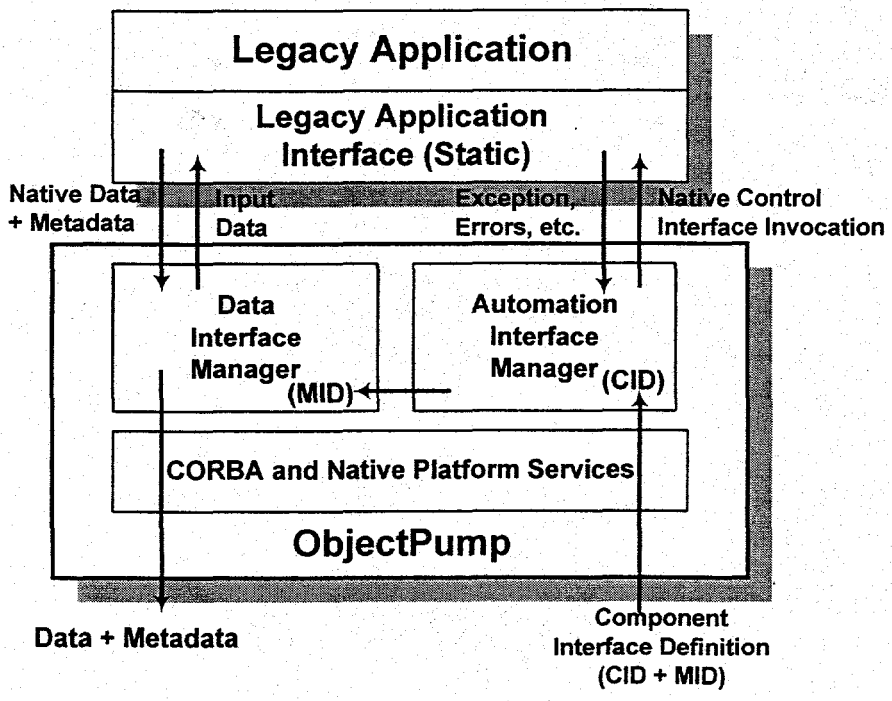


Figure 6. ObjectPump and the Component Interface Definition

In step **0** a client application invokes a component. The ObjectPump retrieves the Component Interface Definition from the Repository. The Automation Interface Manager executes the Component Control Interface Definition (CID) in step **1** using the Native Platform Services (shell, libraries, etc.) of the host where the application/data source resides. All handshaking, exception or error handling is specified by the CID. The Automation Interface Manager provides default handling when control is not specified. In step **2** the resulting raw, unformatted data is returned from the invoking the encapsulated application. In step **3**, the Data Interface Manager transforms the raw data into the data format specified by the Component Metadata Interface Definition (MID). Both the data and the metadata are sent to the target client.

The closest related research of ObjectPump and its capabilities are found in the somewhat related areas of knowledge exchange and metadata management. All of these require a certain level of capability in exchanging metadata, managing data interface specifications and creating complex data structures at runtime. Based on communication and feedback from academic researchers and OMG members the level of design and implementation of ObjectPump is unique in its range of dynamic datatyping and metadata management functionality. It should be noted that ObjectPump is in full-compliance with the OMG CORBA specification.

Component Assembly Methodology

A key capability of the VAW and ComponentWare is the ability to build new functionality by adding new Components to a existing layer of Components. A Component-based application can be assembled, quality-assured, documented, and its behavior well understood. This assembly of Components can then be used as the base layer on which a new application or service can be built. As the system grows, it becomes increasingly easier to expand and evolve because new applications and services are "layered" on a base of ComponentWare that continues to grow in functionality and maturity.

Component-based application assembly using layering occurs at run-time. Component-based layering is a reuse methodology. It compares to object-based inheritance. However, the reusability benefits of inheritance are only available to developers during the design process. Object-based systems are static, there behavior is fixed once installed. Component-based systems are dynamic, there behavior and functionality can be changed once installed. The reusability benefits of layering are available to end-users as well as developers during all phases of the Component's life.

Systems built from layers of component-based services demonstrate the advantages of peer-based distributed systems over client-server systems. Components display peer-to-peer ability as each one can act as both a client or a server.

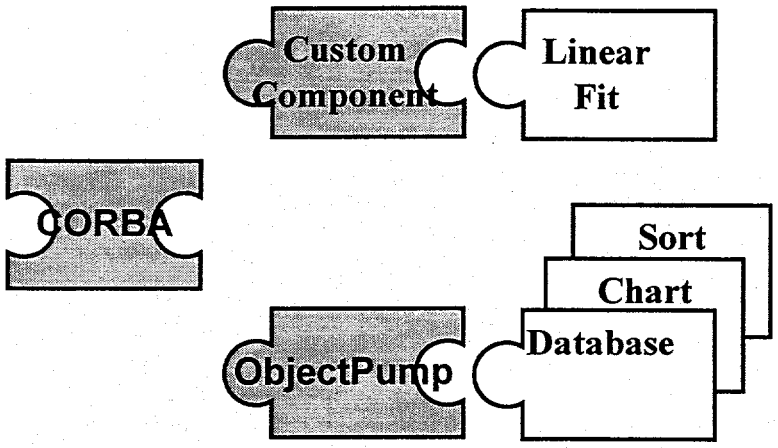


Figure 7. A component assembly demonstrating *Layering*. A Data Analysis assembly is created from Sort Engine, Chart, Database and a Linear Fit components.

Delegation is a special type of layering in which a new Component is created by encapsulating completely an existing Component and adding functionality. The existing Component is hidden within the new Component. One common use of delegation is to upgrade an existing Component with new functionality for new applications while retaining backward compatibility with existing client applications. There actually maybe two Components available to the entire system, but new applications see only the current (new) version of the Component.

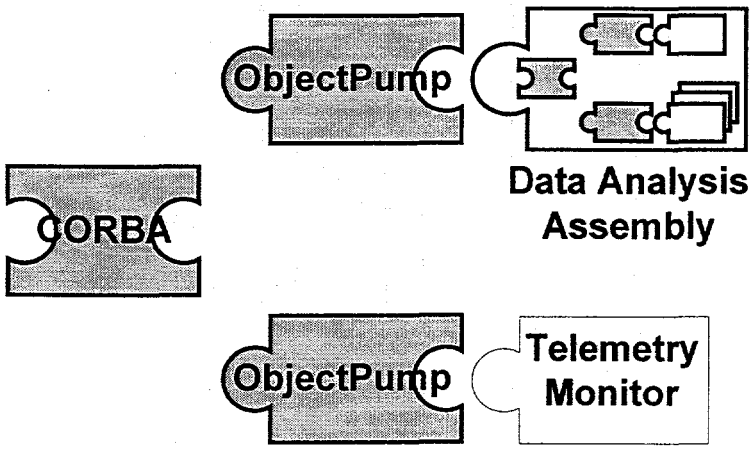


Figure 8. A component assembly demonstrating *Delegation*. The ObjectPump is used to encapsulate the Data Analysis assembly. A Telemetry Monitor component is added creating an Operations Monitoring assembly.

A. Bibliography

[APM90] Oskiewicz, E., J. Warne, and M. Olsen. *A Model for Interface Groups* Technical Report, Architecture Projects Management Ltd., 1990.

[Ah87] Ahamad, M., P. Dasgupta, R. LeBalnc, and C. Wilkes, *Fault tolerant computing in object-based distributed operating systems*. IEEE 6th Symposium on Reliability in Distributed Software **and Database Systems, IEEE, 1987**

[BOW94] Bowman, Mick, Peter B. Danzig, Darren R. Hardy, Udi Manber, Micheal F. Schwartz *The Harvest Information Discovery and Access System,* Second International Conference on the World Wide Web, Geneva, Chicago, Ill. August 1994.

[Ch91] Chin, R. & S Chanson, *Distributed Object Based Programming Systems*, Computing Surveys, March 1991

[Em91] Raj, R., E. Tempero, H. Levy, A. Black, N. Hutchinson, & E. Jul, *Emerald: A General Purpose Programming Language*, Software - Practice and Experience, 1991.

[HRI92] O. Hansson and A. Mayer. *Decision-Theoretic Control of Constraint-Satisfaction and Scheduling.* In Proceedings of the ORSA/TIMS Intelligent Scheduling Systems Symposium, San Francisco, November 1992.

[HRI90] O. Hansson, G. B. Holt, and A. Mayer. *Toward the Modeling, Evaluation and Optimization of Search Algorithms.* In D. Brown and C. White, eds., Operations Research and Artificial Intelligence, pp. 11--28. Kluwer, Boston, 1990.

[IK91] B.H. Cottman and R. Wood, *ObjectExpress: A System for Dynamic MetaData Exchange*. White Paper, I-Kinetics, Inc. Cambridge MA 1991.

[IK92] B.H. Cottman and R. Wood. *A System for Large-Scale Dynamic Integration.* White Paper, I-Kinetics, Inc. Cambridge MA, 1992.

[IK93a] B.H. Cottman. *A Framework for Distributed Planning and Scheduling Systems.* I-Kinetics, Inc., Final report ARPA contract DAAH01-93-C-R255 June, 1993.

[IK93b] B.H. Cottman, *A Framework for Concurrent Engineering Information Systems.* I-Kinetics, Inc, ARPA contact DAAH01-93-C-R155 August, 1993

[IONA93], *Orbix Programmer's Guide*. Iona Technologies Ltd. Dublin, Ireland, September, 1993.

[Jo93] Johnson, D., M. Anderson, R. Carlos, N. Geise, and G. Higgins, A *Pragmatic Distributed Object System*, Position paper, Workshop on Object Oriented Large Distributed Applications, OOPSLA Conference, Vancouver, 1993.

[ISIS92] Hagsand, O., H. Herzog, K. Birman, & R. Cooper. Object Groups: *An Approach to Reliable Distributed Programming*. Pub. in proceedings of 2nd Inter. Workshop on Object-Orientation in Operating Systems, I-WOOS'92.

[ISIS93] K. P. Birman, *The Process Group Approach to Reliable Distributed Computing*. Technical report TR-91-1216, Computer Science Dept., Cornell university. February 16 1993. Accepted for publication in CACM.

[LEE94] Berners-Lee, T., et.al., *The World-Wide Web,* Communications of the ACM, v. 37, n. 8, August 1994, p. 76-82.

[NL92] A. Ewald and M. Roy, *The Evolution of the Client/Server Revolution.* Network World, Nov. 16, 1992.

[NL93] A. Ewald and M. Roy, *Why Object Technology is Good for Systems Integration.* Object Magazine, Feb. 1993.

[OMG91] Object Management Group, *The Common Object Request Broker: Architecture and Specification Version 1.1*, OMG Document Number 91.12.1, OMG, Framingham MA, 1991.

[OMG92] Object Management Group, *Object Management Architecture Guide, Revision 2.0*, OMG Document Number 92.11.1, OMG, Framingham MA, 1992.

[OMG94] Object Management Group, *Common Object Services Specification, Volume 2.0*, OMG, Framingham MA, to be published.

[Su92] Srinivasan, Sumana. *Analysis and Design of an Abstracty Object Model for Fault-Tolerant Distributed Computing*, Thesis, Syracuse University, Department of Computer Engineering, 1992.

[SUN94] *Messaging Object Service.*, White Paper, Distributed Systems Services, Project DOE, Sun Microsystems, Moutain View, CA, April 1994.

2.4 ComponentFirst Methodology

We find ourselves in the early stages of the move to an information technology (IT) infrastructure that unites object technology with the Internet's universal distribution capability. But any IT infrastructure transition represents a potentially very costly and risky change for an organization. This article describes a low-cost and low-risk systematic approach for migrating enterprise IT to a distributed object infrastructure.

The past two years have given us the explosive growth and acceptance of the Internet, the near instantaneous adoption of Java, and the rapid shift of the software industry from client-server to distributed object architectures. The adoption of object technology is motivated by a demand for fast response to requirements changes; objects facilitate complexity management. Internet-based systems provide a mechanism to rapidly and cost-effectively deliver IT to the enterprise's new frontiers.

The fact that the IT infrastructure evolution is driven by significant capability gains for the organization makes it no less uncomfortable. After the initial value is realized from the Internet's low-cost management and distribution of documents, the problem remains of how to integrate into the new infrastructure the enterprise's IT assets that do the organizational heavy lifting. Billions of lines of code, representing thousands of staff-years of domain expertise, are already in production. These mission-critical applications are strategic assets that cannot be discarded, yet redesigning and redeveloping them for the Internet's distributed object infrastructure is blocked by tremendous cost and risk.

The I-Kinetics "Component-First" approach described in this article has been distilled from experience gained from a number of large integration and reengineering projects and three architecture generations of the ComponentWare product line using CORBA distributed object technology. The shared goal of all these projects was to maximize the integration of the enterprise's legacy assets while minimizing disruption of the enterprise's in-production systems and processes. Legacy applications are noninvasively transformed into CORBA-based application components. The enterprise gains new distributed object-based applications from different assemblies of the repackaged legacy application components.

Getting Started: Auditing Your Legacy System

Let's briefly consider the situation facing an organization that has made the decision to move to a distributed object infrastructure. If enterprise IT has a significant investment in existing software, much of this investment is probably at the homegrown infrastructure level. The valuable part of the system, the domain-specific logic representing business rules or proprietary data-processing algorithms, is often hopelessly intertwined with the infrastructure services. This is usually manifested as the well-known "high maintenance" cost of software. It is not the business logic that requires the maintenance; rather it is the need to continuously enhance and extend the infrastructure that soaks up most of the maintenance budget.

It's worth noting that software is not designed this way. Normally these applications begin life as well-designed implementations. They are useful and valuable to the organization, so over time they are "extended" in a number of ways; organizational requirements change and new logic is added. They are modified to communicate with other applications, a user interface is added, state is made persistent, log files are added, and so on. Most of these changes are infrastructure changes: they lie outside of the domain of the business application and exist only to support the implementation of it. These enhancements are also typically *incremental*; the new functionality is not added all at once as part of a redesign, but gradually and more or less independently of other enhancements.

This evolution results in a system that's difficult to maintain and extend. New enhancements often introduce as many bugs as they do features. This is a familiar refrain, but a key distinction is that the homegrown infrastructure is the chief culprit. Generally, such infrastructures evolve to address a narrowly defined problem, while a comprehensive infrastructure solution is too broad to lie within the expertise of any but system vendors. In the pathological case, the state of the system actually drives business decisions instead of supporting them ("We can't do that because our system won't be able to support it...").

Unfortunately, the application logic is still useful; in fact, by this time it is often perceived as "mission-critical". Resources continue to be devoted to its maintenance. Alternatively, maintenance cost and downtime risks become so great that the application can no longer be extended – it's "frozen".

How can such applications be transitioned to a distributed object architecture? There are two basic approaches: complete re-implementation or incrementally integrate and re-engineer the legacy system. The prospect of wholesale reengineering is fraught with well-known risk - all of us have witnessed some spectacular failures in the last ten years. Incremental reengineering using encapsulation may offer a way out of this thicket. We'll look at how encapsulation naturally transforms a legacy system into a set of usable and reusable enterprise components.

Legacy Integration and Reengineering Using Encapsulation

Encapsulation, a well-known benefit of object-oriented architectures, refers to hiding implementation details behind a public interface. Using encapsulation for integration and reengineering enables us to use the functionality of the legacy software while also hiding the legacy application behind a new object model.

In referring to legacy application encapsulation, we'll use the terms *encapsulation* and *wrapping* interchangeably. Also, we'll use the terms *encapsulation wrapper* and *wrapper* interchangeably. Wrapping is not an innovative concept. A casual web search for encapsulation or wrapping will easily yield hundreds of examples where this technique has been used. Accomplishing integration with encapsulation wrappers is a standard technique. It has been used excessively for prototypes or a quick fix where functionality has been traded in for lower project costs. However, because of standards-based

distributed object technology, such as IONA Orbixware,IBM Sanfrancisco, Netscape ONE, JavaSoft Enterprise JavaBeans or Oracle NCA, and when used with a systematic discipline, such as the Component-First approach, wrapping lowers the cost and risk of integration as well as improves the overall architecture and increases the capabilities of the legacy system.

We've found that there are three major encapsulation strategies: Ad-Hoc, Legacy-First, and Component-First. First we'll examine the Ad-Hoc and Legacy-First approaches, showing why these common strategies are not suitable for transitioning legacy applications to distributed object architectures. We'll then consider the requirements a legacy application encapsulation wrapper must meet if it is to satisfactorily serve as an integration mechanism; this is the basis of the Component-First approach, which we'll then look at in detail.

Ad-hoc Approach

The Ad-Hoc encapsulation process is opportunistic and unique for each project. Its major disadvantage is that any Ad-Hoc project approach is very difficult to reproduce. Typically there is no "encapsulation theory" applied (that is, analysis of what makes a "good" wrapper). Each use of the legacy application drives the design of each wrapper. Since wrappers are typically regarded as conveniences, not much effort is devoted to them. Ad-Hoc wrappers tend to be complex, fragile, and error-prone, often introducing instability into a previously stable legacy application. They become maintenance burdens themselves.

Legacy-First Approach

The Legacy-First approach is appropriate when the goal of the wrapping is to export the application interfaces into a new environment. Examples of this are "shell" wrapping like the MS-DOS prompt in MS Windows, or rehosting the terminal screen output of a mainframe application onto a Windows platform using screen scrapping. We don't want to change the user model or add new functionality, we just want it to be available in a different platform environment.

The starting point of the Legacy-First encapsulation approach is the legacy application specification. The question answered is "How can I export the interfaces to this application into the new environment?" This is a slightly different question from "How can I integrate this into a new environment?" or "How can I transition this into a new environment?" Consideration of these questions leads us to view encapsulation in a different light.

Driving the encapsulation from the functional specification of legacy application is not always appropriate. This Legacy-First approach is particularly inappropriate when the application is part of a transition to a distributed object system. Encapsulating an application merely to use it in a new environment suggests different design goals than encapsulating an application as a step in a transition process. Yet this distinction is rarely made.

The chief consequence of Legacy-First approach is the exposure of legacy infrastructure constraints into the new system. For distributed components, such exposure can have disastrous consequences. We typically intend these components to interact; indeed we want to *construct* new applications with the components we have made by wrapping the legacy applications. Yet the applications were never designed to be composable or to work in concert with other components. The interfaces they export do not fit into a framework or normally even adhere to a single strategy. Consequently this technique is not effective as a transition method to distributed object systems.

Our experience, distilled from migrating many large production legacy systems to CORBA, has shown that encapsulation is a complex and difficult problem to address because of the number of roles a legacy application wrapper must fulfill. Encapsulations of different systems seldom look the same. Should they? We feel the answer to that question is "Yes". The basis of transforming legacy systems into application components is that there are sets of needs that are common across the enterprise. These needs translate into patterns of encapsulation – the basis of the systematic Component-First encapsulation approach.

Requirements of a Complete Encapsulation Wrapper

Legacy encapsulation is a complex process because a wrapper has many different responsibilities. The I-Kinetics Component-First approach describes five main requirements for a complete encapsulation wrapper:

1. Provide connection protocol management on different levels.
2. Accomplish data translation and information processing on potentially different levels;
3. Implement some sort of error detection/ recovery mechanism.
4. Manage the application environment.
5. Implement new object interfaces that conform to widely supported distributed object specifications as much as possible.

Let's look in detail at the first four of these Component-First wrapper requirements. The last requirement is a complete discipline in itself, to be covered in future Component-First case studies.

1. Connection Protocol Management

The wrapper must first of all connect to the legacy system. This may involve opening and managing different connections if the legacy communicates over different interfaces. The complexity of managing these connections (which normally consist of platform-specific resources managed by the operating system, like sockets, pipes, and file I/O streams) is a function of how many and in what way the connections are used by the given application..

Second, the wrapper must often implement some application-level protocol (for example, a login or a dialog). It must at least know how to format the different commands that drive the legacy application and how to make sense of its output. The application-level dialog can easily become complex when the legacy application maintains state.

2. Data Translation and Information Processing

Two types of data transformations can appear in a wrapper:

- Architectural-level transformations may need to be performed. Examples are binary buffers of data being passed among machines with different internal representations, or data being inserted or extracted from a bit stream with a proprietary format.
- Higher level transformations may be required if the wrapper interface needs to return a datum derived from application results. For example, if a legacy application returns a time interval and a distance traveled, and the wrapper interface is required to supply velocity, then the remaining computation must be performed in the wrapper.

3. Error Detection and Recovery

Most legacy applications do not have externally specified exception handling. Many do not even have robust error detection, reporting, or recovery functionality built into them. This can present problems if we want to construct a well-behaved component from potentially fragile legacy code. This is one of the more difficult challenges in encapsulating legacy applications, and we won't be addressing it here. It is closely related to the next category, environment management.

4. Environment Management

To properly encapsulate a legacy application, it must appear to the application that it's running in its native environment. If it expects an initialization file of a certain name to be in a certain place, the wrapper must make sure it's there. If the application will write a temporary file in a specific place, the wrapper must ensure that there are no conflicts. If other applications need to be started and initialized because the target legacy application requires them, the wrapper must do this also. This open-ended sort of management is the source of considerable pulling of hair and gnashing of teeth to those tasked with implementing wrappers.

Consider an application that performs a simple data transformation. It reads an input file (in a certain format) and writes an output file (in possibly a different format). Simply to run this program, a wrapper needs to take the following steps: ensure that there are no file naming conflicts, create and write the input file in a specific format, invoke the application, monitor and check completion status, read and parse the output file, remove the input and output files, and return the result. Even this trivial example requires seven or eight actions. It's easy to project the amount of complexity involved in robustly wrapping a moderately complex application.

Component-First Approach

There's a better way to use encapsulation to achieve infrastructure transition and integration. Before describing this process, we'll briefly review the architectural goals we've established:

- To maximally leverage legacy assets (that is, legacy domain logic).
- To minimize the exposure of any legacy architectural constraints into our new system. This is where our use of encapsulation diverges from the Legacy-First approach, which naturally exposes legacy partitioning into the new environment. *The partitioning of the distributed component architecture must be driven by a well-*

defined object model that is based on the enterprise domain, not the existing legacy application partitioning.

This second goal, eliminating legacy architectural constraints, is important for a number of reasons. It means we'll invest additional project cost and time to encapsulate the legacy application in such a way as to not let it pollute our environment in the future. In practical terms, this fundamentally changes the way we approach encapsulation. We no longer wrap applications in isolation.

Rather than exporting legacy application interfaces and environments into a new domain, the Component-First approach mandates constructing our new domain first and then determining the minimal set of legacy application functions we need to populate it. We may want only a small fraction of what some large legacy application can do. In fact, in general we won't export any interfaces of legacy applications -- they will all be hidden behind a new object model.

The Component-First encapsulation strategy enables us to re-engineer functionality piece by piece, at a pace driven by the new business needs, and to control the transition. This control enables us to do incrementally reengineering, which significantly reduces the risk involved in undertaking this sort of infrastructure surgery. Application component specifications driven by the new organizational needs, rather than the legacy needs that are compiled into the legacy interface specification, is the founding principal of the Component-First encapsulation strategy.

The Basic Steps of the Component-First Encapsulation Process

The Component-First process outlines three basic steps that encourage usability and reuse of the legacy application capabilities while suppressing exposure to limitations of the legacy system architecture.

1. Perform domain analysis and generate the object model.
2. Identify public interfaces of the object model.
3. Encapsulate legacy applications to populate specific functionality in the object model.

By generating the object model from analysis rather than a collection of legacy applications, we reduce the likelihood that we'll inadvertently cripple our new system by incorporating legacy architecture constraints.

Legacy Transition Case Study: Telemetry Management System

To illustrate the basic steps in the Component-First process, let's look at a case study drawn from a recent project. Consider the telemetry management and analysis system example shown in Figure 1. The Telemetry Manager processes and distributes data from a high-speed telemetry data stream. In its native environment, the Telemetry Manager communicates with a GUI-based Display & Control process and a Telemetry Analysis process using file-based interprocess communication (IPC). The file-based IPC between the processes is implemented with two shared files, one for outbound and one for inbound messages. Each of the three processes finds these files with pathnames embedded in their software code. The Telemetry Manager also performs process control, creating and destroying the client Display & Control and Telemetry Analysis processes.

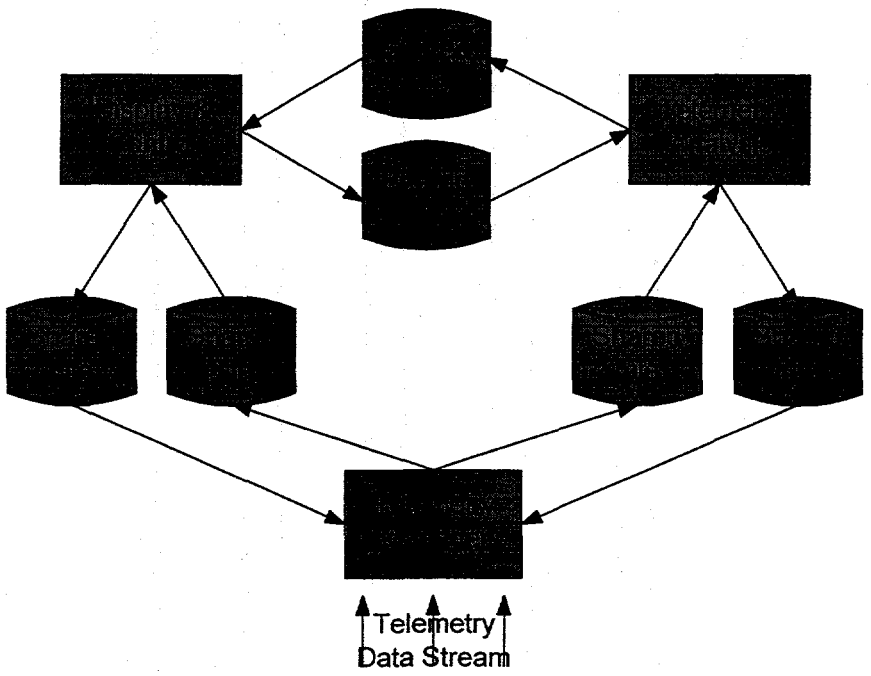


Figure 1. Legacy system for distribution management, analysis and display of satellite telemetry data.

The valuable domain logic part of the telemetry manager is the telemetry data "decommutation", where bit streams are unpacked and data values for dozens of separate fields are extracted. The display & control client "controls" the telemetry manager and the telemetry analysis processes by sending commands through the file IPC mechanism which register the display & control client's interest in specific fields. Either the telemetry manager process or telemetry analysis process sends via the file-based IPC the indicated field values to the display/control process.

Several unpleasant constraints are apparent.

- The system is strictly single-threaded and performance can not be scaled up by upgrading to a multi-processor platform or distributing across multiple hosts. Because of the manner in which the files are specified, only one set of processes can run per machine.
- New processes can not be added. Only one pair of clients can connect to the Telemetry Manager. The Telemetry Manager cannot maintain different sets of fields for different clients.

We were tasked with transitioning this mission-critical system to a CORBA-based infrastructure. The client initially wanted to "wrap" the system by using a Legacy-First approach. The entire system was to be encapsulated, creating a single monolithic component, by exporting the system's command interface with the CORBA Interface Definition Language (IDL). However, the resulting encapsulation wrapper would have propagated the limitations of the legacy system to the distributed object architecture.

Only one transaction management component could be created on a single machine. Only one client application could connect to it. Not only would the new architecture offer no relief from these problems, but it would also severely limit adding new applications that used the legacy telemetry management system.

However, the sponsor needed the next generation of the telemetry management system to have scalability, fault-tolerance based on redundancy, and integration with other legacy systems, such as high-performance analytics hosted at remote installations. The sponsor agreed to use the Component-First approach, trading up for the critical capabilities they needed.

Following the three step process of the Component-First strategy, we did the following:

1. Designed an object model from a telemetry management domain analysis.
2. Used the object model to specify telemetry management component interfaces with CORBA IDL.
3. Wrapped the telemetry management system to populate the specific functionality of each component.

Figure 2 shows the first generation of the object model resulting from a domain analysis is shown. The Field object represents the telemetry data fields decoded from the raw Telemetry Frame data by the telemetry manager. New system component interact with Field objects and not with the encapsulated application. This is a different partitioning of the problem, one, which would be unlikely if the design had been based on the legacy telemetry management system architecture.

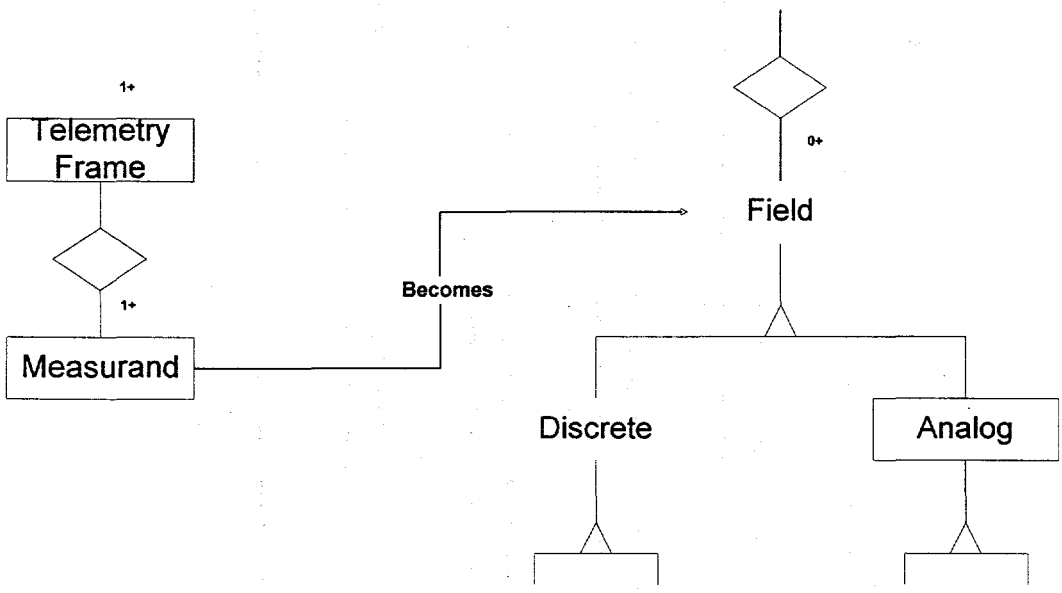


Figure 2. A portion of the telemetry management system object model.

Listing 1 shows the IDL specification for some of the component interfaces of the telemetry management system. Exception handling has been omitted in favor of brevity. The legacy telemetry management system is encapsulated with the DataSource interface. New applications and client components interact through the FieldManager interface, which in turn is the only component to interact with the DataSource interface. These well-defined telemetry management component interfaces completely hide the existence of the legacy telemetry management system.

```
module Telemetry {
// client subscription list for fields
typedef sequence<string> FieldNameList;

// base interface for all telemetry fields
interface Field
{     readonly attribute string name;
      readonly attribute long dev_id;
      attribute long time;
      attribute any value;
      attribute short color;
};

// Manager for all Fields; interacts with DataSource interface
interface FieldManager{
      boolean setup(in long dev_id, in string pass, in Output outRef);
      FieldNameList subscribe(in FieldNameList list);
      boolean advise(in string fname, in double displayValue,
            in long seconds, in short colorValue);

      boolean updateSpecial(in string name, in string value);

      oneway void startReplay();
      boolean stopReplay();
};

// legacy system wrapper interface
interface DataSource                {
      boolean setup(in long dev_id, in string pass,
                  in FieldManager dsRef);
      oneway void replayBegin();
      boolean replayEnd();
      void subscribe(in FieldNameList slist);
};
};          // end module Telemetry
```

Listing 1 CORBA IDL specifications for Telemetry Management Components.

Figure 3 depicts the new distributed object architecture. The Field Manager component is multi-threaded, can be replicated, and be hosted on different machines. The Field Manager accesses the legacy telemetry management system via the DataSource component. The DataSource component encapsulates the legacy Telemetry Manager and Telemetry Analysis processes.

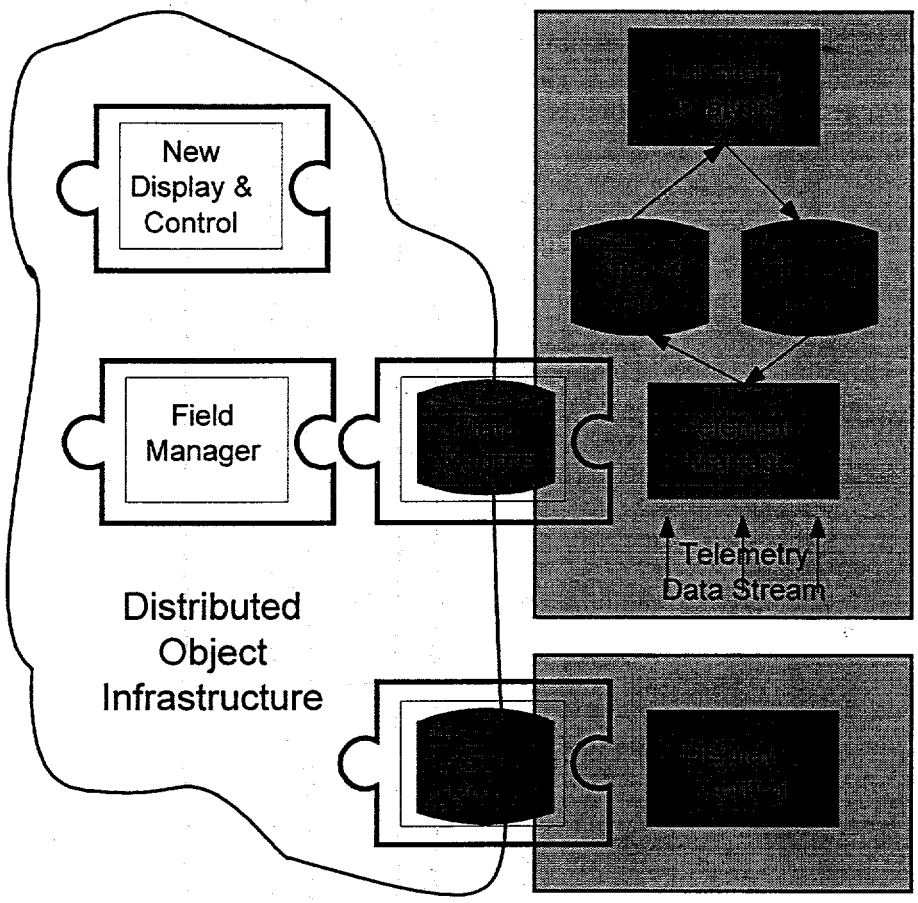


Figure 3. The Telemetry Management System is migrated to a distributed object architecture using the DataSource and GUI Proxy component encapsulation wrappers.

The DataSource component itself is subject to certain constraints, such as running on the same machine as the legacy telemetry management system. However, because the Field Manager is a logically centralized source of all telemetry Field objects, the full benefits of a CORBA-based telemetry management system can be realized. We are freed from the constraints of the legacy system, as they are isolated and contained by the DataSource component encapsulation wrapper.

The Field Manager can receive (and discard) redundant copies of uniquely identified Field objects from two or more DataSource components. We can achieve fault-tolerance by replicating the legacy telemetry system with a "hot-swap" redundant spare. There can be multiple Display & Control components, enabling access to the Telemetry Management System from multiple sites. The system can scale to meet the demands of a growing population of users by copying and broadcasting the Field objects and running multiple Field Managers on multiple machines. Finally, we can answer to the large and constant need to add new telemetry analysis processing. Using the same DataSource encapsulation pattern, specialized "hothouse" analytics can be incrementally added that

consume the raw telemetry Field objects and output new derived data Field objects back into the system.

The component wrappers protect new applications from changes in the legacy system. We can proceed with reengineering the legacy implementations with no effect on client applications. A client requesting notification upon change in state of one of the Field objects has no way of knowing (nor does it care) whether the notification is originally generated by the legacy system or some new mechanism. If we had chosen the Legacy-First approach and created a wrapper that mirrored the legacy telemetry management system command interface, we would have been able to achieve the full benefit of the distributed object infrastructure. Clients using the legacy server would be manipulating it directly through interfaces customized for the legacy application; any reengineering of the original application would be far more likely to impact those interfaces (and thus any new application using them).

The Right Perspective

Encapsulation is an effective reengineering strategy for integrating and transitioning legacy applications to distributed object infrastructures. When used for transition, encapsulation must be approached from the perspective of the new object architecture. The object model must drive the partitioning of the new system. Starting from the point of the legacy architecture will allow unwanted constraints and legacy infrastructure effects to "leak" into the new architecture.

Legacy encapsulation is inherently complex, due to the many functions a robust legacy encapsulation wrapper must perform. Almost every legacy application seems unique when viewed in isolation; from this view the prospect of automating the wrapping process seems dim. However, decomposing wrapper functionality gives insight into this complexity. Viewing the functions separately allows us to apply design patterns to the problem of automating encapsulation.

The systematic Component-First legacy encapsulation strategy enables us to better predict the cost of and identify risk points earlier in the process of transitioning legacy systems to a distributed object infrastructure, such as CORBA. The Component-First approach is continually being advanced and matured into a formal methodology, distilling our experience in transitioning enterprise systems to Internet distributed object architectures using CORBA.

Our experience in the last four years with CORBA legacy encapsulation projects has produced an application "taxonomy" from which we continue to identify encapsulation patterns. These patterns are the basis of the I-Kinetics ComponentFactory™, a tool to automate the Component-First encapsulation process.

3. Future Research and Development

3.1 Component Generation Tools

The Component Factory procedure is the basis for automating Object Adapter generation tools. Using a CASE-based methodology, developers will interactively generate CORBA Interface Definition Language (IDL) specifications from the native data and control interface specifications of legacy code. The vision is to automate the generation of a complete CORBA component from existing application or data source

3.2 Component Repository Management

CASE tools for Intranet or CORBA-based systems have yet to be developed. The continued commercialization of the Phase II spin-offs will requirement an assortment of tools to aid the developer in managing Internet-based CASE tool systems. The first wave of tools will be component management in the form of a repository and tools for managing the repository. Other tools will be required to create and browse CORBA IDL (object interface specifications) as well as visualize object interface relationships.

3.3 Real-Time Survivable Systems

Integrated, distributed systems result in more software and hardware elements that must be effectively coordinated. Tools for developing survivable real-time systems must be developed and evolve as the scale and sophistication of the embedded system changes.

A. Quarterly Technical Status Reports

A.1 01Jun-30 Sep95

Detailed status reports from each CWC/TRP/OORAD joint venture member is given in the following sections. Each task item is noted as (3.x).

BBN

Reported by Ed Walker
The objectives of BBN's work to date in the CWC/TRP/OORAD consortium have fallen into three broad areas.

1. General TRP program planning and familiarization with CWC software components. This work has included early discussion with Raytheon of arrangements for Raytheon to use the BBN CORBA compliant map system.

2. Preliminary work devoted to demonstrating a CORBA based distributed application running over the Internet, showing connectivity, reliability & security, and taking first steps toward developing a world-wide object environment.

3. Our own non-federal funds have been expended on expansion of our GIST tool for performance analysis of multiple distributed processes and on comparative analytical studies of Orbix and other ORB components.

HRI

Reported by Andrew Mayer, President

(3.6) Security

- Delivered and configured CWC/Firebox! @ I-Kinetics, HRI
- Experimenting with secure key exchange protocols for interoperability demonstrations
- Exploring options for Orbix security mechanisms
- Prototyping environments for firewall-firewall technology demos using Orbix

(3.6) Data Analysis

In the area of Data Analysis, we have been investigating the development of CORBA-based statistical data analysis tools. These tools will be "servers" whose methods can be invoked to perform data analysis on data retrieved via an ORB. Our example has been the S-Plus statistical package from Statistical Sciences: it is basically object-oriented, but not directly accessible to other applications.

Our initial demonstration target is to encapsulate S-Plus so that (1) an application can send a message to S-Plus asking for a statistical analysis (e.g., linear regression), (2) passing iterators or object references for the data set to be analyzed. After the analysis is complete, (3) S-Plus returns a reference to a "result" class instance, which can be queried for details (such as coefficients, intercept, goodness-of-fit), or sent a "plot" message. Some subtleties in this process involve sending sufficient statistics (e.g., the covariance matrix) across the network, rather than the complete data set (e.g., a million objects), etc.

I-Kinetics, Inc. (Lead)

Reported by Bruce H. Cottman, President

- (3.1,3.3,3.8) ComponentWare Architecture Whitepaper completed.
- (3.1,3.3,3.8,3.9,3.10) Virtual Application Warehouse Whitepaper completed. Discussion of use of ComponentWare and how it effects the way large distributed information systems will be built using ComponentWare.
- (3.8) Database Components (formely RDBMS Components) beta completed. Demonstrated at August 14, ObjectWorld.
- Attended ObjectWorld August 14, ObjectWorld.
- (3.11) Hosted inter-consortia TRP/OORAD planning meeting Oct. 11, 1995.
- Spun off CWC, Inc. as subsidiary of I-Kinetics, Inc.
- (3.11) WWW.COMPONENTWARE.COM established as Web Site
- (3.11) In partnership with BBN and HRI launched ComponentWare Internet Testbed initiative.

IONA Technologies, Inc.

Reported by Sean O'Sullivan, Business Development Manager

- (3.4) Delivered latest version of ORBIX to I-Kinetics, P&W, HRI, NetLinks.

- (3.4,3.5) Completed development and distributed Orbix+ISIS beta to I-Kinetics.

Pratt & Whitney

Reported by Robert Landgraff, Computer Specialist

- (3.10) Commissioned CWC/TRP/OORAD testbed area
- (3.10) Installed Orbix
- (3.10) Developing prototypes of VAW for Turbine Engine Agile Design

NAVSEA

(3.9,3.11) Commissioned ComponentWare Learning Center
(3.11) Developing demo for ObjectWorld
(3.10) Project plans and requirements specification have been accomplished for Logistics Virtual Application Warehouse.
(3.10) Training and technology rollout plans for Q2 have been finalized.

NetLinks Technology, Inc.

Reported by Alan Ewald, V.P. of Technology

(3.2) ORBitize 1.1

ORBitize 1.1 work included the design, implementation, qualification and productization of a follow-on release to ORBitize 1.0. Enhancements to the ORBitize 1.0 base include:

- Porting of software to the Windows NT 3.5, OS/2 2.1/3.0 and Solaris 2.4 platforms
- Support for browsing Interface Repositories for supported ORBs on those platforms
- Minor usability enhancements and bug fixes

(3.10) Pratt & Whitney

Alan has delivered a four day workshop on Orbix, ORBitize, and building distributed object applications for engineers and managers at Pratt & Whitney in West Palm Beach. Since that time, several projects have undertaken prototyping efforts including the concurrent design application discussed in earlier CWC meetings at Pratt.

(3.2) ORBitize 2.0

Reported by Bruce Cottman, Executive Program Manager, CWC/TRP/OORAD JV

Due to a unexpected shortfall of funding, ORBitize 2.0 has not proceeded beyond the design stage. The current Year-1 effort in this task area (3.2) has been delayed until Year-2. Replacing this task is Interoperability (3.11) work in the form of the Internet-based ComponentWare testbed. This work will be performed by BBN, I-Kinetics and HRI.

CWC/TRP/OORAD, Common Joint Venture

The CWC has evolved beyond its TRP effort to become an entity for marketing and promoting ComponentWare technology. I will describe briefly what has happened:

- The CWC has become a separate corporation
- for use by the CWC/TRP/OORAD members. Different memberships have been designed and membership is now open.
- SIEMENS AG has joined as an Associate Member

The LEGOS (CWC/OORAD/TRP) effort will benefit from the CWC marketing effort and for reference purposes will be called the CWC/TRP/OORAD program. The CWC/TRP/OORAD Program is independently governed by the Cooperative Agreement currently under negotiation with DOE. I-Kinetics is responsible for the CWC marketing vehicle effort and it represents an additional intellectual property contribution.

The mission of the CWC is to make it easier to use, promote, and market components for organizations wishing to build, use, or sell component technology. Each type of member brings its unique blend of skills and needs to create synergistic relationships to enable component sale and adoption. The CWC will actively recruit key organizations for each type of membership in order to insure success. Technology providers will be provided with new channels and customers, solution providers will have enabling tools and accounts to work with, and strategic members will be provided with early access to technology as well as a vertical package of technology and services.

A summary of benefits to CWC members include:
- business partners for co-marketing and new distribution channels
- Early access to technology
- Vendor neutral, industry backed effort providing open systems image
- Consulting services
- Benchmarking and certification for ComponentWare products
- Total package of products and services can be offered to customers by CWC members, which is necessary for larger sites
- WWW based services: "To utilize the Internet to foster electronic commerce between ComponentWare providers and ComponentWare users"
- Leads database via WWW
- On-line distribution of demos
- Solution Provider Index
- Technology Provider Index
- Pointers to training facilities
- ComponentWare Shopping Mart

Currently, the CWC offers four different types of memberships. These memberships are designed to meet the specific needs of each group, provide vertical solutions, and to build new revenue channels for particpating firms. They are as follows:

Associate Member
Associate Memberships are offered to academic institutions and corporate Advanced Technology Groups (a.k.a Corporate R&D Groups). The Associate Membership allows organizations to have early access to technology from design to release. This access will

provide them with the means to evaluate, shape and plan for the use of reusable component-software technology. The Associate Membership also provides access to CWC Technology Provider products.

Solution Provider
Solution Provider Memberships are for consulting and training organizations wanting to leverage or gain expertise in component software technology. The CWC provides the Solution Provider with early access to technology, beta versions of products, training, co-marketing opportunities, and consulting leads. The CWC has certification requirements that the Solution Provider must pass before becoming certified. This ensures a high level of quality to customers and maintains the high standards the CWC demands.

Strategic Member
Strategic Memberships are offered to end user organizations that want a managed and partner approach to implementing component software in their organization. This membership includes everything in the Associate Membership as well as training, consulting, support and project rollout assistance. These features provide the organization with total support and mentoring throughout the evaluation and project rollout reducing both risk and deployment times.

Technology Provider
Technology Provider Memberships are offered to commercial software vendors whose products are complementary or can be components. As a Technology Provider, a firm gains access to new distribution channels via the CWC along with co-development and co-marketing support from multiple vendors. The CWC also has strict software quality requirements that the Technology Provider must pass before they can join. This ensures a high level of product quality to all ComponentWare customers.

There is considerable synergy between the types of memberships. Technology Providers provide tools to Solution Providers who use them to a competitive advantage for building solutions for Strategic Members. Strategic Members get a total multi-vendor package of consulting, products, training and mentoring. Technology Providers get new distribution channels and access to strategic accounts.

A.2 01Oct -31 Dec95

Status on statement of work tasks from each CWC/TRP/OORAD joint venture member is given in the following sections. Each task item is noted as (3.x) and corresponds to the numbered SOW task items.

BBN

Reported by Ed Walker
The objectives of BBN's work to date in the CWC/TRP/OORAD consortium have fallen into three broad areas.

1. General TRP program planning and familiarization with CWC software components.

2. Expanded role in CWC Enterprise Testbed. This work is devoted to demonstrating a CORBA based distributed application running over the Internet, showing connectivity, reliability & security, and taking first steps toward developing a world-wide object environment.

3. Assumed task **(3.6) Security** from HRI.

HRI

Reported by Bruce Cottman, Chairman, CWC

HRI has gone to an inactive status in October, 1995, pending resolution of stockholder dispute. HRI's workplan task **(3.6) Security** has been assumed by BBN and I-Kinetics.

I-Kinetics, Inc. (Lead)

Reported by Bruce H. Cottman, President

- (3.1,3.3,3.8,3.11) Case study completed based on building DB Components for CORBA and OLE infrastructures.
- (3.1,3.3,3.8,3.9,3.10) Component Warehouse talk developed based on experience from current CWC host site projects.
- (3.8,3.9,3.10) Training course for Database Components developed.
- (3.8,3.10) Training of Database components were given to NAVSEA in two workshops in November, 1995. In the second workshop, the "Survey" application was developed and rolled out to production using DB Component for CORBA and OLE platforms.
- (3.11) Hosted inter-consortia TRP/OORAD planning meeting Oct. 11, 1995.
- (3.11) ComponentWare Enterprise Internet Testbed has installed nodes at I-Kinetics and BBN.
- (3.11) Presented talk "CORBA+Internet" to OMG Internet SIG.

IONA Technologies, Inc.

Reported by Sean O'Sullivan, Business Development Manager

- (3.4) Working on version of ORBIX 2.0 for release to CWC.

Pratt & Whitney

Reported by Robert Landgraff, Computer Specialist

- (3.10) Turbine Engine Agile Design VAW has progressed with first version rollout planned for end of January, 1996.
- (3.11) Hosting CWC meeting March 5-8, 1996.

NAVSEA

(3.10) Project plans and requirements specification have been accomplished for Logistics Virtual Application Warehouse.

(3.10) Training and CWC technology rollout occurred in two successive week-long workshops.

NetLinks Technology, Inc.

Reported by Alan Ewald, V.P. of Technology

(3.2) ORBitize 1.1

Completed.

(3.2) ORBitize 2.0

Reported by Bruce Cottman, Executive Program Manager, CWC/TRP/OORAD JV

Due to a unexpected shortfall of funding, ORBitize 2.0 has not proceeded beyond the design stage. The current Year-1 effort in this task area (3.2) has been delayed until Year-2. Replacing this task is Interoperability (3.11) work in the form of the Internet-based ComponentWare testbed. This work will be performed by BBN and I-Kinetics. This action was approved by ARPA and DOE technical program managers at Oct 11, 1995 meeting.

CWC/TRP/OORAD, Common Joint Venture

The CWC has evolved beyond its TRP effort to become an entity for marketing and promoting ComponentWare technology. Current new members include:

- Siemens
- Interactive Objects
- OrbAware

A.3 01Jan -31 Mar 96

Status on statement of work tasks from each CWC/TRP/OORAD joint venture member is given in the following sections. Each task item is noted as (3.x) and corresponds to the numbered SOW task items.

3.1 DYNAMIC COMPONENTS AND COMPONENT COMPOSABILITY

~~3.2 ORBITIZE™ : DOMS INTERACTIVE DEVELOPMENT ENVIRONMENT~~ (PUSHED INTO 3.3)

3.3 COMPONENT DEVELOPMENT TOOLS

3.4 CORBA SERVICES AND FACILITIES

3.5 RELIABLE COMPONENTS

3.6 SECURITY AND DATA ANALYSIS COMPONENT

3.7 JOINT MAP-SERVER COMPONENT

3.8 "PLUG&PLAY" COMPONENTS

3.9 INFORMATION FACTORY FOR CALS

3.10 INFORMATION FACTORY FOR CONCURRENT ENGINEERING

3.11 INTEROPERABILITY

3.1 General CWC/TRP/OORAD Activity

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.1-3.11) General workshop on CWC/TRP/OORAD. Focus theme was "Transforming the Enterprise Investment into Component Software." Workshop proceedings are available at http://www.componentware.com/wshop2, March, 1996
- (3.9,3.11) Workshop hosted by Logistics Management Institute, Evolving the Logistics Systems Investment to Component Software, I-Kinetics, NAVSEA presenting, April, 1996.

3.2 CWC/TRP/OORAD Member Activity

3.2.1 BBN

Reported by Rick Schantz, BBN

The objectives of BBN's work to date in the CWC/TRP/OORAD consortium have fallen into three broad areas.

Task 3.6 Security and Data Analysis Component

We began studying the problem of providing security services for the testbed and demonstration applications. The focus of the effort was on the feasibility of utilizing public key encryption techniques, along with certification as a way to both authenticate and protect inter-orb invocations. This study has been temporarily suspended due to changes in personnel.

Task 3.7 Joint Map-Server Component

The JTF MATT server, which is a CORBA based component providing end user and embeddable map functionality, has been upgraded to use the newest release of IONA Orbix. It has also been reorganized internally to better serve as an embeddable

object component. A next step is to incorporate the map functionality in the Virtual Enterprise Testbed experiments.

Task 3.11 Virtual Enterprise Testbed

We acquired, installed and made operational a BBN hosted TRP machine to serve as a node on the enterprise testbed we are constructing. We have developed a test application and made it operational using the Orbix Orb. We are awaiting installation of a companion node machine at I-Kinetics to begin testing wide area operation of the test application. Following that, we will begin improving the test application to include consortium components for demonstration and evaluation.

In addition, we prepared for and attended consortium meetings in Florida and Massachusetts. We made a group presentation on the CORBA related activities ongoing at BBN.

3.2.2 HRI

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

3/30/96 HRI has dissolved due to a shareholder dispute. The principals have prepared and submitted a final report documenting their work accomplished upto the date of becoming an inactive CWC/TRP/OOAD joint venture member. There principal area of work was in task area 3.6 Security and Data Analysis Components. This task area has been assumed by I-Kinetics and BBN.

3.2.3 I-Kinetics, Inc. (Lead)

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.1,3.3,3.5,3.11) Completed technology plan and architecture for ComponentWare 2.0. The major technical objectives targeted are support of CORBA IIOP and the ability to dynamically wrap a legacy application and map its interface to a given IDL specification.
- (3.8) Released and supported Database Components to 12 beta sites.
- (3.8) Added JAVA client support for Database Component.
- (3.9) Off-site and on-site technical consulting to support the NAVSEA host site.
- (3.10) Off-site and on-site technical consulting to support the P&W host site.
- (3.1,3.3,3.4,3.11) Submitted position paper and in OMG/W3C joint workshop on Distributed Objects and Mobile Code (DOMC).
- (3.11) Joint submission to OMG RFP4, CORBA Common Facilities, Data Interchange Facility, with MITRE Corporation and Objectivity.
- (3.11) "Internet and Component Software", Bruce H. Cottman, talk presented to OMG Internet SIG, January, 1996.

3.2.4 IONA Technologies, Inc.

Reported by Sean O'Sullivan, Business Development Manager, IONA

- (3.4) ORBIX 2.0, CORBA 2.0 compliant, completed and released to CWC.
- (3.3) ORBIX for WWW 1.0 beta released.
- (3.4) CORBA Event and Name Service beta released.
- (3.4) CORBA Persistence Service in the form of ORBIX+ObjectStore released.

3.2.5 Pratt & Whitney

Reported by Robert Landgraff, Computer Specialist, Pratt & Whitney

- (3.10) Turbine Engine Agile Design VAW has progressed with first version rollout planned for end of January, 1996. The system uses Task 3.3, 3.4 and 3.8 deliverables.

EXTERNAL AIRFOIL PASSES STRESS TEST FOR 3D AIRFOIL PARALLEL DISTRIBUTED OPTIMIZATION. The Common Object Request Broker architecture (CORBA) has shown its ability to improve distributed parallel processing. Beta testing of the Common Object Request Broker Architecture (CORBA) Parallel Distributed Optimization Toolkit with the 3D External Airfoil has resulted in the following execution statistics: 9000 executions of a 3D single stage high pressure turbine ran in approximately 52 hours across 40 workstations. This amounts to 40X performance increase over sequential analysis and is near linear in its scaleability. The reliability of the system was 100% and can be attributed to the built-in exception handling features provided by CORBA.

- (3.3) INCREASED FIDELITY ENGINE SIMULATION INCORPORATES CORBA In order to rapidly encapsulate trusted component analyses, the CORBA IDL is being used to wrap legacy systems for use as computational servers by CORBA client objects. The new technology enables the monolithic FORTRAN systems to be integrated together with variable fidelity engine object models. This minimizes re-engineering of the older analyses into the new model environment and facilitates parallel distribution of simulation scenarios across the network.

- (3.11) Hosted CWC meeting March 5-8, 1996.

3.2.6 NAVSEA

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

(3.10) Stage 1 of the reengineering of a logistics management system (TLMS) has been completed. The system uses Task 3.4 and 3.8 deliverables.
(3.11) TLMS Stage 1 moved to Virtual Enterprise Testbed.

3.2.7 NetLinks Technology, Inc.

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

(3.2) ORBitize 1.1
Completed.

(3.2) ORBitize 2.0
Reported by Bruce Cottman, Executive Program Manager, CWC/TRP/OORAD JV
Due to a unexpected shortfall of funding, ORBitize 2.0 has not proceeded beyond the design stage. The current Year-1 effort in this task area (3.2) has been delayed until Year-2. Replacing this task is Interoperability (3.11) work in the form of the Internet-based ComponentWare testbed. This work will be performed by BBN and I-Kinetics. This action was approved by ARPA and DOE technical program managers at Oct 11, 1995 meeting.

3/30/96 NetLinks has been acquired and may seek to rejoin the CWC/TRP/OORAD. They are currently inactive. NetLinks work was to develop successive versions of CORBA Interactive Development Environment (IDE) tools. This task (3.2) has been rolled into task 3.3, Component Tools.

4. Federal Dual Use

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

ARMY/ARDEC Automation and Robotics Design and Lifecycle Management of Real-Time Embedded Systems.

Funded by a SBIR Phase II, the I-Kinetics will transform a set of design tools for ARDECs real-time embedded systems and reference architecture management environment into interoperable components. A key objective is to enable collaborative architecture design and specification among ARDEC clients. Another is to create a general framework for real-time systems design that is independent of any one tool and its specific specification model. This project will apply the products of task 3.1, 3.3, 3.4, 3.8, 3.11.

NASA JPL Small Mission Operations

Leveraging a SBIR Phase II, the NASA/JPL Small Mission Operations is benefiting from the CWC/TRP/OORAD joint venture R&D. I-Kinetics will build a Component Warehouse from JPLs substantial investment in spacecraft command and control, inter-planetary navigation, telemetry management, mission planning, and science package management. A key objective is to expand mission operations outside of the JPL site to the science community using the Internet. Pushing the management and analysis of the science packages out to the science teams is estimated to reduce JPLs mission support costs by an order of magnitude. This project will apply the products of task 3.1, 3.3, 3.4, and 3.8.

Air Force, Phillips Lab, Satellite Telemetry Command & Control Mission Operations

Under a Air Force SBIR Phase I has tasked I-Kinetics with adding a Component Warehouse to their Satellite Telemetry, Command and Control (STCC) testbed. Continuous availability is a key requirement for this mission-critical system. I-Kinetics will transform key parts of the STCC into replicated components. Component replication will enable critical STCC operations and services to continue when any one component fails. This project will apply the products of task 3.1, 3.3, 3.4, 3.5 and 3.8.

Air Force, Rome Laboratory, Satellite Image Analysis Operations

Under a Air Force SBIR Phase I has tasked I-Kinetics with migrating a image analysis toolset to a set of components. The ability to access these components from CORBA, OLE or WWW client environments is a key success criteria. This project will apply the products of task 3.1, 3.3, 3.4 and 3.8.

A.4 01 April - 30 June 96

Status on statement of work tasks from each CWC/TRP/OORAD joint venture member is given in the following sections. Each task item is noted as (3.x) and corresponds to the numbered SOW task items.

3.1 DYNAMIC COMPONENTS AND COMPONENT COMPOSABILITY

~~3.2 ORBITIZE™ : DOMS INTERACTIVE DEVELOPMENT ENVIRONMENT~~ (PUSHED INTO 3.3)

3.3 COMPONENT DEVELOPMENT TOOLS

3.4 CORBA SERVICES AND FACILITIES

3.5 RELIABLE COMPONENTS

3.6 SECURITY AND DATA ANALYSIS COMPONENT

3.7 JOINT MAP-SERVER COMPONENT

3.8 "PLUG&PLAY" COMPONENTS

3.9 INFORMATION FACTORY FOR CALS

3.10 INFORMATION FACTORY FOR CONCURRENT ENGINEERING

3.11 INTEROPERABILITY

3.11.1 Evangelism, Workshops, Technology Transfer, Dual Use

3.1 General CWC/TRP/OORAD Activity and Significant Events

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- Versant has committed to joining CWC. The CWC will realize a strong partner in Object Database Management Systems (ODBMS).
- Lockheed has committed to joining CWC. The CWC will realize a strong partner in Satellite ground systems command and control research and development.
- 3.11.1) Invited talks given at 1st Component Users Conference, Munich Germany July.1996. Conference proceedings to be published by SIGS. Talks given were keynote, "What is Component Software". , and session talks "Data Integration and CORBA"., "Internet and Component Software", "Evolving Legacy Applications into Components", "Lifecycle Management of Components.
- (3.11.1) Invited talks given at ObjectWorld/East May.1996. Focus theme was "Transforming the Enterprise Investment into Component Software."

- (3.9,3.11.1) Workshop hosted by Logistics Management Institute, Evolving the Logistics Systems Investment to Component Software, I-Kinetics, NAVSEA presenting, April, 1996.

I-Kinetics and the CWC members market potential increased overnight from approximately 10,000 to 40,000,000 with the announcement that Netscape will ship CORBA with their WWW browser and server product line. The analysis of this event follows this report in Appendix A.

3.1.1 Upcoming Events

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.11.1) 2-day Workshop to Veterans Hospital Administration on the theme track of "CORBA and Medical Patent Management Information Systems", September, 1996.
- (3.11.1) 6 talks and technology demonstrations at ObjectWorld/West, August, 1996.
- (3.11.1) 1-day workshop to ARMY ARDEC, N.J August, 1996.
- (3.11.1) 1-day workshop to Air Force Rome Lab, Image Lab, Rome, NY, August, 1996.

3.2 CWC/TRP/OORAD Member Activity

3.2.1 BBN

Reported by Rick Schantz, BBN

The objectives of BBN's work to date in the CWC/TRP/OORAD consortium have fallen into three broad areas.

Task 3.6 Security and Data Analysis Component

No activity this period.

Task 3.7 Joint Map-Server Component

We traced DARPA-sponsored BBN progress in this area for future inclusion in the testbed. (-BHC Rick is referring to the Virtual Enterprise Web (VEW) testbed.

Task 3.11 (Virtual) Enterprise (Web) Testbed

During this period we refined the Orbix demonstration application on the BBN-hosted TRP machine. Our plans to get this working across the internet were postponed by I-Kinetics' problems in their IP problems for their testbed machine, the accessibility of which we are awaiting. We consulting with them a number of times on this to offer assistance.

In addition, on another host, with Oracle pre-existing, we installed I-Kinetics Database Component version 3.0 and also Orbix MT 1.3.5, which will be key parts of the testbed demonstration. We also went through a number of iterations with the product's support staff in helping them work through bugs in their documentation and configuration.

3.2.2 I-Kinetics, Inc. (Lead)

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.1,3.3,3.5,3.11) Completed technology plan and architecture for ComponentWare 2.0. The major technical objectives targeted are support of CORBA IIOP and the ability to dynamically wrap a legacy application and map its interface to a given IDL specification. Continued design and development of ObjectPump (3.1,3.3). "Alarm Factory" demonstration of legacy encapsulation methodology has been completed.
- (3.8) Released and supported Database Component Server to 34 beta sites.
- (3.11.1) Database Component Server was awarded "Best New Object Technology Library or Component" at ObjectWorld East, May, 1996.
- (3.3,3.8) Database Components and future "Plug&Play" components have been integrated with Bulletproof's JDesignerPro JAVA GUI Builder tool. I-Kinetics has entered into a co-marketing and co-development agreement with Sun, IONA, Bulletproof centered around this technology bundling.
- (3.9) Off-site and on-site technical consulting to support the NAVSEA host site.
- (3.10) Off-site and on-site technical consulting to support the P&W host site.
- (3.11) Joint submission to OMG RFP4, CORBA Common Facilities, Data Interchange Facility, with MITRE Corporation and Objectivity.
- (3.11.1) Invited talks given at 1st Component Users Conference, Munich Germany July.1996. Conference proceedings to be published by SIGS. Talks given were keynote, "What is Component Software". , and session talks "Data Integration and CORBA"., "Internet and Component Software", "Evolving Legacy Applications into Components", "Lifecycle Management of Components.
- (3.11.1) Invited talks given at ObjectWorld/East May.1996. Focus theme was "Transforming the Enterprise Investment into Component Software."
- (3.9,3.11.1) Workshop hosted by Logistics Management Institute, Evolving the Logistics Systems Investment to Component Software, I-Kinetics, NAVSEA presenting, April, 1996.

3.2.3 IONA Technologies, Inc.

Reported by Sean O'Sullivan, Business Development Manager, IONA

- (3.4) ORBIX 2.0, CORBA 2.0 compliant, completed and released to CWC.
- (3.4) ORBIX 2.02 with multi-threaded and full IIOP released to CWC.
- (3.3) ORBIXWeb 1.0 released.
- (3.4) CORBA Event and Name Service full release.

The CWC/TRP/OORAD is currently negotiating with IONA to expand the number of licenses available for Federal site CWC projects.

3.2.4 Pratt & Whitney

Reported by Robert Landgraff, Computer Specialist, Pratt & Whitney

- (3.10) Turbine Engine Agile Design VAW has progressed with first version rollout planned for end of January, 1996. To limited product rollout of 50 Unix workstations distributed equally at Florida and Connecticut sites. All goals and milestones

identified for the 2-year plan for the P&W site have been achieved. The system uses Task 3.3, 3.4 and 3.8 deliverables.

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

The resulting Airfoil design simulation has been deployed at Pratt&Whitney Florida and Connecticut sites. Pratt&Whitney is currently negotiating a 3000 node run-time license with IONA so as to achieve wide-availability of this and other planning CORBA-based systems. The next step in the Virtual Engine system is to expand the system to include more design simulation codes for other parts of the turbine design. The ultimate goal is to be able to completely simulate the entire turbine engine.

Above all, this project has impressed the importance of sound software engineering practices and object-oriented architecture. The use of CORBA has enabled the development team to focus on the domain problem, verses the development of a distributed computing infrastructure.

3.2.5 NAVSEA

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

(3.10) Stage 1 of the reengineering of a logistics management system (TLMS) has been completed. The system uses Task 3.4 and 3.8 deliverables.
(3.11.1) TLMS Stage 1 demonstrated and displayed at ObjectWorld East, May, 1996

4. Federal Dual Use

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

ARMY/ARDEC Automation and Robotics Design and Lifecycle Management of Real-Time Embedded Systems.

Funded by a SBIR Phase II, the I-Kinetics is transforming a set of design tools for ARDECs real-time embedded systems and reference architecture management environment into interoperable components. A key objective is to enable collaborative architecture design and specification among ARDEC clients. Another is to create a general framework for real-time systems design that is independent of any one tool and its specific specification model. This project is applying the products of task 3.1, 3.3, 3.4, 3.8, 3.11.

Update: 96.06.30

Work with ARDEC has led to the birth of a co-development and co-marketing partnership with ObjecTime. The ObjecTime product has two key capabilities: generation of C++ code from the Real-Time Object Model (ROOM), a state diagram based notation for real-time embedded systems, and the ability to simulate ROOM specifications of a RT system. The opportunity for I-Kinetics to enable a ROOM state node to be an actual CORBA-based component. An immediate expected benefit is that new systems capabilites can be verified from mixtures of new (unverified) and verified production system components without reducing the reliability of current production environment. New capabilities can be verified (or upgraded) by testing new components in a distributed testbed which shares most or all of the production real-time embedded system environment.

NASA JPL Small Mission Operations

Leveraging a SBIR Phase II, the NASA/JPL Small Mission Operations is benefiting from the CWC/TRP/OORAD joint venture R&D. I-Kinetics will build a Component Warehouse from JPL's substantial investment in spacecraft command and control, inter-planetary navigation, telemetry management, mission planning, and science package management. A key objective is to expand mission operations outside of the JPL site to the science community using the Internet. Pushing the management and analysis of the science packages out to the science teams is estimated to reduce JPL's mission support costs by an order of magnitude. This project will apply the products of task 3.1, 3.3, 3.4, and 3.8.

Update: 96.06.30

The JPL Small Mission Operation initiative has been operating for three months, with JPL HQ demonstrations planned for mid-September 1996. Science planning and spacecraft (S/C) planning and navigation systems assembled from JPL's legacy investment in Directional Pointer (Pointer), SC command and control (SEQGEN, SEQTRAN, CMD), Flight System Gateway (FST GW) are being packaged as CORBA based components.

Air Force, Phillips Lab, Satellite Telemetry Command & Control Mission Operations

Under a Air Force SBIR Phase I, the Air Force has tasked I-Kinetics with adding a Component Warehouse to their Satellite Telemetry, Command and Control (VISTA) testbed. Continuous availability is a key requirement for this mission-critical system. I-Kinetics will transform key parts of the STCC into replicated components. Component replication will enable critical STCC operations and services to continue when any one component fails. This project will apply the products of task 3.1, 3.3, 3.4, 3.5 and 3.8.

Update 6/30/96.
Since the start of this project, LTC Crowley has been promoted to Division Chief, of Missile, Space and Simulation Command. Additionally, the "Standard Satellite Control System" has been awarded to Loral/Loch-Mart. A final review of I-Kinetics Phase I results was conducted with VISTA program managers LTC Crowley and CPT. Russell. The decision was made to proceed to Phase II with the following milestones over the 2-year period:

1. M1: Transform the telemetry raw data transformation and analysis software into CORBA components (referred to as the Sofware Decomutation System (SDS).
2. M2: Enable the replication and management of SDS components.
3. M3: Enable the reconfiguration of a SDS client from a SDS source upon failure.
4. M4: Demonstrate C2 security capability for all components.

Air Force, Rome Laboratory, Satellite Image Analysis Operations

Under a Air Force SBIR Phase I the Air Force has tasked I-Kinetics with migrating a image analysis toolset to a set of components. The ability to access these components from CORBA, OLE or WWW client environments are key success criteria. This project will apply the products of task 3.1, 3.3, 3.4 and 3.8.

A.5 01 July - 30 Sep 96

Status on statement of work tasks from each CWC/TRP/OORAD joint venture member is given in the following sections. Each task item is noted as (3.x) and corresponds to the numbered SOW task items.

3.1 Dynamic Components and Component composability
~~**3.2 ORBitize™ : DOMS Interactive Development Environment**~~ **(Pushed into 3.3)**
3.3 Component Development Tools
3.4 CORBA Services and Facilities
3.5 Reliable Components
3.6 Security and Data Analysis Component

3.7 Joint Map-Server Component
3.8 "Plug&Play" Components
3.9 Information Factory for CALS
3.10 Information Factory for Concurrent Engineering
3.11 Interoperability
3.11.1 Evangelism, Workshops, Technology Transfer, Dual Use

3.1 General CWC/TRP/OORAD Activity and Significant Events

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.4) Orbix 2.0 was released to the CWC/TRP/OORAD community.
- (3.8) DataBroker 3.1 was released to the CWC/TRP/OORAD community.
- (3.11) The Virtual Enterprise Web came on-line with nodes hosted at BBN and I-Kinetics.
- (3.11) The first application demonstration has been rolled out to the CWC Virtual Enterprise Web. The demonstration shows a Java client and DataBroker server using IIOP. It is available at http://tesla.i-kinetics.com/dbcs_jfdemo/
- (3.11.1) I-Kinetics has been awarded a contract from Naval Air Command Center to design and prototype agent-based workflow for an object-based distributed system. The key innovation is to build agents dynamically from rule-based scripts and CORBA-based services and application components. This project will directly benefit from the technology and developed by the CWC. Pending program manager approval this project will join the other CWC/TRP/OORAD federal dual-use projects.
- (3.11.1) I-Kinetics has been awarded a contract from Naval Sea Command to design and prototype application component-based logistics document management system. This project will directly benefit from the technology and developed by the CWC. Pending program manager approval this project will join the other CWC/TRP/OORAD federal dual-use projects.
- (3.11.1) Oracle has joined Netscape in the adoption of CORBA for their distributed computing infrastructure. Oracle has termed their version of component software "Cartridges". CORBA has established a dominant base as the server backbone distributed computing infrastructure. The analysis of this event follows this report in Appendix A.

3.1.1 Upcoming Events

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.9,3.10, 3.11.1) 5-day Workshop and Training for "Transforming Legacy Investment into Application Components", scheduled for Nov. 11, 1996 at I-Kinetics, Burlington, MA.
- (3.11.1) CWC/TRP/OORAD Workshop Nov. 14, 15, 1996 at I-Kinetics, Burlington, MA,
- (3.11.1) 6 talks and technology demonstrations at ObjectWorld/West, August, 1996.
- (3.11.1) 1-day workshop to Air Force Phillips Lab, Space Command, Griffins Air Force Base, NM, November, 1996.
- (3.11.1) 1-day workshop to ARDEC, Intelligent Systems and Software, NM, November, 1996.
- Release of IONA Orbix 2.1. This release will improve the Orbix's IIOP compatibility.
- ComponentFactory 1.0 to be released in November to the CWC/TRP/OORAD community.
- DataBroker 3.1 was released to the CWC/TRP/OORAD community.

3.2 CWC/TRP/OORAD Member Activity

3.2.1 BBN

Reported by Rick Schantz, BBN

The objectives of BBN's work to date in the CWC/TRP/OORAD consortium have fallen into three broad areas.

Task 3.6 Security and Data Analysis Component

No activity this period.

Task 3.7 Joint Map-Server Component

A. Mapping Demo available within BBN Done.
B. Mapping Demo between 2 CWC sites (BBN and I-Kinetics)
Target: end of October
C. Mapping Demo between >2 CWC sites (BBN and I-Kinetics and other(s))
Target: TBD (optional)

Task 3.11 (Virtual) Enterprise (Web) Testbed

A. DataBroker demo within BBN
Target: mid November (possibly much sooner)
B. DataBroker demo between two CWC sites
Target: end of November
C. DataBroker demo between many CWC sites
Target: early December
D. Web accessible DataBroker demo available continuously
Target: mid February

Mapping Demo integration with Database Engine
Goal: replace Oracle calls in OpenMap with DataBroker, and hopefully use a 2nd database product underneath.

A. Mapping Demo integrated with DataBroker demo available within BBN
Target: end of January
B. Integrated Demo between CWC sites
Target: end of February
C. Web based integrated Demo available continuously
Target: end of March

3.2.2 I-Kinetics, Inc. (Lead)

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.11.1) Invited talks given at ObjectWorld/West May.1996. Of the 12 keynotes in the "Industry in Action", CWC was represented in 3. No other organization had more than one. The CWC has met and exceeded its original goals of building CORBA-based system success stories.
- (3.8) DataBroker 3.1 was released to the CWC/TRP/OORAD community.
- (3.11) The Virtual Enterprise Web came on-line with nodes hosted at BBN and I-Kinetics.
- (3.11) The first application demonstration has been rolled out to the CWC Virtual Enterprise Web. The demonstration shows a Java client and DataBroker server using IIOP. It is available at http://tesla.i-kinetics.com/dbcs_jfdemo.htm
- (3.11.1) Publications completed and released
    - ComponentWare Technology Vision and Product Roadmap White Paper.
    - DataBroker 3.1 White Paper
    - ComponentFactory (Early Peek) Technology White Paper
    - DataBroker Technology Plan
    - ComponentFactory Technology Plan
    - Evolving the Legacy Investment to Application Component Software White Paper
- (3.8) Completed implementation of I-Kinetics DataBroker V4.0. Released to SQA. Planned release to beta sites October 11, 1996.
- (3.3,3.8) I-Kinetics has entered into a strategic agreement with Netscape and NetDynamics to incorporate the CORBA-based ComponentWare technology with each company's Web server technology.
- (3.9) Off-site and on-site technical consulting to support the NAVSEA host site.
- (3.10) Off-site and on-site technical consulting to support the P&W host site.
- (3.11) Joint submission to OMG RFP4, CORBA Common Facilities, Data Interchange Facility (DIF), with MITRE Corporation and Objectivity. DIF has passed technical review and has been submitted to OMG Technical Committee for final review.

3.2.3 IONA Technologies, Inc.

Reported by Sean O'Sullivan, Business Development Manager, IONA

- (3.4) ORBIX 2.0, CORBA 2.0 compliant, completed and released to CWC.

- (3.4) OrbixWeb 2.0 released.

3.2.4 Pratt & Whitney

Reported by Robert Landgraff, Computer Specialist, Pratt & Whitney

Now that the Virtual Jet Engine Program is being deployed at multiple sites at Pratt & Whitney, CWC/TRP/OORAD joint funding will no longer be used. We will continue to develop and expand the Virtual Engineering infrastructure beyond the Virtual Jet Engine Program as a strategic competitive program fully funded by Pratt & Whitney.

Working with I-Kinetics, we have identified a new challenge project for Year 2. Currently, our information systems visionaries are starting Intranet pilots that combine the CORBA, JAVA and component software technologies.

The first project for Year-2 is to add new capability to the Cost Schedule Performance (CPS) system. The urgency is that by October 17, 1996 we have to decide on the architecture of
the entire CPS application and determine if it is ready to support the Joint Strike Fighter Program. (Alias JAST).

In joint partnership with I-Kinetics, we are developing a CPS application prototype. The application. Our vision would be to input via the Java Form and get the output back into an Excel spreadsheet from which the user could generate his own reports/charts. They have strong hopes that this technology is the right fit. CWC/TRP/OORAD technology that will be utilized are I-Kinetics DataBroker (formerly Database Component Server) and IONA's OrbixWeb.

3.2.5 NAVSEA

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

(3.10) NUWC partner organization was undergoing reorganization during this quarter and will resume full activity at the end of October, 1996.

4. Federal Dual Use

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

ARMY/ARDEC Automation and Robotics Design and Lifecycle Management of Real-Time Embedded Systems.

Funded by a SBIR Phase II, the I-Kinetics is transforming a set of design tools for ARDECs real-time embedded systems and reference architecture management environment into interoperable components. A key objective is to enable collaborative architecture design and specification among ARDEC clients. Another is to create a general framework for real-time systems design that is independent of any one tool and its specific specification model. This project is applying the products of task 3.1, 3.3, 3.4, 3.8, 3.11.

Update: 96.09.30

Work with ARDEC has led to the birth of a co-development and co-marketing partnership with ObjecTime. The ObjecTime CORBA server has been completed.

NASA JPL Small Mission Operations

Leveraging a SBIR Phase II, the NASA/JPL Small Mission Operations is benefiting from the CWC/TRP/OORAD joint venture R&D. I-Kinetics will build a Component Warehouse from JPL's substantial investment in spacecraft command and control, inter-planetary navigation, telemetry management, mission planning, and science package management. A key objective is to expand mission operations outside of the JPL site to the science community using the Internet. Pushing the management and analysis of the science packages out to the science teams is estimated to reduce JPL's mission support costs by an order of magnitude. This project will apply the products of task 3.1, 3.3, 3.4, and 3.8.

Update: 96.09.30

The JPL Small Mission Operation initiative has been operating for three months, with JPL HQ demonstrations completed in mid-September 1996.

I-Kinetics participated in the July Small Missions Operations Working Group meeting held at Stanford University. I-Kinetics personnel traveled to JPL in July to install the Orbix CORBA implementation and retrieve updated copies of the TDS software and related support modules. By early August the JPL environment was replicated at I-Kinetics, along with radio science data processing applications provided by Dick Simpson of Stanford.

I-Kinetics worked with Jim Grimes of JPL to develop an object model for the downlinked telemetry data and the radio science processing domain. This was completed in August, and the necessary legacy applications (MicroTOT and TDS from JPL, Prepmo and Preplook from Stanford) were encapsulated to support this object model. IDL describing the object model interface was supplied to the working group for integration with client applications being developed by other contractors.

I-Kinetics assisted with the integration by providing a Java client application which manipulated the object model via its IDL interface. This was used by the GUI provided by Reticular Systems to extract data from the TDS and populate the radio science objects in a successful demonstration held at JPL on 11 September.

I-Kinetics presented two talks at the September demonstration: One talk focused on legacy application integration and encapsulation as a transition mechanism to a distributed object system, while the second talk explained the architecture of the demonstrated object model and its implementation with CORBA.

Air Force, Phillips Lab, Satellite Telemetry Command & Control Mission Operations

Under a Air Force SBIR Phase I, the Air Force has tasked I-Kinetics with adding a Component Warehouse to their Satellite Telemetry, Command and Control (VISTA) testbed. Continuous availability is a key requirement for this mission-critical system. I-Kinetics will transform key parts of the STCC into replicated components. Component replication will enable critical STCC operations and services to continue when any one component fails. This project will apply the products of task 3.1, 3.3, 3.4, 3.5 and 3.8.

Update: 96.09.30

The Phase II plan has been approved and completed with the following major milestones over the 2-year period:

1. M1: Transform the telemetry raw data transformation and analysis software into CORBA components (referred to as the Software Decomutation System (SDS).
2. M2: Enable the replication and management of SDS components.
3. M3: Enable the reconfiguration of a SDS client from a SDS source upon failure.
4. M4: Demonstrate C2 security capability for all components.

Air Force, Rome Laboratory, Satellite Image Analysis Operations

Under a Air Force SBIR Phase I the Air Force has tasked I-Kinetics with migrating a image analysis toolset to a set of components. The ability to access these components from CORBA, OLE or WWW client environments are key success criteria. This project will apply the products of task 3.1, 3.3, 3.4 and 3.8.

Update: 96.09.30

I-Kinetics has completed the Phase I deliverables and milestones. Rome Image Lab personnel visited I-Kinetics September 29, 1996 for the final review. The Rome Image Lab Toolkit was transformed into a CORBA-based server object as well at MatLab image analysis module. Full interoperability and integration with an image exploitation workbench was shown.

A.6 01 Oct - 31 Dec 96

Status on statement of work tasks from each CWC/TRP/OORAD joint venture member is given in the following sections. Each task item is noted as (3.x) and corresponds to the numbered SOW task items.

3.1 Dynamic Components and Component composability

~~**3.2 ORBitize™ : DOMS Interactive Development Environment**~~ **(Pushed into 3.3)**

3.3 Component Development Tools

3.4 CORBA Services and Facilities

3.5 Reliable Components

3.6 Security and Data Analysis Component

3.7 Joint Map-Server Component

3.8 "Plug&Play" Components

3.9 Information Factory for CALS

3.10 Information Factory for Concurrent Engineering
3.11 Interoperability
3.11.1 Evangelism, Workshops, Technology Transfer, Dual Use

3.1 General CWC/TRP/OORAD Activity and Significant Events

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.3) Orbix 2.1 was released to the CWC/TRP/OORAD community.
- (3.3) OrbixWeb 2.0 was released to the CWC/TRP/OORAD community
- (3.4) IONA and Transarc released CORBA Object Transaction Service beta
- (3.8) DataBroker 4.0 was released to the CWC/TRP/OORAD community.
- (3.7,3.11) OpenMap has been rolled out to the CWC Virtual Enterprise Web. It is available at http://bbncwc1.bbn.com:8000/trp/intro.html

The following press release from the OMG summarizes the growing momentum behind CORBA and enterprise component software and the significant increase in interest of the CWC/TRP/OORAD venture.

OMG Press Release, 12/4/96
Distributed object computing has emerged as the next phase of the software revolution. As the Internet becomes the network for an ever increasing number of businesses it becomes imperative that business information systems are based on industry standard component software. Corporate developers and end users are turning to component software and services that incorporate the standards of the Object Management Group to solve problems such as legacy systems integration, complex data- sharing, and distributed computing via the Internet. Distributed systems based on OMG's Common Object Request Broker Architecture and the Internet Inter-ORB Protocol (IIOP) are fast becoming as ubiquitous as those using HTTP, CGI, and TCP/IP.

Recent market research collected by the OMG indicates that the trend toward business use of CORBA/IIOP based software will soon become the dominant back-end computing architecture. Chris Stone, President and CEO of the Object Management Group, explains, "In any revolution there comes a pivotal moment where success or failure is determined. I believe that the distributed object computing revolution has successfully passed this moment and is on the fast track to near-universal acceptance. The internet has revolutionized business networks, and CORBA and IIOP are helping businesses take advantage of this new networking architecture."

3.1.1 Recent & Upcoming Events

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.9,3.10, 3.11.1) 2-day CWC/TRP/OORAD Project Review and Planning March 19,20, 1997, West Palm Beach, Florida.
- (3.11.1) CWC/TRP/OORAD Workshop Nov. 14, 15, 1996 at I-Kinetics, Burlington, MA,
- (3.11.1) CWC/TRP/OORAD presentations and technology demonstrations at ObjectWorld/East, March 3-9, 1997.

- (3.11.1) 1-day workshop to Air Force Phillips Lab, Space Command, Griffins Air Force Base, NM, November, 1996.
- (3.11.1) 1-day workshop to ARDEC, Intelligent Systems and Software, NM, November, 1996.
- ComponentFactory 1.0 to be released in November to the CWC/TRP/OORAD community.
- Open JDBC Component V1.0 March, 1997.

3.2 CWC/TRP/OORAD Member Activity

3.2.1 BBN

Reported by Rick Schantz, BBN

- (3.7,3.11) OpenMap has been rolled out to the CWC Virtual Enterprise Web. It is available at http://bbncwc1.bbn.com:8000/www/intro.html

Task 3.6 Security and Data Analysis Component

No activity this period.

Task 3.7 Joint Map-Server Component

Completed A. Mapping Demo available within BBN.
Completed B. Mapping Demo between 2 CWC sites (BBN and I-Kinetics)
Incomplete C. Mapping Demo between >2 CWC sites (BBN and I-Kinetics and other(s)).
Pending any further activity awaiting access to a host machine

Task 3.11 (Virtual) Enterprise (Web) Testbed

Completed A. DataBroker demo within BBN
Completed B. DataBroker demo between two CWC sites
Completed C. DataBroker demo between many CWC sites
Completed D. Web accessible DataBroker demo available continuously

Reference:
http://bbncwc1.bbn.com:8000/trp/tasks.html
http://bbncwc1.bbn.com:8000/www/intro.htm

3.2.2 I-Kinetics, Inc. (Lead)

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.8) DataBroker 4.0 was released to the CWC/TRP/OORAD community.
- (3.11.1) Publications completed and released
  - ComponentWare Technology Vision and Product Roadmap White Paper V2.0
  - DataBroker 4.0 White Paper
  - ComponentFactory Case Study: Image Exploitation
- (3.3,3.8) I-Kinetics has entered into a strategic agreement with Netscape and NetDynamics to incorporate the CORBA-based ComponentWare technology with each company's Web server technology.
- (3.10) Off-site and on-site technical consulting to support the P&W host site.

We have agreement with Netscape and Oracle on a product bundling partnership using the DataBroker. Since I-Kinetics is in the lead with a full multi-threaded, CORBA server based database driver manager, we have the opportunity to offer a JDBC implementation using IIOP. This would be the first ubiquitous, fully standard implementation, eliminating all points of proprietary implementation for Java access to databases.

The next big deal opportunity in our sights is the ComponentFactory/Perl Application Wrapping Toolkit -- Enables developers to transform any Perl script into a CORBA object server. Each vendors Web Server product can be the first to offer a complete upgrade path to distributed objects for the existing investment in Perl Web scripts.

Mike Higgs, V.P. Product Development writes about Stone-Axe (ComponentFactory V.1) from his status report...

We evolved the design and architecture significantly. In particular we've redefined ComponentFactory as a component based architecture, with IDL defined sub-components for persistence, code generation and "Binders". This supports (eventually) user extensibility, rapid prototyping of ComponentFactory features using ComponentFactory and retargetting ComponentFactory at different infrastructures (CORBA, DCOM, Web Servers, etc).

The next build will be F, The major objective of the F build is to complete the internal round-trip (building Stone-Axe using Stone-Axe) and to start incorporating the "canonical component" functionality/pattern which is being prototyped to support this round-trip.

Note: ComponentFactory is a tool for building components. Using ComponentFactory to build itself is especially powerful.

3.2.3 IONA Technologies, Inc.

Reported by Sean O'Sullivan, Business Development Manager, IONA

- (3.3) ORBIX 2.1, CORBA 2.0 compliant, completed and released to CWC.
- (3.3) OrbixWeb 2.0 released 12/96.
- (3.4) CORBA Names Service released 11/96.
- (3.4) CORBA Events Service released 11/96.
- (3.4) IONA and Transarc released CORBA Object Transaction Service beta

Transarc Corporation and IONA Technologies announced their agreement and plans on 12/10/96 to develop an integration between the Encina Object Transaction Service (OTS) and the Orbix object request broker to produce a fully CORBA-compliant OTS. The integration of the two products will conform to the Object Management Group's (OMG's) specified Object Transaction Service, which was written by a coalition of vendors, including Transarc and IONA.

The Object Transaction Service standardizes distributed object transactions and complements the CORBA (Common Object Request Broker Architecture) specification.

With the evolution of a commercially available solution for the OTS, customers can build and implement robust distributed objects over intranets and the Internet.

The joint solution for distributed object transactions currently is in limited beta and is distributed by IONA and Transarc. General availability will be announced at a later date. Initial platform support includes Solaris, HP-UX, AIX and Windows NT.

3.2.4 Pratt & Whitney

Reported by Robert Landgraff, Computer Specialist, Pratt & Whitney

Working with I-Kinetics, we have identified a new challenge project for Year 2. Currently, our information systems visionaries are starting Intranet pilots that combine the CORBA, JAVA and component software technologies.

The first project for Year-2 is to add new capability to the Cost Schedule Performance (CPS) system. The urgency was that by October 17, 1996 we have to decide on the architecture of the entire CPS application and determine if it is ready to support the Joint Strike Fighter Program. (Alias JAST).

In joint partnership with I-Kinetics, we are developing a CPS application prototype. The application. Our vision would be to input via the Java Form and get the output back into an Excel spreadsheet from which the user could generate his own reports/charts. They have strong hopes that this technology is the right fit. CWC/TRP/OORAD technology that will be utilized are I-Kinetics DataBroker (formerly Database Component Server) and IONA's OrbixWeb.

Pratt & Whitney uses a mainframe application called CPSIII to accomplish Cost Schedule Performance reporting. With the internal mandate to off-load mainframe usage to Sun Workstations and PC Desktops, cost performance information is periodically downloaded and warehoused into Oracle 7 on an Sun Ultra Enterprise server.

The application uses the Java Workshops Visual Java classes to present the user with a graphical user interface that accepts various input allowing the user to generate a variety of adhoc report formats. Oracle authentication and data access is performed through the use of DataBroker 4.0. It's multi-threaded capabilities are essential, since 20 or more concurrent users can be expected to be using the system during peak cost reporting periods.

This application was successfully demonstrated at the United Technologies Engineering Conference (UTECA) in December. With follow-on support through the CWC and direct contracting, this application will be the first of a wave of application development based on Web and CORBA technologies.

3.2.5 NAVSEA

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

(3.10) NUWC partner organization was undergoing reorganization during this quarter and will resume full activity at the end of October, 1996.
(3.11)

4. Commercial and Federal Dual Use

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

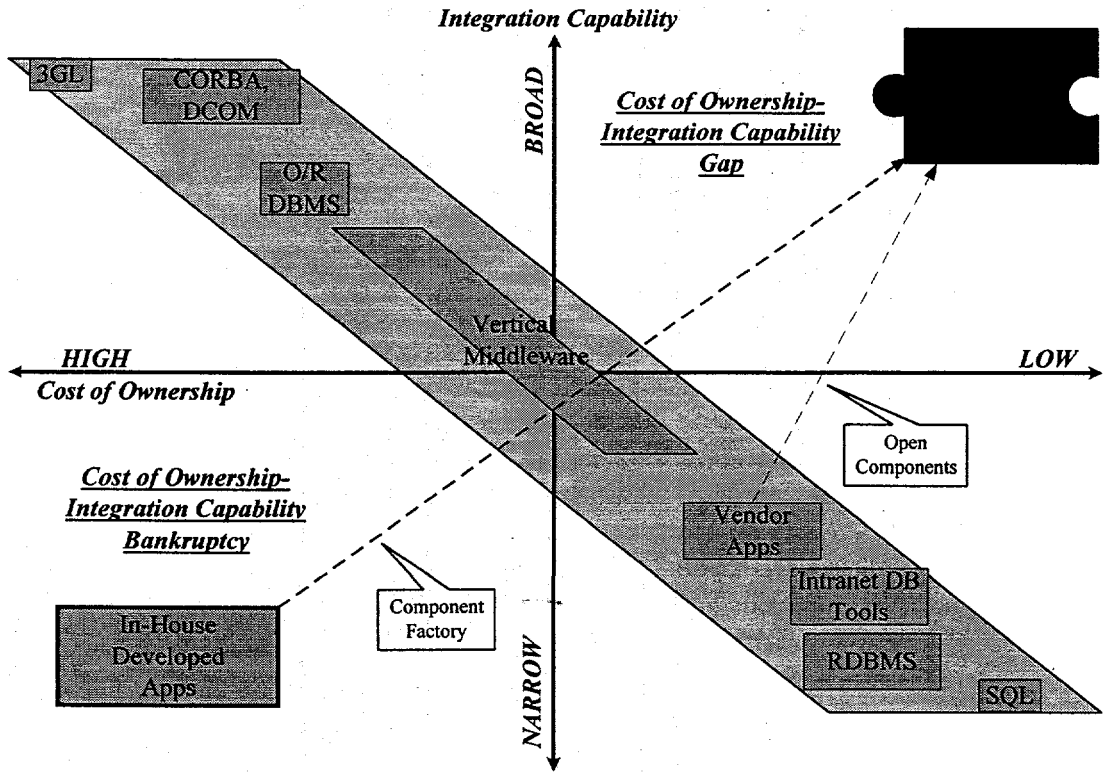
There exists a tradeoff between cost of ownership and the integration capability of migrating existing enterprise applications into enterprise components. Current CORBA tools enable almost any application to be converted into an enterprise component. However, CORBA tools, such as IDL/C++ compilers, have high cost of ownership because of intensive coding and large administration and maintenance costs.

The leading Intranet DB Tools, such as PowerSoft's NetImpact, Bluestone's Web/Sapphire, and NetDynamics have a low cost of ownership. These tools supply full "point and click" integration with relational database management systems such as Oracle, Sybase, Informix, DB2, and ODBC capable data sources.

However, the Intranet DB Tool has accepted a very narrow range of integration capability to keep its low cost of ownership. Each tool supports one or more languages (C, C++, VB, Java) for "business logic" extensions at the server. This is the method by which adapters or gateways integrate to non-RDBMS applications can be accomplished. This level of integration requires custom manual coding and a resulting high cost of ownership.

Over the years, companies have acquired a vast amount of in house developed applications that represent user, departmental, and enterprise business knowledge. Additionally, their Information Technology (IT) has developed a skill set in building these systems. These applications are characterized by a high cost of ownership and a narrow, even brittle, integration capability.

The existing investment in applications and information technology (IT) staff represent a vast resource of business and knowledge capital that a enterprise must reuse. At the same time, the enterprise needs to incorporate the benefits of component-based applications. This need is the Cost of Ownership-Integration Capability Gap.



I-Kinetics is filling the Cost of Ownership-Integration Capability Gap with the following products and services:

- ComponentFactory™ , a development tool for transforming UNIX and Windows applications into CORBA application components, without having to change source code, or disturbing the application in its production environment.
- DataBroker™ is a CORBA application object providing universal access for the CORBA-capable application to record-oriented data and applications which generate record-oriented data.
- Open Components: Package existing high-value UNIX and Windows NT applications into CORBA-capable plug&play application components.
- I-Kinetics Professional Services works with clients to build adaptive information systems for competitive advantage. Our leadership in technology, methodology and training for building and managing very large component-based distributed systems is a key asset to our customers in assuring early adopter success.

Hughes Corporate IS, Enterprise Infrastructure Consulting
Situation: Design to Cost initiative pilot project supporting the next generation Interface Technology corporate initiative.

Critical Issue: Pressure from Sr. Management to reduce software development costs and move to COTS solutions.

Reasons: High cost of adhoc software development in a downsizing environment.

Vision: To move to a 100% COTS solution.

Provided (Solution): A proof of concept prototype application has prompted P&W to purchase I-Kinetics Consulting Services to provide a production Program Planning and Control report generation application.

Result: I-Kinetics and IONA selected as COTS tool vendors.

Next Opportunity: The success of this pilot will no doubt lead to large software and consulting sales.

Sandia Labs, DataBroker 4.0
Situation: Team Leader, Enterprise Integration project for next generation energy management systems. Purchased 3 licenses of DataBroker for pilot project.

Critical Issue: Cost pressures and capability pressures from senior management to reduce cost of software systems development.

Reasons: Developers were doing to much ad-hoc "stove pipe" integration with databases, taking away resources from developing actual systems.

Vision: He (Bob Whiteside) told us they needed a standard means for accessing databases within an overall software infrastructure, that could be used across many differing applications.

Provided (Solution): We provided them with the DataBroker which gave them one consistent means for accessing databases in a common software framework.

Result: Development efforts were reduced by an order of magnitude for database access across applications and languages.

Lockheed IRAD, Enterprise Infrastructure Consulting
Situation: Project Manager, Satellite Systems

Critical Issue: Pressure from management to reuse existing systems, reduce development costs, and improve productivity with shrinking staff.

Reasons: Budgetary cuts and too much devoted to developing distributed systems as opposed to the actual systems themselves.

Vision: They told us they needed a way to reduce the risk of moving to new technologies for achieving reduced development times and resue.

Provided (Solution): We provided them with the capability to reuse existing systems, mentor personnel, and reduce development times, reducing project risk.

Result: Team is able to develop new, distributed systems with reduced costs and development times, using reduced staff.

Air Force/Hanscom DataBroker 4.0
Situation: Prism Consortium, United States Air Force standards body for COTS software

Critical Issue: Upper management wants to reduce custom development and move to COTS model, no more "re-inventing the wheel." Cost pressures are forcing cuts in staff while output requirements are rising.

Reasons: Staff was developing stove pipe applications. An example is is writing their own database independent middleware and wrappers. Too costly too maintain.

Vision: He (Bruce Swanson) told us they needed COTS solutions for database access that were compliant with new software infrastrucutre standards, eliminating the need to develop this software.

Provided (Solution): We provided him with the DataBroker for database access via CORBA.

Result: Hanscom was able to replace their proprietary, in house developed middleware with open standards and a COTS product for database access. This significantly reduced short term and long term development costs.

A.7 01 Jan - 31 Mar 97

Status on statement of work tasks from each CWC/TRP/OORAD joint venture member is given in the following sections. Each task item is noted as (3.x) and corresponds to the numbered SOW task items.

3.1 Dynamic Components and Component composability
~~**3.2 ORBitize™: DOMS Interactive Development Environment**~~ **(Pushed into 3.3)**
3.3 Component Development Tools
3.4 CORBA Services and Facilities
3.5 Reliable Components
3.6 Security and Data Analysis Component
3.7 Joint Map-Server Component
3.8 "Plug&Play" Components
3.9 Information Factory for CALS
3.10 Information Factory for Concurrent Engineering
3.11 Interoperability
3.11.1 Evangelism, Workshops, Technology Transfer, Dual Use

3.1 General CWC/TRP/OORAD Activity and Significant Events

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.3) Orbix 2.1 was released to the CWC/TRP/OORAD community.
- (3.3) OrbixWeb 2.1 was released to the CWC/TRP/OORAD community

- (3.4) IONA and Transarc released CORBA Object Transaction Service beta
- (3.8) DataBroker 5.0 entered final QA.
- (3.9,3.11) I-Kinetics DataBroker won best new Component at the 1997 ObjectWorld Conference.
- (3.11) I-Kinetics has been awarded a 2-year effort to broaden the use of distributed component software for Air Force Satellite Telemetry Mission Operations
- (3.9,3.10, 3.11.1) 2-day CWC/TRP/OORAD Project Review and Planning March 19,20, 1996, West Palm Beach, Florida.
- (3.11.1) CWC/TRP/OORAD presentations and technology demonstrations at ObjectWorld/East, March 3-9, 1996.
- BBN completed CTO efforts.
- IONA Technologies went public with revenue of $21,000,000 in 1996 and a market capitalization of $350,000,000. This marks significant commercial success of the CTO effort.

3.1.1 Upcoming Events

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- DataBroker 5.0/Open JDBC Component to be released April, 1997.
- Orbix 2.2 to be released June, 1997.
- (3.3) Orbix Trader service released to the CWC/TRP/OORAD community
- (3.9) Pratt & Whitney Composite Material Design project to start.
- I-Kinetics, IONA and P&W to wind down CTO efforts.
- (3.11) I-Kinetics and JPL are presently planning a 2nd continuing 2-year effort to broaden the use of distributed component software for Small Mission Operations
- (3.11) I-Kinetics and ARDEC are presently planning a 2-year effort to build a testbed for real-time CORBA integration and deployment investigations.
- (3.11) I-Kinetics will release to an estimated 400 sites DataBroker 5.0 in April, 1997. This marks significant commercial success of the CTO effort.

3.2 CWC/TRP/OORAD Member Activity

3.2.1 BBN

Reported by Rick Schantz, BBN

- (3.7,3.11) OpenMap has been rolled out to the CWC Virtual Enterprise Web. It is available at http://bbncwc1.bbn.com:8000/trp/intro.html
- All work by BBN in the CWC/TRP/OORAD has been completed as of workshop. BBN made final report at this workshop.

3.2.2 I-Kinetics, Inc. (Lead)

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- (3.8) DataBroker 4.0 was released to the CWC/TRP/OORAD community.
- (3.11.1) Publications completed and released
  - ComponentWare Technology Vision and Product Roadmap White Paper V2.0
  - DataBroker 5.0 White Paper
  - Legacy Integration
- (3.10) Off-site and on-site technical consulting to support the P&W host site.
- I-Kinetics has been awarded a 2-year effort to broaden the use of distributed component software for Air Force Satellite Telemetry Mission Operations

3.2.3 IONA Technologies, Inc.

Reported by Sean O'Sullivan, Business Development Manager, IONA

- (3.3) ORBIX 2.1, CORBA 2.0 compliant, completed and released to CWC.
- (3.3) OrbixWeb 2.1 released 3/97.
- (3.4) CORBA Names Service 2.0 released 3/97.
- (3.4) CORBA Events Service 2.0 released 3/97.
- (3.4) IONA and Transarc released CORBA Object Transaction Service Beta 2.

3.2.4 Pratt & Whitney

Reported by Robert Landgraff, Computer Specialist, Pratt & Whitney

- Pratt & Whitney Composite Material Design project to start.

Working with I-Kinetics, we have identified a new challenge project for Year 2. Currently, our information systems visionaries are starting Intranet pilots that combine the CORBA, JAVA and component software technologies.

To create a user friendly Composite Material Input program to be used by all UTC composite groups to pre-process composite material properties into proper format for analysis. The program will have a platform independent GUI (Graphic User Interface) for the user to input the information necessary configure the selected analysis to meet the users input (Vibra, UG, Catia) and output (Ansys, Nastran) requirements. The program will wrap existing legacy code as well as new code written to provide accurate material properties for 2 or 3d configurations. It will be written in modular (object oriented) fashion to allow incremental release of capabilities and easy upgrades without adversely affecting the user. The scientific program analyses will reside on dedicated servers, accessed with the GUI interface from the client machines.

Work Required (Composites Engineering)

Define and create a Demo version of the GUI interface. The interface would have all the visual features (buttons, windows, etc.) that the final working version has. The purpose of
creating the working demo version is to create and modify the feel of program, without costly iterations of background code.

Modify existing core programs to do what is required (Blades and Vanes). Combine the capabilities of the Hamilton Blade design program and Lamina.

Add an optimization routine (like TURBO) to reduce iterations required of the operator.
Modify WeaveMat to run in the batch mode.
Add capability to have platforms and multiple airfoils.

Create or acquire a ply generation routine which will calculate ply angles as fabric draped over the mold. The routine should take an existing finite element breakup and apply the ply properties to it (thickness and angle). The Lamina program should take the output and create the property cards for all parts. A cross link routine should be created to iterate ply layup until parameters are acceptable.

3.2.5 NAVSEA

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

(3.10) NAVSEA has identified new projects and switched their site from NUWC to Business Systems, NAVSEA. They will resume full activity in Q2, 1997.

4. Commercial and Federal Dual Use

Reported by Bruce H. Cottman, Chairman, CWC/TRP/OORAD Joint Venture

- IONA Technologies went public with revenue of $21,000,000 in 1996 and a market capitalization of $350,000,000. This marks significant commercial success of the CTO effort.

- (3.11) I-Kinetics will release to an estimated 400 sites DataBroker 5.0 in April, 1997. This marks significant commercial success of the CTO effort.

ARMY/ARDEC Automation and Robotics Design and Lifecycle Management of Real-Time Embedded Systems.

Funded by a SBIR Phase II, I-Kinetics is transforming a set of design tools for ARDECs real-time embedded systems and reference architecture management environment into interoperable components. A key objective is to enable collaborative architecture design and specification among ARDEC clients. Another is to create a general framework for real-time systems design that is independent of any one tool and its specific specification model. This project is applying the products of task 3.1, 3.3, 3.4, 3.8, 3.11.

Update: 3/30/97

I-Kinetics and ARDEC are presently planning a 2-year effort to build a testbed for real-time CORBA integration and deployment investigations.

NASA JPL Small Mission Operations

Leveraging a SBIR Phase II, the NASA/JPL Small Mission Operations is benefiting from the CWC/TRP/OORAD joint venture R&D. I-Kinetics will build a Component Warehouse from JPL's substantial investment in spacecraft command and control, inter-planetary navigation, telemetry management, mission planning, and science package management. A key objective is to expand mission operations outside of the JPL site to the science community using the Internet. Pushing the management and analysis of the science packages out to the science teams is estimated to reduce JPL's mission support costs by an order of magnitude. This project will apply the products of task 3.1, 3.3, 3.4, and 3.8.

Update: 3/30/97

I-Kinetics and JPL are presently planning a 2nd continuing 2-year effort to broaden the use of distributed component software for Small Mission Operations

Air Force, Phillips Lab, Satellite Telemetry Command & Control Mission Operations

Under a Air Force SBIR Phase I, the Air Force has tasked I-Kinetics with adding a Component Warehouse to their Satellite Telemetry, Command and Control (VISTA) testbed. Continuous availability is a key requirement for this mission-critical system. I-Kinetics will transform key parts of the STCC into replicated components. Component replication will enable critical STCC operations and services to continue when any one component fails. This project will apply the products of task 3.1, 3.3, 3.4, 3.5 and 3.8.

Update: 3/30/97

I-Kinetics has been awarded a 2-year effort to broaden the use of distributed component software for Air Force Satellite Telemetry Mission Operations

B. Reference Component Architecture Tools Capabilities Analysis

Analysis Summary

There are a number of major technology trends that will increasingly impact and shape the OO CASE tool market for the next 3 years:

- **De-facto Standardization of Middleware:** Middleware can be loosely described as the software to glue a new application with the existing installed base of software. The Intranet platform and software vendors are rapidly converging on two middleware choices: OMG's CORBA and Microsoft's DCOM. *Starting in ~~1~~997 and beyond it will become increasingly difficult to sell applications that are not CORBA or DCOM capable.*
- **De-facto Standardization of OO CASE Tool Notation:** UML is fast becoming the industry standard analysis and design notation for OO CASE tools. It provides common notation, terminology, and semantics for object-oriented analysis and design. This standardization will greatly accelerate the acceptance of object-oriented approaches and technologies, because it protects an organization's previous object-oriented investments by limiting maintenance costs and training time and ensuring forward compatibility. *Starting in 1997 and beyond it will become increasingly difficult to sell OO CASE tools that are not UML capable.*
- **De-facto Standardization of OO CASE Tool Repository:** At the end of 1997 there will be two de-facto standards for OO CASE repositories: Microsoft Repository and OMG Repository. The industry further expects that these two repositories will be interoperable by directly supporting UML.

The resurrected CASE market seems set for a new period of growth and expansion. It is predicted that by Q4'97 there will be an adopted OMG specification for OO notation that will be based on UML. Additionally, there will be two dominant specifications for a standard CASE tool repository, the OMG repository and the MS repository. Both promise to be interoperable through the exchange of UML-based meta-data.

The promise of component-based applications is to enable developers to "snap together" applications by mixing and matching prefabricated software components. However, the lack of standard middleware, CASE tool notation and CASE tool repository has limited the realization of component software. UML-based CASE Tools using either the MS Repository or the OMG repository available for less than $1,000 per seat, will change the market dramatically and catalyze component reuse in the enterprise.

Identified Risks

The identified risks in achieving the technical objectives are:

- The estimated quality and time of availability of UML capable CASE tools is subject to the plans of independent software vendors.
- All UML capable CASE tools, with the exception of Rational Rose, are in the alpha or beta stage or have not been released. Only the MS Repository is available. I-

Kinetics may not be able to obtain early release versions of their CASE tool or the vendor may charge fees that are not supportable within the scope of this SBIR effort.

- The capabilities of the MS Repository may prove to be extremely limited. The early release quality of the candidate UML-capable CASE tools or CASE repositories may be so poor as to be unusable and unsupportable within the scope of future work.

The First CASE Tool Generation: Structured

The first generation of commercial CASE (computer-aided software engineering) tools appeared in the mid-1980's, following the widespread acceptance of the structured methodologies created and popularized by Ed Yourdon, Tom DeMarco, and James Martin. The promise of CASE tools was that they could automate the software lifecycle given a complete design specification.

A design was specified using structural analysis and design (A&D) diagrams which in turn specified procedural programming constructs. These constructs and their relationships were then used to generate the software code. All maintenance, logic changes, and new feature addition were accomplished by changing the design in the CASE tool and then generating the updated code.

Originally there were two general types of CASE tools:

- analysis and design, the front-end CASE tool (CASE/AD)
- code generation from structured design (CASE/G).

By the late 1980's there were a few comprehensive CASE tools that did both A&D and code generation (CASE/ADG). By 1990, KnowledgeWare's product was the most widely used of the comprehensive CASE tools and Texas Instrument's IEF was the best. Developers who wanted to use IEF effectively needed to follow a rather rigorous methodology, but those who did created some large and complex systems. The developers created the structured diagrams that described the system and its interfaces, and the IEF tool generated all the code for the system.

However by 1991, the "re-engineering the enterprise" was at its peak in the Fortune 1000 created and CASE tool users demanded new capabilities:

- **Structured Notation for Distributed Applications Problem:** Migration from mainframes and VAX/VMS platforms to UNIX was on the rise. CASE tools needed to support new types of architectures, such as client-server and embedded real-time applications. However, the structured methodologies that existed implicitly assumed that an application was a monolith that ran on one platform. CASE found itself with no formal notation for distributed applications. The leading vendors created their own ad-hoc, ill-defined proprietary notations.
- **Tool Interoperability Problem:** New CASE tools appeared to fill the gap, each with their own proprietary method for handling a narrow niche of client/server and real-time architecture patterns. The enterprise found itself buying multiple tools to support their systems. However, the promise of complete software lifecycle automation was broken. As the application evolved it would jump out of its original

architecture pattern into several architecture patterns. Using multiple CASE tools to support multiple architecture design patterns was not possible as the tools were proprietary and could not share a common application design.

- **Extracting Design from Legacy Code Problem:** There was great demand was for a CASE/ADG tool that could be run "backwards". Essentially 100% of the production code had been created manually and not from an existing structured design diagram. The market wanted a tool that could automatically walk over code and extract architecture patterns and from that back-engineer the complete design. There was no solution known at this time and still none today. All commercially popular languages are incapable of capturing design specification. The legacy base of code does not intrinsically have enough information to extract design. The study of languages that can capture design is an active exploratory field of research today. Extracting design from legacy code is a very strenuous cognitive process. Additionally, the original design is not actually extracted, but rather a new design is produced that supports the target functional specification of the application. However, this did not stop the CASE vendors from trying to sell "re-engineering" CASE tools.

IBM in the late 1980's recognized and decided to solve the tool interoperability problem with a standardized design repository approach. An intrinsic component of any CASE tool is a repository -- a database in which all of the analysis, designs, and coding elements are stored. A good repository keeps track of all the interdependencies between all the diagrams created in the tool. Thus a developer can change the names and relationships between two entities on one diagram and see the change reflected in all the other diagrams in which those entities occur. In the late 1980's, IBM began the AD/Cycle repository project. Its goal was to create a single repository that all CASE products could use. With such a repository, it was assumed that a company could use diagrams created on one CASE tool to generate subsequent applications in another CASE tool.

IBM's killed off the AD/Cycle repository project in 1991. New types of CASE tools that could handle new types of architectures, such as client-server and embedded real-time applications were appearing. Also, object-based methodologies and OO development systems were starting to emerge. IBM was overwhelmed with the number of different tools, approaches, data types, and A&D constructs the AD/Cycle repository was being asked to accommodate. The commercial market for this generation of CASE tools collapsed in 1991.

The 2nd CASE Tool Generation: Object Oriented

The early 1990's witnessed a proliferation of OO methodologies, each with it's own notation, and most popular OO CASE tools have arranged to support all of the popular ones. Thus in 1995, an OO CASE tools like Paradigm Plus, Rational Rose and Software through Pictures (StP) supported Booch, Rumbaugh (OMT), Jacobson (Use Case), Martin/Odell, and Shlaer-Mellor, to name only a few.

The 3rd CASE Tool Generation: Unified and Components

In 1995, one of the leading OO CASE vendors, Rational Software, which employed Grady Booch and supported Booch's methodology, hired James Rumbaugh and

announced that the two methodologists would like would work together on a common OO methodology. In 1996, after they had released the first draft of their Unified Method (UM), Rational hired Ivar Jacobson. The strategy was that the three leading design methodologists would work together, along with others who had responded to the initial draft of UM to create the final version of the Unified Method.

During this same period, it became apparent that a valid distinction could be drawn between the notation used to describe the design patterns of an OO application and the methodology, which specified the steps a developer should follow to create an OO application. Methodologies required a notation to record the results and analysis and design phases, but a notation did not necessarily imply an specific methodology. Hence, Rational's three UM architects (Booch, Rumbaugh, Jacobson) announced they would not be producing a methodology. Instead, they would standardize on a notation that could be used with any OO methodology. The new name for this shift in standardization strategy was the Unified Modeling Language (UML).

During this period, the Object Management Group (OMG) decided to create the Analysis and Design task force to adopt an OMG standard for OO notation. This group issued an RFP in 1996 and proposals, including Rational's just-completed UML 1.0, were submitted in January 1997. Most have assumed that the OMG would adopt the UML notation and in recent months the announcement that other submitters would be working with Rational on a revised proposal has all but assured it. The status of the OMG specification process and related proposals can be found at http://www.omg.org/library/schedule/AD_RFP1.htm.

One of the more interesting features of the UML notation is the degree that it accommodated the needs of Microsoft's OLE/ActiveX constructs. Grady Booch was an early visionary on the promise of component software.[1] Rational based their commercial tool strategy on a bet that OLE and ActiveX components would play a growing role in component development. The Rational submittal to the OMG provided all of the constructs and ActiveX developer could want to develop an applications, and Microsoft was one of the OMG members that joined Rational in cosponsoring the UML presentation.

[1] Dr. Grady has agreed to desist from using the term ComponentWare ® when he was informed that it was a registered trademark of I-Kinetics, Inc.

CASE TOOL REPOSITORIES AND STANDARDS

A second trend in the OO CASE market involves the development of a "universal" OO repository. Although it isn't yet a established as cost-effective, broadly adopted practice, enterprise information technology (IT) expects to develop libraries of business objects for reuse and hope to use OO CASE tools to manage and assemble components into new applications. Universal OO repositories are emerging that support companies that want to develop applications with several different OO tools while simultaneously maintaining libraries of components and business objects for reuse.

UML defines the information that tools will store about specifications of software systems. However, the defacto standardization of OO modeling does not address the problem of data exchange. Tools will still have their own format for representing the elements of UML in disk-resident data structures. To use a different modeling tool or to exchange model information between tools, the proprietary storage format of one tool has to be converted to that of the other. To avoid building N x (N-1) converters between pairs of tools, a standard exchange format such as CASE Data Interchange Format (CDIF) may be used, so that only 2xN converters are needed. However, a common interchange format still has disadvantages: it is hard to evolve; to view or process the information, it needs to be parsed into in-memory structures; and dependencies between models are hard to maintain since each model is usually contained in a separate file.

A standard repository model provides universal CASE tool interoperability by providing standardized storage for UML diagrams. It enables inter CASE tool data exchange through:

- an open information model that describes the objects and relationships used to store an object model diagram in the repository,
- mechanisms for extending the meta-data model;
- a set of published interfaces to manipulate the object diagram stored in the repository independent of the on-disk storage format;
- a shared database that spans many diagrams, allowing the expression of inter-diagram relationships.

This meta-data model is stored together with the information it describes which makes the data self-descriptive. A tool is therefore able to query the contents of a meta-data model and may use the results to adapt its processing. The interfaces of the repository completely encapsulate the stored information. This makes it possible to evolve data in response to information model changes, since tools depend only on the meta-data model, not on the stored representation of data. It also enables vendors to tailor interfaces to support extensions, since extensions do not affect the interface to non-extended objects.

In December 1995, new universal OO repositories that had been introduced by Unisys and IBM. Texas Instruments (TI) and Microsoft created a strategic partnership to create a repository based on OLE technology. Since then, Unisys and IBM have improved their offerings and OMG has set up a task force to standardize an object model for a universal repository. http://www.omg.org/library/schedule/AD_RF13.htm In February, Microsoft

announced the Microsoft Repository and at Software Development (April'97). Bill Gates proclaimed that the beginning of 'real" automated software development (CASE) was finally at hand.

The OMG Repository Specification

Jim Odell, chair of the A&D task force "71051.1733@compuserve.com"
Sridhar Iyengar, chair of the repository model task force
(Sridhar,Iyengar@mv.unisys.com).
"http://www.omg.org/library/schedule/CF_RF13.htm"

The OMG Common Facilities Platform Task Force issued an RFP for a repository model in 1996. After some consideration, the task force decided that it was really trying to develop a Meta Object Facility (MOF) -- an object information model that would specify the kinds of data that could be stored about OO development efforts. This recognition in 1996 led to two immediate complications. The MOF task force realized that any information model it standardized on would have to be compatible with the OMG Analysis and Design meta-model[2] and in the model that was used in the OMG Business Object Model.[3] The three different task forces, as well as the OMG technical committee, are now working to ensure that all three task forces endorse a common information meta-model.

In the case of the A&D task force, while there is wide support for the UML notational specification, there is a lot more energy being focused on arriving at a widely acceptable OO analysis and design (A&D) meta-data model. The same effort is taking place in the MOF and the Business Objects task forces. At the moment, OMG hopes to have a final specification for both the repository and the A&D models completed in July'97.

There are four submissions to the repository RFP. One was led by Unisys and supported by IBM, Oracle, Rational, Platinum, MCI Systemhouse, Digital, HP, EDS, and others. Data Access Corp., DSTC (Australia), and a joint proposal submitted three other proposals by Gemstone and Objectivity. At this point it looks like a version dominated by the Unisys et al. submittal will emerge from the MOF task force. Similarly, the Rational, et al. "UML" A&D meta-model will emerge from the A&D task force. Each will have been adjusted to correspond to the other. These two models, once adopted, will then provide constraints for the Business Object task forces' modeling efforts.

---

[2] "http://www.omg.org/library/schedule/AD_RFP1.htm"

[3] "http://www.omg.org/library/schedule/AD_RFP4.htm"

The Microsoft Repository

"http://www.microsoft.com/repository"
Phillip Bernstein, Repository Architect

The MS repository was jointly designed by TI and Microsoft, but was entirely implemented by Microsoft. The interfaces in the repository are based on COM and ActiveX interfaces. The Microsoft Repository consists of five components: repository databases, a repository engine, a type information model, several specific tool information models, and several generic tools for use with the repository (draw a figure). In addition, there are the application development tools that use the repository.

The repository engine will support any ODBC database. The repository engine stores and retrieves objects and relationships from the database. At the moment, Microsoft is demonstrating and shipping the repository with Microsoft SQL Server RDBMS.

The type information model is an abstract definition of all the types of information that the repository can handle. In effect, the repository can handle objects and components and relationships between them and all of the documentation needed to define a system.

A tool information model is an instantiation of the type information model designed to interface between one or more specific tools and the repository engine. For example, the Microsoft Development Object (MDO) Model is the tool information model that specifies the types of data of interest to Visual Basic programmers. Using the MDO in conjunction with a repository-aware VB product allows programmers to store VB information in the repository or to examine other VB data that has already been stored in the repository. Someone using the repository can select which of the various tool information models they want to include in their implementation.

Another information tool model is UML, which was developed by Rational, working in conjunction with Microsoft and several other vendors. The current version of Rational Rose for Visual Basic can use the UML model to store and retrieve UML date types in the repository.

A general demonstration of the repository and the type information model was held February 18, 1997 in Redmond, Washington. Five vendors that had been working with Microsoft demonstrated full cross-tool interoperability with each other's tools and with Microsoft's Visual Modeler. Each vendor's tool was able to put data into the repository and retrieving data entered by other tools. The vendors were Logic Works, Popkin Software, Rational Software, Select Software, and Texas Instruments.

As a result of this demonstration and a review of the repository specification, some 21 vendors, including all the major OO CASE vendors, agreed to support the Microsoft Repository effort. In effect, no matter what standard the OMG repository task force comes up with, there will be two repository standards -- the OMG's and Microsoft's and the CASE vendors will be committed to support both.

Version 1.0 of the repository is shipping in the Professional and Enterprise editions of the Visual Basic 5.0 programming system and with the Visual Studio 97 suite of development tools. It is also available with the Rational Rose Visual Basic 5 CASE product, but it is not yet available in the stand-alone version. TI's IEF Composer product, version 4.0, does not yet support the Microsoft Repository, but TI plans to add support when the Microsoft Repository has been enhanced.

The current version of the repository is, in fact, only an early version, and is more suited for storing components than for being used with CASE tools. For example, the current version does not have version control and thus couldn't be used by a team engaged in a serious CASE project. Version control and many other missing elements are scheduled to be added to later releases of the product, and TI expects that the next version of Composer, which is scheduled for release before the end of 1997, will be linked with the Microsoft Repository. The Microsoft Repository will be available in a general release form sometime in 1998.

Market Dynamics and Analysis of Standards-Based CASE Tools

There are a number of major technology trends that will increasingly impact and shape the OO CASE tool market for the next 3 years. The short list of the significant trends is:

- **Intranets:** Low cost of deployment and high benefit of ownership has resulted in the hyper-rapid deployment of enterprise Intranets. *Substantial return on investment (ROI) within months of deployment of Intranets have drastically altered Fortune 1000 IT strategies and application vendor product plans.* 95% of the Fortune 1000 have deployed at least one Intranet. 40% will have a budget of $1 Million or greater for Intranet-based projects for 1997.
- **De-facto Standardization of Middleware:** Middleware can be loosely described as the software to glue a new application with the existing installed base of software. The client/server era saw the birth of a vast range of different types of middleware. Standard low-cost middleware is the missing element needed to build Intranet-based distributed applications. The use of standard middleware eliminates the high costs and risk of being locked into a single platform or vendor solution. The Intranet platform and software vendors are rapidly converging on two middleware choices: OMG's CORBA and Microsoft's DCOM. The Common Object Request Broker Architecture ("CORBA") has quickly become pervasively available on every leading desktop and server platform. Implementations are available from over 10 vendors including HP, IBM, IONA, Netscape, Oracle, Sun and Visigenic. The other contender, Microsoft's DCOM, is very similar in capability to CORBA and has started shipping with every Windows platform in 1997. When asked in 1995, 14% of a group of Fortune 1000 IT managers surveyed indicated that they would deploy CORBA in 24 months. One year later, in 1996, this had grown to 40%. *Starting in 1997 and beyond it will become increasingly difficult to sell applications that are not CORBA or DCOM capable.*
- **De-facto Standardization of OO CASE Tool Notation:** UML is fast becoming the industry standard analysis and design notation for OO CASE tools. It provides

common notation, terminology, and semantics for object-oriented analysis and design. This standardization will greatly accelerate the acceptance of object-oriented approaches and technologies, because it protects an organization's previous object-oriented investments by limiting maintenance costs and training time and ensuring forward compatibility. *Starting in 1997 and beyond it will become increasingly difficult to sell OO CASE tools that are not UML capable.*

- **De-facto Standardization of OO CASE Tool Repository:** At the end of 1997 there will be two de-facto standards for OO CASE repositories: Microsoft Repository and OMG Repository. The industry further expects that these two repositories will be interoperable by directly supporting UML.
- **De-facto Standards Increase Return on Investment (ROI):** The enterprise buyer is attracted to "de-facto" standards, such as UML, IIOP, CORBA and Java that protect against being locked into a single platform or vendor solution. They are able to leverage an industry wide effort for third party products, rather than rely on one vendor for a complete infrastructure solution. This results in faster innovation with lower costs and better quality.
- **New Legacy Investment:** As enterprise information technology (IT) moves to standardized middleware (CORBA/DCOM) and the Internet, they must bring their installed base of production applications and data. Applications achieve "legacy" status if they have been in production for more than 24 hours and can not be pulled back because of user demand. At the expense of UNIX, Windows NT will increasingly gain market share of the workgroup database and Web server. UNIX will defend the high end server market with "enterprise server must have" features such as high performance 64-bit processing, clustering, and high availability. New Intranet systems must evolve by integrating with the existing mix of Windows NT and UNIX legacy systems. *The installed base of Windows NT and UNIX applications represent a legacy investment comparable to the mainframe investment.*

While re-packaging this existing investment in server-side applications, Enterprise Information Technology ("IT") has a strategic competitive need to change information systems as fast as their business changes. The promise of component-based applications is to enable developers to "snap together" applications by mixing and matching prefabricated software components. However, the lack of standard middleware, CASE tool notation and CASE tool repository has limited the potential of component software. UML-based CASE Tools using either the MS Repository or the OMG repository, available for less than $1,000 per seat, will change the market dramatically and catalyze component reuse in the enterprise.

Summary

At present, the resurrected CASE market seems set for a new period of growth and expansion. It is predicted that by Q4'97 there will be an adopted OMG specification for OO notation that will be based on UML. Additionally, there will be two dominant specifications for a standard CASE tool repository, the OMG repository and the MS repository and that they will interoperable.

Rational has positioned itself as the dominant OO CASE vendor. Enterprise technology gatekeepers are experimenting with this new generation of OO CASE tools, and the interest in developing software by reusing components is becoming more widespread. All vendors will also be able to take advantage of a standardized base OO notation (UML) and universal repositories. Those vendors that adopt these emerging standards will experience renewed growth. Those that do not support UML will face a rapidly diminishing market. Those that purchase UML incapable CASE tools will soon face owning unsupported and extinct products.

In the following sections, briefs are given of the leading OO CASE vendors that have publicly announced or have disclosed to I-Kinetics that their products would support UML by late 1997 or early 1998. Additionally, we have selected for analysis those vendors whom have predicted that their customers planned to transition from a proprietary notation (i.e. Booch or OMT) to UML by the end of 1998.

C. Component Software Tool Interoperability Roadmap

There exists a wealth of mature software manufacturing infrastructure and process. This investment is captured in design & analysis software Engineering tools (CASE); integrated development environments (IDE), project management methodologies, process and tools, code change management and quality assurance, test and complexity metrics tools. There is also an estimated US investment of $10 Trillion in code that captures valuable implementation of domain specific disciplines such as analytics, image processing, data management, real-time process control, etc. The two necessary ingredients are in-place for component-based assembly of software systems:design and development infrastructure and wealth of existing software.

However, there are barriers that prevent component software manufacturing. Both manufacturing tools and the "raw manufacturing material" are separated by integration and interoperability cost barriers caused by different:
languages,

- operating systems,
- network protocols,
- proprietary and standard application programming interfaces,
- application development tools.

Systems, although architectural modular, are implemented as valuable but inflexible components that can not be reused with new systems. The major defining behavior of a "legacy system" is that cost of ownership and putting achieved quality at risk inhibits evolution of functionality and capability.

The critical barriers to manufacturing component software can be eliminated with the adoption of standards. These standards need to evolve based on market driven commoditization. Dictated standards will not be adopted unless they are competitively viable and match the market dynamics of the software industry. Additionally, standardization on horizontal platform technologies such as operating systems, programming language, hardware, databases are not viable because of continuing competitive pressures. Instead we need to look for vertical technologies where standardization will create horizontal markets.

Standardization of tool metadata, such as OO CASE Tool or project management metadata, have the potential of creating a larger market as they would lower software manufacturing costs.

A key innovation in obtaining reusability is software reference architectures. Software architectures enable component reuse. A programmer cannot develop a component to meet reuse requirements unless those requirements are defined. A reference architecture, which models how components interact and gives common patterns of component interaction, provides such requirements. The reference architecture contains the information a programmer needs in order to write a component so that it can be reused. A second contribution is the notion that a software architecture is something that can be

captured in a formal language and reused across multiple applications. The formalization and reuse of software architectures provide major cost and schedule improvements.

The promise of component-based application manufacturing is to enable developers to "snap together" applications by mixing and matching prefabricated software components. However, the lack of OO CASE tool metadata model, metadata repository standard distributed computing middleware, has limited the realization of component software. There are a number of major technology trends that will increasingly impact and shape the OO CASE tool market for the next 3 years:

De-facto Standardization of OO CASE Tool Notation: UML is fast becoming the industry standard analysis and design notation for OO CASE tools. It provides common notation, terminology, and semantics for object-oriented analysis and design.

De-facto Standardization of OO Metadata Repository: At the end of 1997 there will be two de-facto standards for OO CASE repositories: Microsoft Repository and OMG Repository. The industry further expects that these two repositories will be interoperable by directly supporting UML.

De-facto Standardization of Middleware: Middleware can be loosely described as the software to glue a new application with the existing installed base of software. The Intranet platform and software vendors are rapidly converging on two middleware choices: OMG's CORBA and Microsoft's DCOM.

In 1996, high-end CASE tools average over $10,000 per seat. In 1997, emerging UML-based CASE Tools averaged $5,000 per seat. In 1998, CASE tools that can interoperate with each other based on UML and a standardized repository will drive per seat prices below $2,000. The dramatic drop in the entry level cost of CASE tools will change the market dramatically and catalyze component reuse in the enterprise.

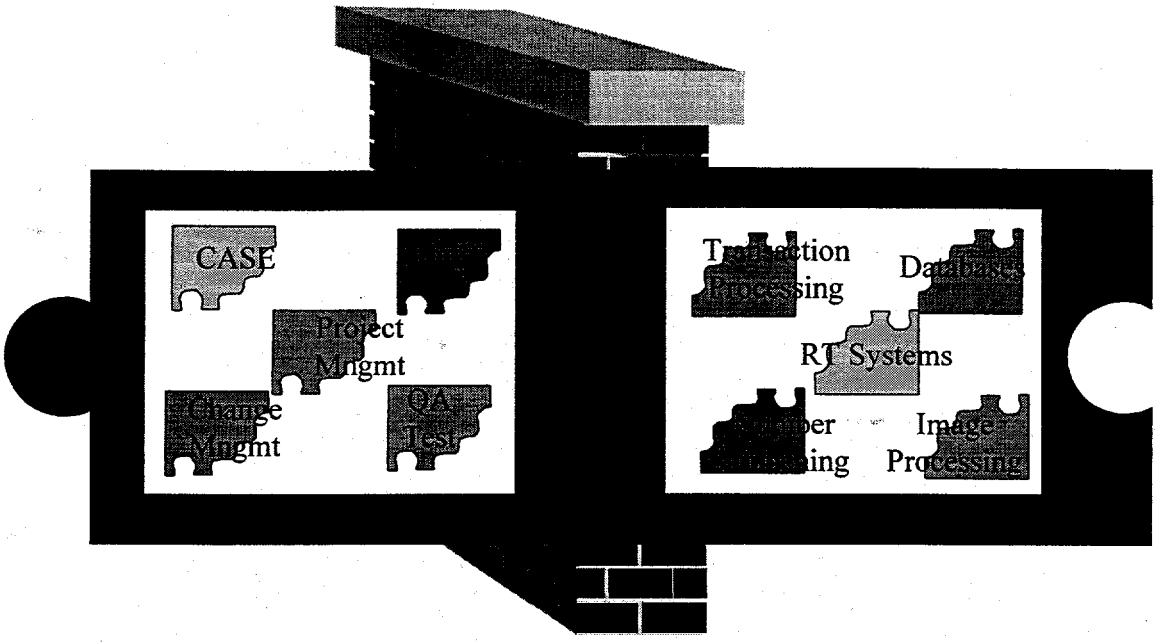


Figure 1. Tools and reference architecture cross over integrations barriers with interoperability standards such as UML and OMG Repository. We will explore UML and the emerging UML CASE tool repository standards to achieve tool interoperability.

C.1 What is UML?

The Unified Modeling Language (UML) is a language for modeling complex systems. UML started in 1995, as a Rational, Inc. market strategy to obtain dominant market share by consolidating Booch, Rambaugh's Object Modeling Technique (OMT) and Jacobson's USE CASE OOA/OOD methodologies. Rational, led by Grady Booch, hired Jim Rambaugh and Ivar Jacobson to work together to design UML.

In 1996, Rational teamed with HP, TI, I-logix, and IBM to propose UML as a standard for OO methodology to the OMG. This catalyzed the design of UML to include other leading object modeling theorists, such as Peter Coad, Jim Odell and David Harel (Statecharts). UML goes beyond integrating the "best-of-breed" of the various OO models, it also introduces new concepts learned from the previous generation. The result is that UML is a third-generation OO modeling language.

UML is a formal approach to completely modeling a complex systems behavior. If completely modeled, then a system is considered to be completely specified. UML primary approach is to define the model elements, and then secondarily how the elements are graphically defined. A complete UML-based model can be represented in either a graphical representation (diagrams) or in a text representation (statements).

The modeling capabilities of UML are:

- **Object Model**:The backbone of the UML is the object model. The major elements are classes, instances, associations.

- **Use Cases and Scenarios:** A scenarios is a specific path of interactions in the system. Groups of related scenarios are use cases. Scenarios are specified by object collaboration or message sequence charts. For example, "Access Database" is a use case. The actual process of getting a specific set of data from the database is a scenario. Each use case typically represents dozens of scenarios.
- **Behavioral Modeling and Statecharts:** UML incorporates the latest work in statecharts to including decomposition of states into states (nested states), orthogonal states, entry and exit actions on states and transition actions. Statecharts are particularly important for having mathematically closed models of real-time systems.
- **Packaging** enables grouping of UML elements into subsystems.
- **Tasking** represents task-specific message synchronization
- **Physical Topology** enables the modeling of relative location in a distributed system. It is particularly useful in real-time embedded systems because you can annotate a diagram to locate packages and objects.
- **Source Code Organization:** ability to model source code compile dependency of the modules and sub-systems (packages) of the entire system
- These areas define a stable base that unifies OO modeling. Additionally, experimental notation can be added, allowing UML to evolve, while still maintaining a common standard.

UML is fast becoming the universal standard analysis and design notation for OO CASE tools. Under a SBIR, ARDEC investigated over 20 OO CASE tools. Over 60% have UML support in Q3'97. Based on vendor announcements, over 90%+ of OO CASE tool vendors will be shipping UML capable tools by Q3'98.

The OMG Analysis and Design task force issued an RFP in 1996 and proposals, including Rational's just-completed UML 1.0, were submitted in January 1997. Most have assumed that the OMG would adopt the UML notation and since Q1'97 the joining of other submissions into the Rational-lead proposal has all but assured acceptance by Q4'97. The status of the OMG specification process and related proposals can be found at http://www.omg.org/library/schedule/AD_RFP1.htm.

Starting in 1998 and beyond it will become increasingly difficult to find OO CASE tools that are not UML capable.
This standardization will greatly accelerate the acceptance of object-oriented approaches and technologies, because it protects an organization's previous object-oriented investments by limiting maintenance costs and training time and ensuring forward compatibility

C.2 OO Case Tool Interoperability Problem

UML defines the information that tools will store about specifications of software systems. However, the de-facto standardization of OO modeling does not address the problem of data exchange. Tools will still have their own format for representing the elements of UML in disk-resident data structures. To use a different modeling tool or to exchange model information between tools, the proprietary storage format of one tool has

to be converted to that of the other. If there are N tools in the market, then each tool has to develop N-1 converters to interoperate with the other tools. Additionally, each tool must support 2*N versions, because each tool will have at least a previous version, and the current version. As an industry, N x (N-1) converters need to be developed to achieve universal interoperability.

This is too expensive and thus the CASE tool market has never achieved interoperability.

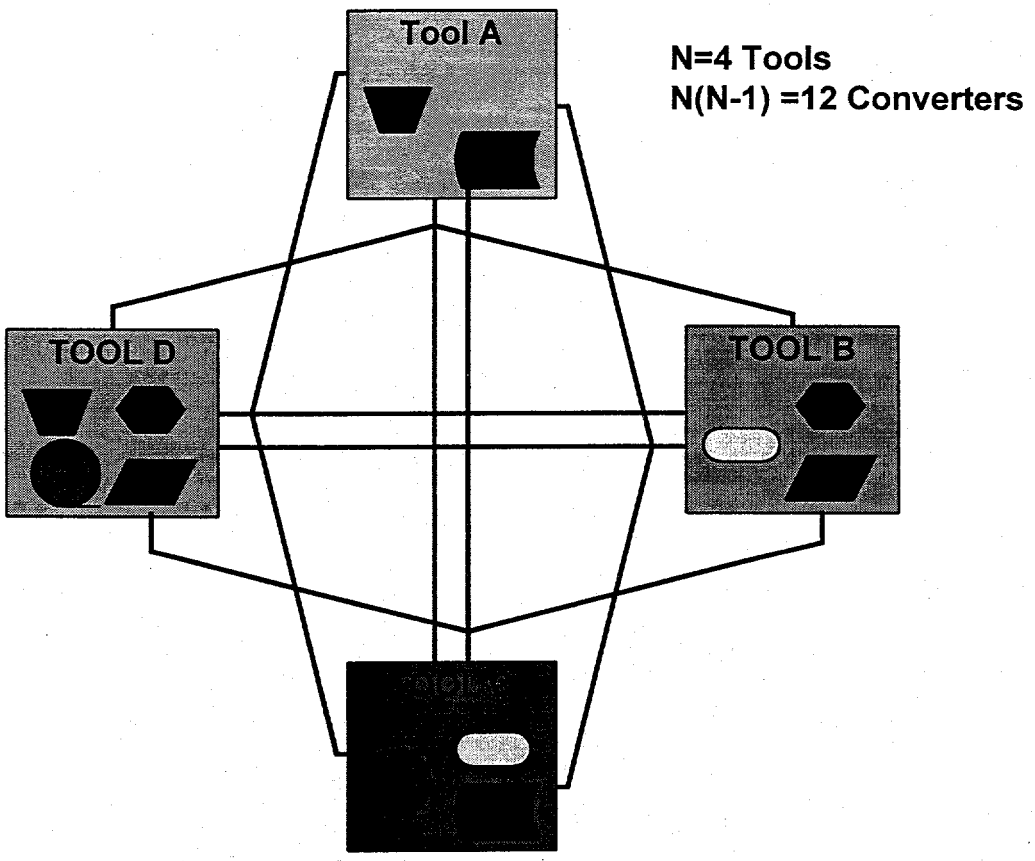


Figure 2. Tool interoperability requires Nx (N-1) converters.

C.3 Interoperability Solution: Standard notation and Universal Repository

A universal repository model provides universal CASE tool interoperability by providing standardized storage for UML diagrams. It enables inter CASE tool data exchange through:

- an open information model that describes the objects and relationships used to store an object model diagram in the repository,
- mechanisms for extending the meta-data model;
- a set of published interfaces to manipulate the object diagram stored in the repository independent of the on-disk storage format;

- a shared database that spans many diagrams, allowing the expression of inter-diagram relationships.

The repository interface completely encapsulates the stored information. This makes it possible to evolve data in response to information model changes, since tools depend only on the meta-data model, not on the stored representation of data.
This meta-data model is stored together with the information it describes which makes the data self-descriptive. A tool is therefore able to query the contents of a meta-data model and may use the results to adapt its processing. It also enables vendors to tailor interfaces to support extensions, since extensions do not affect the standard UML metadata.

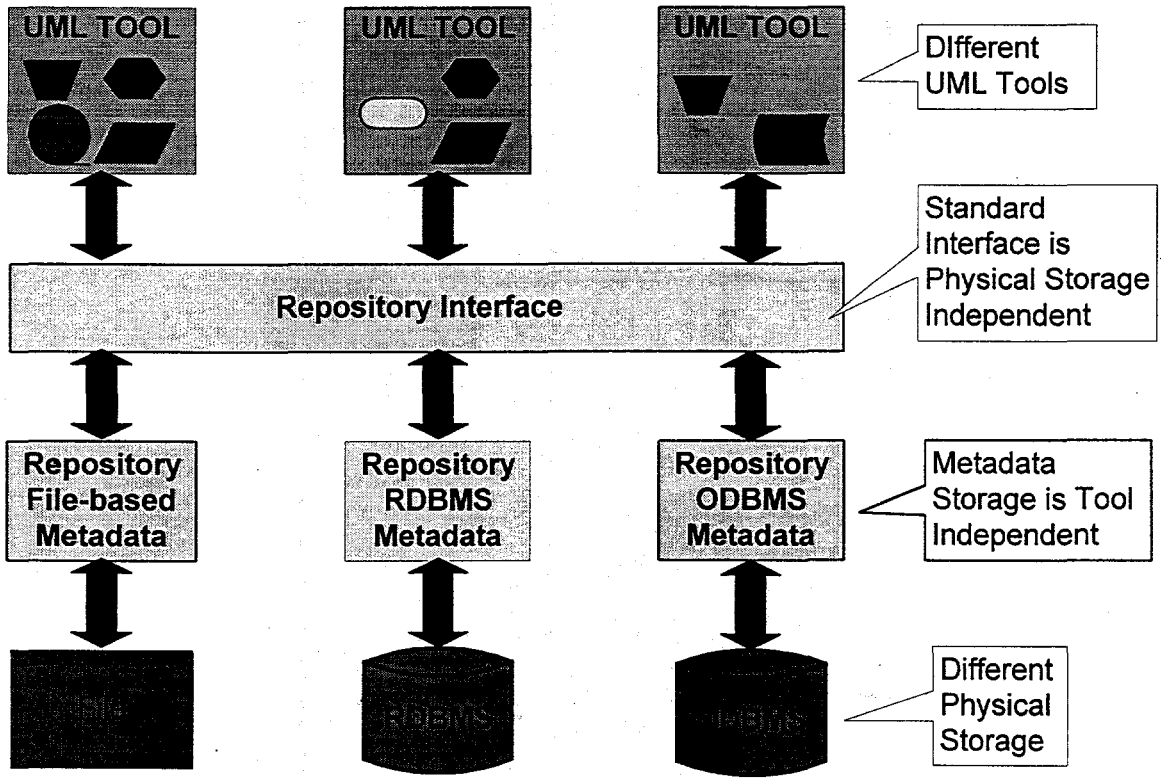


Figure 3. Architecture of an Universal Repository

CASE tool interoperability eliminates the remaining technical infrastructure barriers for use and reuse of reference architecture models of our complex systems. In 1998, we will achieve pervasive CASE tool interoperability by wide market support of UML and either the MS Repository or the OMG repository.

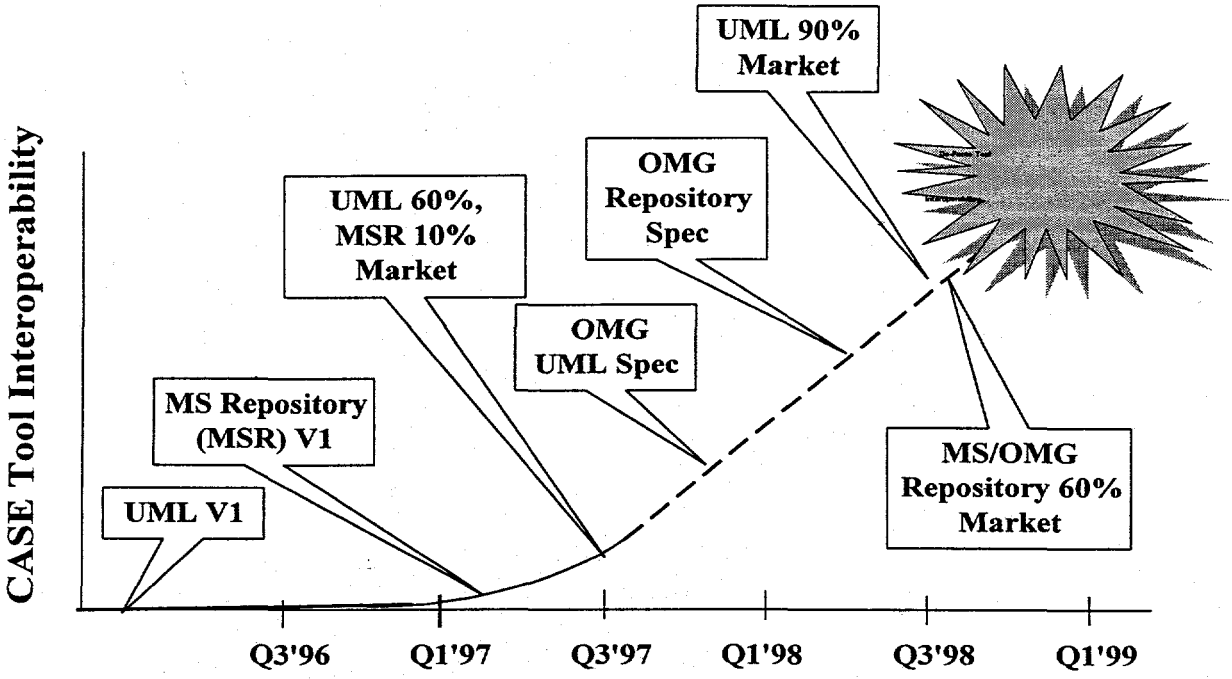


Figure 4. By Q3'98, 90% of all OO CASE tools will support UML and 60% will support either the MS or the OMG repository, achieving practical universal tool interoperability.

D. Case Study: Migration of Army ARDEC Investment in Reference Architecture Designs to UML

I-Kinetics researched the ability of the emerging standard for object-oriented analysis and design (OOA/OOD), the Unified Modeling Language (UML) to specify real-time systems.

ARDEC Automation and Robotics has an existing investment of reference architectures specified in ObjecTime's OOA/OOD notation, Real-Time Object-Oriendted Modeling (ROOM). One of these reference architectures, the ArmyFiringMission was chosen to test the ability to translate ROOM into UML.

I-Kinetics and ARDEC RAI worked together in investigating UML capable tools and repositories. The top priority was to demonstrate CASE tool interoperability for Reference Architecture Initiative The goal was to have two or more UML CASE tools capable of exchanging UML Reference Architecture Design (RAD) through the MS Repository. It was realized that this was of high risk as there is only one UML capable tool that has been commercially released at the time (Rational Rose).

The key risk point was the availability of translators from ROOM to UML. If none were available, the fall back position was to accomplish manual translation. It was determined that ObjectTime translators were not available, but would be available (according to ObjecTime product management) sometime during the period between Q4'97 and Q1'98.

Manual translation of parts of the ArmyFiringMission were accomplished. The following procedure was determining from analyzing ROOM and UML and designing a procedure for mapping notation from ROOM to UML.

1. Mapping of Keywords

ROOM	UML
actor	object
actor class	class
composite actor class	package
data class	class
protocol class (set of related messages)	class
protocol (specification of pattern defining a protocol class, e.g., direction and sequence of messages)	message sequences
port (reference to protocol class)	interface
replication (of actors or ports)	multiplicity
reference	use of a class name as an object type in

	another class
reference name	object name

2. Mapping of Diagrams

ROOM	UML
actor class diagram	class diagram, collaboration diagram
composite actor class diagram	package diagram
ROOMchart	state diagram
internal message sequence diagram	sequence diagram

3. Algorithm for Mapping ROOM Actor Class Diagrams to UML Class Diagrams

- Represent the whole system as a top-level UML package in a UML class diagram.

- Below the top-level package, make other UML class diagrams. In each of these class diagrams, represent a ROOM shaded "box" from a (composite) actor class diagram as a UML package if it has a lower-level diagram that shows other shaded boxes. Otherwise (i.e., its lower-level actor class diagram is just an un-shaded rectangle with ports on the border), represent it as a UML class. For both packages and classes, use names starting with upper-case letters.

- In UML class diagrams that contain UML classes, show their operations (methods) and their associations. To do this, look at the shaded box that mapped to a UML class. If the shaded box has a port represented as a *white* square with a *black* circle inside, then represent the name next to the port as a class operation (method). If the port name is a verb, then use the same name for the operation. If the port name is a noun, then use the same name preceded by "get." (We model the ports represented as *white* squares with *black* circles inside as returning a method call to the class that invoked the method. We model the ports represented as *black* squares with *white* circles inside as the invocation of the method. Thus, the class that returns the method call is the one that contains the method.) For every line connecting two boxes in the ROOM diagram, put an association line in the UML diagram. If a port is replicated, use the multiplicity symbol for "0 or more" in the UML diagram.

- In class diagrams containing more than one package, the dependency between the packages should be shown (there cannot be an association between a package and a class). The direction of the dependency arrow should match the port representation as described above, i.e., the arrow goes *from* the package containing the method that does the invoking *to* the package containing the invoked method.

4. Algorithm for Mapping ROOM Actor Class Diagrams to UML Collaboration Diagrams

- For every UML class diagram described above that shows associations, make a corresponding UML collaboration diagram in which every box in the diagram is a UML class (i.e., all packages have been reduced to their lowest-level classes). The classes are represented as objects (starting with lower-case letters), and the associations are represented as messages. The direction of the message arrows should be *from* the object invoking the message *to* the object returning the method call, with the name of the message near the arrow. (In Rational Rose, it is easier to make a sequence diagram first and then generate

the corresponding collaboration diagram. The order of the messages must be specified, so make the best guess if that information is missing.)

- Remember to include collaboration diagrams for dependencies between packages shown in a high-level ROOM composite actor class diagram, if that information is available (see, for example, the collaboration diagram called "fireStrat-autoloadCont".

5. Algorithm for Mapping ROOMcharts to UML State Diagrams

   - Basically, this is a one-to-one mapping showing states and state transitions.

   - The *initialize* state is mapped to the *start* state in UML. There is no ending state in the ROOMcharts, so there will be no *end* state in the UML diagrams unless more information is given.

   - Nested states in UML can be used to model hierarchical states in ROOMcharts, i.e., states in which lower-level state diagrams can reside. (In Rational Rose, just place lower-level states inside a high-level state, and the high-level-state box will expand to accommodate them.)

Resources

Further reading on standardization of OO CASE notation and repository efforts.

ROOM

- Real-Time Object-Oriented Modeling by Bran Selic, Garth Gullekson, and Paul Ward

UML References

- UML Distilled by Martin Fowler with Kendall Scott

- General Reference Index *http://www.rational.com/uml/index.html*

- UML Document Set *http://www.rational.com/uml/start/* This series of documents describes the Unified Modeling Language (UML), a language for specifying, visualizing, and constructing the artifacts of software systems, as well as for business modeling. The UML represents a collection of "best engineering practices" that have proven successful in the modeling of large and complex systems.

- OMG Analysis and Design Task Force
  http://www.omg.org/library/schedule/AD_RFP1.htm

UML Repository Initiatives

- Microsoft Repository http://www.microsoft.com/repository

- OMG Common Facilities (Repositories)
  http://www.omg.org/library/schedule/CF_RFI3.htm

E. Case Study: Transformation of Pratt & Whitney Turbine Design Simulation Codes into Distributed Software Components

CWC/TRP/OORAD Success Story
The Pratt & Whitney Virtual Jet Engine
Parallel, Distributed Optimization Toolkit

The Virtual Jet Engine application utilizes CORBA technology to enable multi-dimensional, aerothermal simulation and optimization of jet aircraft engines on a network of computers. Millions of lines of trusted analytical software have been encapsulated in a reusable object-oriented framework of jet aircraft components. The Virtual Jet Engine architecture features run-time configurable analysis and optimization components.

Significant reductions in time to manufacture are bolstered by the systems use of parallel, distributed processing. Performance benchmarks are impressive: 9000 executions of a 3D single stage high pressure turbine ran in 52 hours across 40 workstations, resulting in a 40X performance increase over the sequential analysis. All Pratt & Whitney commercial and military engine programs will directly benefit from this technology and provide Pratt & Whitney a significant advantage over its competitors.

Basic OO Functionality:

The systems architecture is based on a distributed object-oriented framework of jet engine domain objects or components that encapsulate legacy analytical engineering analysis applications written in FORTRAN. Figure 1 depicts a high level view of the Virtual Jet Engine.

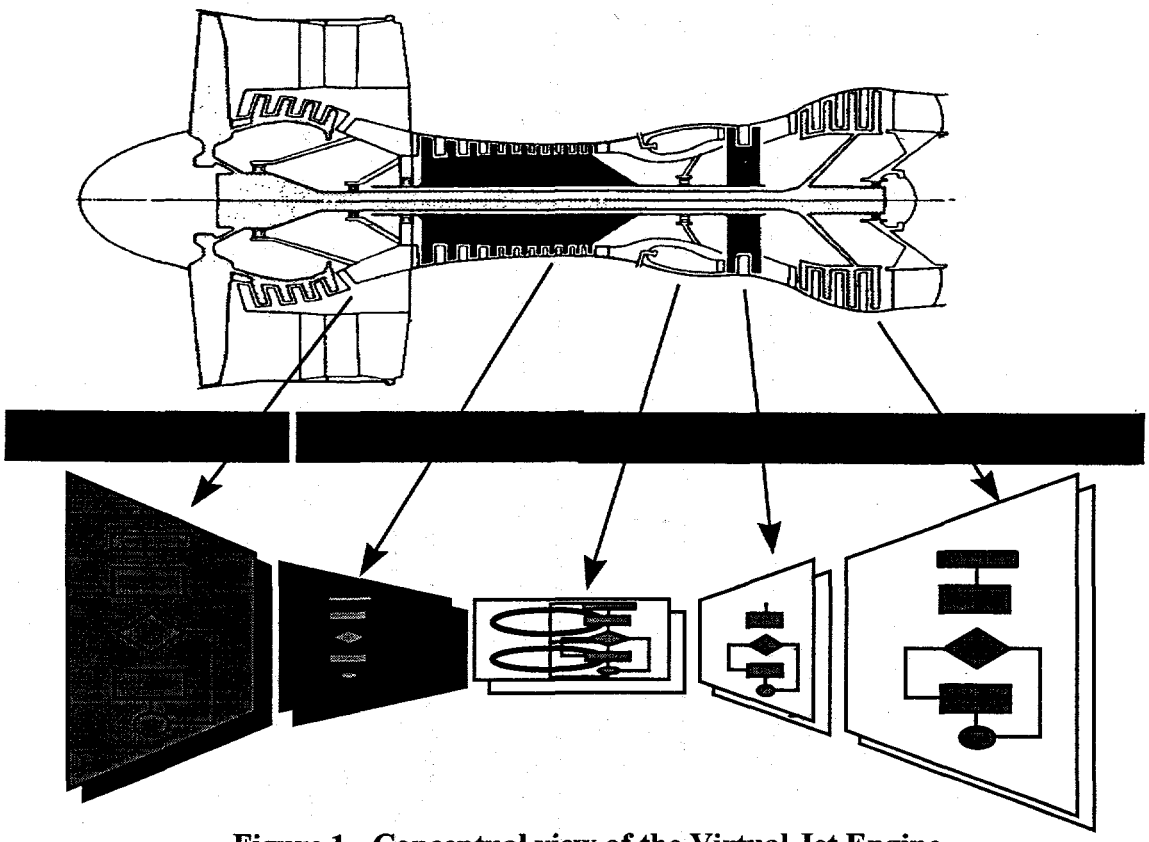


Figure 1 - Conceptual view of the Virtual Jet Engine.

The Parallel, Distributed Optimization Toolkit is a C++/CORBA-based distributed, parallel engineering analyses/optimization framework. Featuring run-time configurable, plug-n-play analysis and optimization components. Figure 2 depicts an orbital view of the toolkit. Parallelism was achieved by implementing a threaded consumer-producer algorithm in an implementation of the Evaluator object. The domain analysis is an object that is sub-classed off of the Analysis class. The domain analysis object provides a map method that translates generic DesignVariables into domain specific entities.

Several Optimizer implementations have been developed including integration with Vanderplats ADS gradient search techniques and multiple derivatives of the canonical genetic algorithm. Plug and play behavior is achieved by providing alternate class implementations that are specified by the client at runtime.

Development Environment:

The system was developed entirely on SUN Solaris 2.x with the SPARCworks development environment. The CORBA development environment was IONA Orbix 1.3 MT. The Graphical User Interface was developed with the assistance of X-Designer from Imperial SoftwareTechnology and a proprietary X11/Motif user interface C++ class framework. The C++ Standard Template Library and string class from ObjectSpace Inc. were heavily utilized and come highly recommended.

Project Life Cycle:

Pratt & Whitney has been utilizing object-oriented technologies since the late 80's. This particular project used the Spiral Life Cycle Model with a heavy emphasis on prototyping, risk management and incremental development. A development team with sound software engineering skills enabled this systems success. External Airfoil as a complete design tool has over 3 man years of applied software development over a period of 4 years.

Benefits:

The architecture of the system is clean, concise and models both the physical and logical domain. Object technology has brought ability to abstract the extremely complex problem domain of Jet Engine Design and Simulation into manageable components. CORBA brought a distributed object model with the ability to easily encapsulate legacy systems.

Conclusion:

The resulting Airfoil design simulation has been deployed at Pratt&Whitney Florida and Connecticut sites. Pratt&Whitney is currently negotiating a 3000 node run-time license with IONA so as to achieve wide-availability of this and other planning CORBA-based systems. The next step in the Virtual Engine system is to expand the system to include more design simulation codes for other parts of the turbine design. The ultimate goal is to be able to completely simulate the entire turbine engine.

Above all, this project has impressed the importance of sound software engineering practices and object-oriented architecture. The use of CORBA has enabled the development team to focus on the domain problem, verses the development of a distributed computing infrastructure.

F. Case Study: Migration of Air Force Satellite Image Analysis to Distributed Software Components

Under a Air Force SBIR Phase I the Air Force has tasked I-Kinetics with migrating a image analysis toolset to a set of components. The ability to access these components from CORBA, OLE or WWW client environments are key success criteria. This project will apply the products of task 3.1, 3.3, 3.4 and 3.8.

I-Kinetics has completed the Phase I deliverables and milestones. Image processing (IP) applications require the use of graphical and mathematical libraries. The different possible way an image can be analyzed often surpasses the functionality provided by any one commercial product. Image processing applications can be very difficult and costly to maintain because of the number of different packages that need to be integrated.

The solution to this problem is to create application components that have the following benefits:

- Components are easily *customizable* for new applications through reassembly.
- Components are easily *manageable* due to their inherent canonical characteristics.
- Components export *legacy* application functionality via a *standard interface* thereby hiding the complexities of the application.
- Components are conducive to *plug-and-play* in heterogeneous Internet and Intranet environments.

In the case of Image Processing applications, the ComponentFactory transformed a commercial library, Matlab® and a specialized satellite image processing toolkit, IPToolkit, into components. Each library was transformed onto a component on its native platform. The ComponentFactory distributed object technology enables you to put together highly specialized image analysis by assembling components. You can build distributed applications by integrating different image processing components in their native habitat, while accessing each from an environment where it does not currently reside. Additionally, using "plug&play" components, such as the DataBroker™, you can quickly add data management to your image analysis application using popular databases such as Oracle, Sybase or Informix.

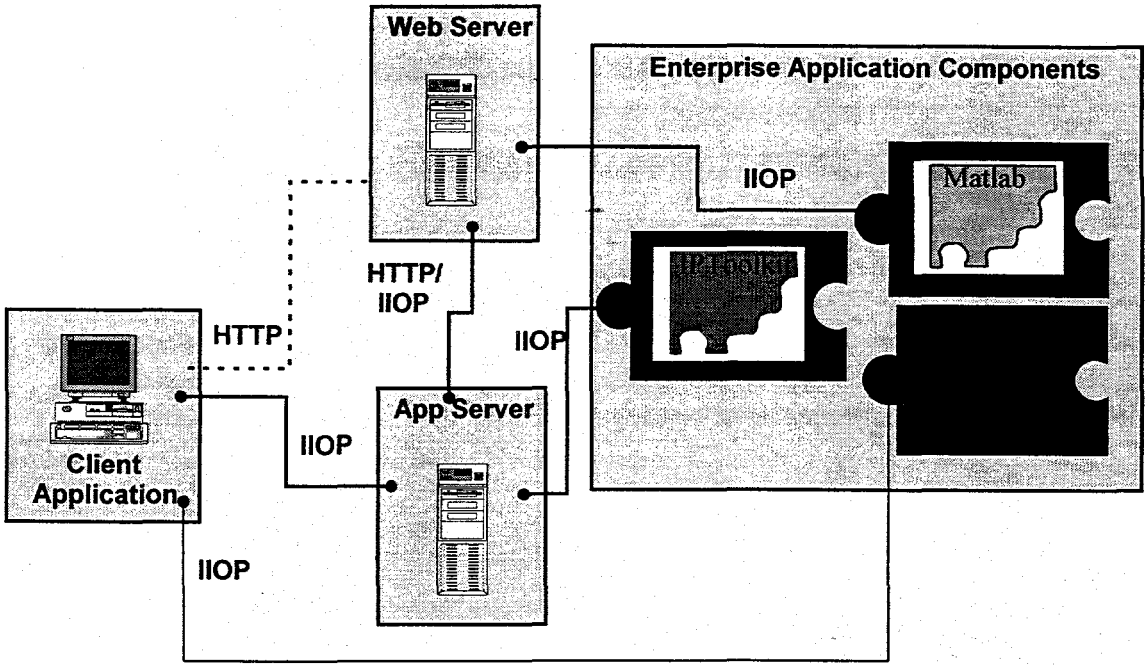


Image processing components result in an unlimited arrangement of image analysis application compositions that are determined by the client application demand. Client applications can access enterprise applications directly using IIOP or indirectly though a Web Server or application Server using IIOP or HTTP/HTML.

G. Case Study: Migration of Air Force Mission Operations to Distributed Software Components

Air Force, Phillips Lab, Satellite Telemetry Command & Control Mission Operations

Under a Air Force SBIR Phase I, the Air Force tasked I-Kinetics with adding a Component Warehouse to their Satellite Telemetry, Command and Control (VISTA) testbed. Continuous availability is a key requirement for this mission-critical system. I-Kinetics will transform key parts of the STCC into replicated components. Component replication will enable critical STCC operations and services to continue when any one component fails. This project applied the products of task 3.1, 3.3, 3.4, 3.5 and 3.8.

Component-First Approach

Let's briefly review the architectural goals we've established:

- To maximally leverage legacy assets (that is, legacy domain logic).
- To minimize the exposure of any legacy architectural constraints into our new system. This is where our use of encapsulation diverges from the Legacy-First approach, which naturally exposes legacy partitioning into the new environment. *The partitioning of the distributed component architecture must be driven by a well-defined object model that is based on the enterprise domain, not the existing legacy application partitioning.*

This second goal, eliminating legacy architectural constraints, is important for a number of reasons. It means we'll invest additional project cost and time to encapsulate the legacy application in such a way as to not let it pollute our environment in the future. In practical terms, this fundamentally changes the way we approach encapsulation. We no longer wrap applications in isolation.

Rather than exporting legacy application interfaces and environments into a new domain, the Component-First approach mandates constructing our new domain first and then determining the minimal set of legacy application functions we need to populate it. We may want only a small fraction of what some large legacy application can do. In fact, in general we won't export any interfaces of legacy applications -- they will all be hidden behind a new object model.

The Component-First encapsulation strategy enables us to re-engineer functionality piece by piece, at a pace driven by the new business needs, and to control the transition. This control enables us to do incrementally reengineering, which significantly reduces the risk involved in undertaking this sort of infrastructure surgery. Application component specifications driven by the new organizational needs, rather than the legacy needs that are compiled into the legacy interface specification, is the founding principal of the Component-First encapsulation strategy.

The Basic Steps of the Component-First Encapsulation Process

The Component-First process outlines three basic steps that encourage usability and reuse of the legacy application capabilities while suppressing exposure to limitations of the legacy system architecture.

4. Perform domain analysis and generate the object model.
5. Identify public interfaces of the object model.
6. Encapsulate legacy applications to populate specific functionality in the object model.

By generating the object model from analysis rather than a collection of legacy applications, we reduce the likelihood that we'll inadvertently cripple our new system by incorporating legacy architecture constraints.

Legacy Transition Case Study: Telemetry Management System

To illustrate the basic steps in the Component-First process, let's look at a case study drawn from a recent project. Consider the telemetry management and analysis system example shown in Figure 1. The Telemetry Manager processes and distributes data from a high-speed telemetry data stream. In its native environment, the Telemetry Manager communicates with a GUI-based Display & Control process and a Telemetry Analysis process using file-based interprocess communication (IPC). The file-based IPC between the processes is implemented with two shared files, one for outbound and one for inbound messages. Each of the three processes finds these files with pathnames embedded in their software code. The Telemetry Manager also performs process control, creating and destroying the client Display & Control and Telemetry Analysis processes.

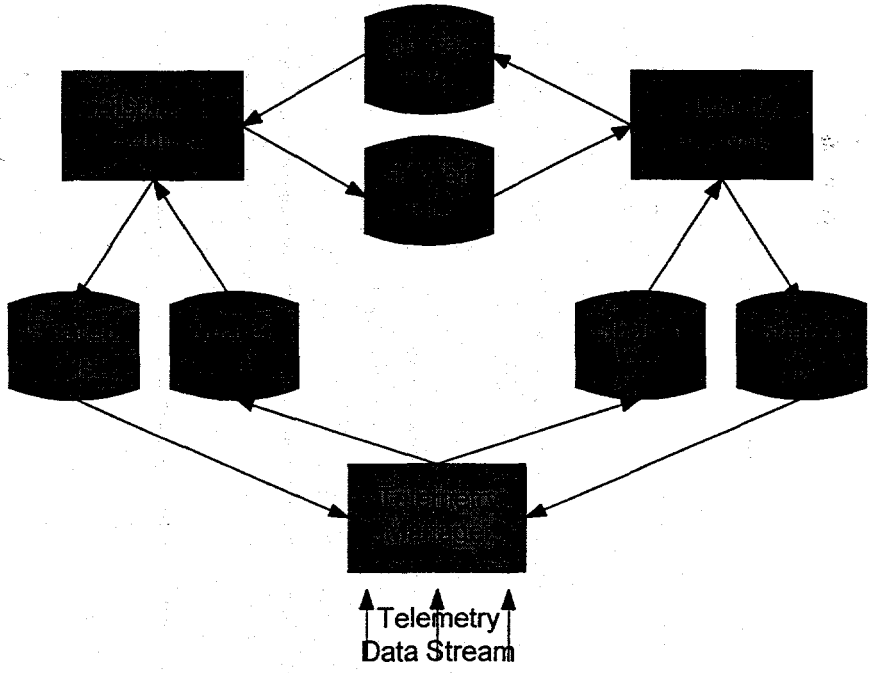


Figure 1. Legacy system for distribution management, analysis and display of satellite telemetry data.

The valuable domain logic part of the telemetry manager is the telemetry data "decommutation", where bit streams are unpacked and data values for dozens of separate fields are extracted. The display & control client "controls" the telemetry manager and the telemetry analysis processes by sending commands through the file IPC mechanism which register the display & control client's interest in specific fields. Either the telemetry manager process or telemetry analysis process sends via the file-based IPC the indicated field values to the display/control process.

Several unpleasant constraints are apparent.

- The system is strictly single-threaded and performance can not be scaled up by upgrading to a multi-processor platform or distributing across multiple hosts. Because of the manner in which the files are specified, only one set of processes can run per machine.
- New processes can not be added. Only one pair of clients can connect to the Telemetry Manager. The Telemetry Manager cannot maintain different sets of fields for different clients.

We were tasked with transitioning this mission-critical system to a CORBA-based infrastructure. The client initially wanted to "wrap" the system by using a Legacy-First approach. The entire system was to be encapsulated, creating a single monolithic component, by exporting the system's command interface with the CORBA Interface Definition Language (IDL). However, the resulting encapsulation wrapper would have propagated the limitations of the legacy system to the distributed object architecture. Only one transaction management component could be created on a single machine. Only one client application could connect to it. Not only would the new architecture offer no relief from these problems, but it would also severely limit adding new applications that used the legacy telemetry management system.

However, the sponsor needed the next generation of the telemetry management system to have scalability, fault-tolerance based on redundancy, and integration with other legacy systems, such as high-performance analytics hosted at remote installations. The sponsor agreed to use the Component-First approach, trading up for the critical capabilities they needed.

Following the three step process of the Component-First strategy, we did the following:

4. Designed an object model from a telemetry management domain analysis.
5. Used the object model to specify telemetry management component interfaces with CORBA IDL.
6. Wrapped the telemetry management system to populate the specific functionality of each component.

Figure 2 shows the first generation of the object model resulting from a domain analysis is shown. The Field object represents the telemetry data fields decoded from the raw Telemetry Frame data by the telemetry manager. New system component interact with Field objects and not with the encapsulated application. This is a different partitioning of the problem, one, which would be unlikely if the design had been based on the legacy telemetry management system architecture.

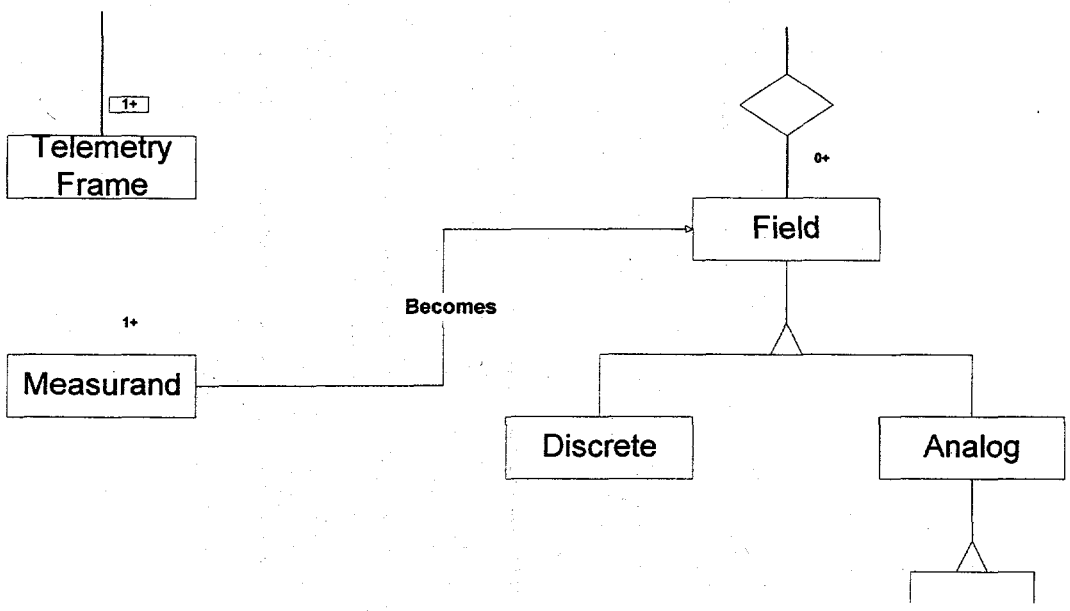


Figure 2. A portion of the telemetry management system object model.

Listing 1 shows the IDL specification for some of the component interfaces of the telemetry management system. Exception handling has been omitted in favor of brevity. The legacy telemetry management system is encapsulated with the DataSource interface. New applications and client components interact through the FieldManager interface, which in turn is the only component to interact with the DataSource interface. These well-defined telemetry management component interfaces completely hide the existence of the legacy telemetry management system.

```
module Telemetry {
// client subscription list for fields
typedef sequence<string> FieldNameList;

// base interface for all telemetry fields
interface Field
{     readonly attribute string name;
      readonly attribute long dev_id;
      attribute long time;
      attribute any value;
      attribute short color;
};

// Manager for all Fields; interacts with DataSource interface
interface FieldManager{
      boolean setup(in long dev_id, in string pass, in Output outRef);
      FieldNameList subscribe(in FieldNameList list);
      boolean advise(in string fname, in double displayValue,
            in long seconds, in short colorValue);

      boolean updateSpecial(in string name, in string value);

      oneway void startReplay();
      boolean stopReplay();
};

// legacy system wrapper interface
interface DataSource                {
      boolean setup(in long dev_id, in string pass,
                  in FieldManager dsRef);
      oneway void replayBegin();
      boolean replayEnd();
      void subscribe(in FieldNameList slist);
};
};          // end module Telemetry
```

Listing 1 CORBA IDL specifications for Telemetry Management Components.

Figure 3 depicts the new distributed object architecture. The Field Manager component is multi-threaded, can be replicated, and be hosted on different machines. The Field Manager accesses the legacy telemetry management system via the DataSource component. The DataSource component encapsulates the legacy Telemetry Manager and Telemetry Analysis processes.

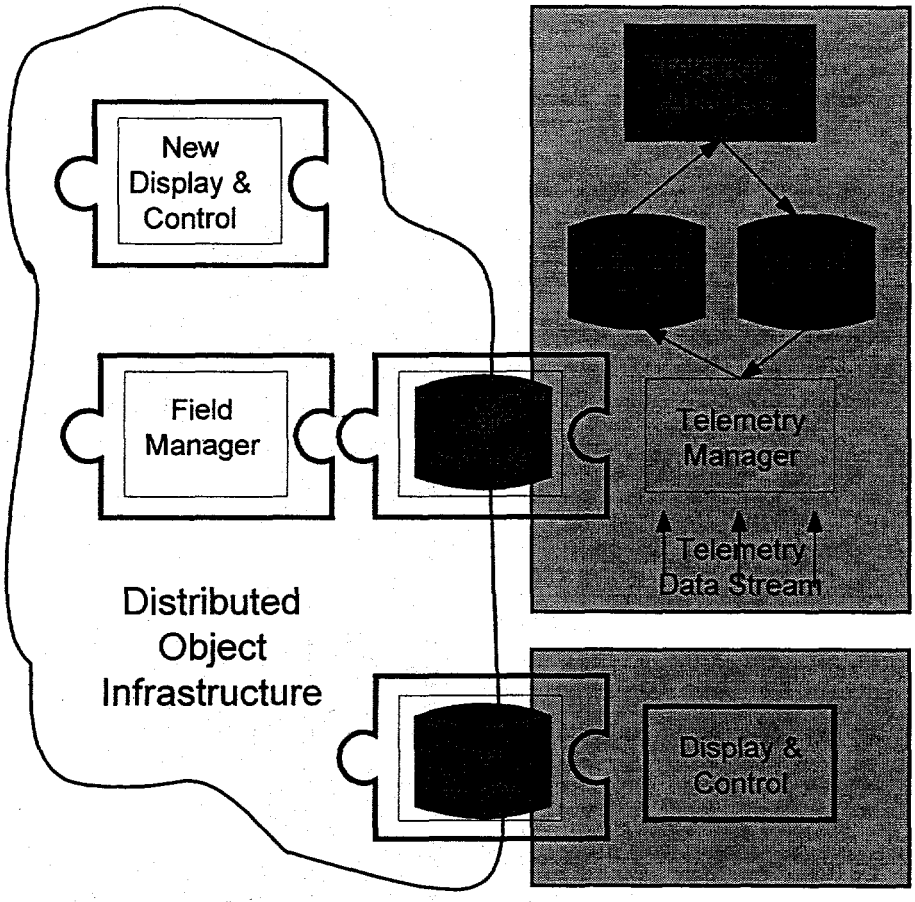


Figure 3. The Telemetry Management System is migrated to a distributed object architecture using the DataSource and GUI Proxy component encapsulation wrappers.

The DataSource component itself is subject to certain constraints, such as running on the same machine as the legacy telemetry management system. However, because the Field Manager is a logically centralized source of all telemetry Field objects, the full benefits of a CORBA-based telemetry management system can be realized. We are freed from the constraints of the legacy system, as they are isolated and contained by the DataSource component encapsulation wrapper.

The Field Manager can receive (and discard) redundant copies of uniquely identified Field objects from two or more DataSource components. We can achieve fault-tolerance by replicating the legacy telemetry system with a "hot-swap" redundant spare. There can

be multiple Display & Control components, enabling access to the Telemetry Management System from multiple sites. The system can scale to meet the demands of a growing population of users by copying and broadcasting the Field objects and running multiple Field Managers on multiple machines. Finally, we can answer to the large and constant need to add new telemetry analysis processing. Using the same DataSource encapsulation pattern, specialized "hothouse" analytics can be incrementally added that consume the raw telemetry Field objects and output new derived data Field objects back into the system.

The component wrappers protect new applications from changes in the legacy system. We can proceed with reengineering the legacy implementations with no effect on client applications. A client requesting notification upon change in state of one of the Field objects has no way of knowing (nor does it care) whether the notification is originally generated by the legacy system or some new mechanism. If we had chosen the Legacy-First approach and created a wrapper that mirrored the legacy telemetry management system command interface, we would have been able to achieve the full benefit of the distributed object infrastructure. Clients using the legacy server would be manipulating it directly through interfaces customized for the legacy application; any reengineering of the original application would be far more likely to impact those interfaces (and thus any new application using them).

H. Bibliography

[Bh87] Edited by B. Bhargava, *Concurrency Control and Reliability in Distributed Systems*. Van Nostrand Reinhold, Pub., New York, N.Y. 1987.

[COOL97] CHORUS/COOL R4 Product Description, Chorus Systems report, February 1997.

[Coulson95] G. Coulson, G. S. Blair, F. Horn, L. Hazard, and J. B.Stefani: "Supporting the Real-Time Requirements of Continuous Media in Open Distributed Processing", Computer Networks and ISDN Systems, Special Issue on Open Distributed Processing, Vol.27 No 8, July 1995.

[CWC94] B.H. Cottman, *ComponentWare Consortium Technology Plan.* White Paper, ComponentWare Consortium, Inc, http://www.componentware.com/techrmwp.htm, December 1994

[CWC95] B.H. Cottman, *Software Component Factory*. White Paper, ComponentWare Consortium, Inc, http://www.componentware.com/SCF_wp.htm, June 1995.

[CWC95b] Mike Higgs and Amitava Maulik, *ComponentWare Architecture*. White Paper, ComponentWare Consortium, Inc, http://www.componentware.com/arch_wp.htm, September 1995.

[IK89] R.M. Adler and B.H. Cottman. *A Development Framework for Distributed Artificial Intelligence*. The Fifth IEEE Conference on Artificial Intelligence Applications, Miami, Florida, March 1989.

[IK90] R.M. Adler and B.H. Cottman. *A Development Framework for Distributed Operations Support Systems*, Fifth Conference on AI for Space Applications, Huntsville, Alabama, May 1990.

[IK91] B.H. Cottman and R. Wood, *ObjectExpress: A System for Dynamic MetaData Exchange*. White Paper, I-Kinetics, Inc. Burlington MA 1991.

[IK96] B.H. Cottman. *A System for Large-Scale Dynamic Integration.* White Paper, I-Kinetics, Inc. Burlington MA, 1996.

[IK97a] Brian Cottman and Bruce Cottman, *The Future of Database Access With Netscape ONE, CORBA and JDBC*, May, 1997.
"http://developer.netscape.com/news/viewsource/cottman_JDBC.html"

[IK97b] B.H. Cottman, ComponentWare: Component Software for the Enterprise, March, 1997, "http://www.i-kinetics.com/wp/cwvision/CWVision.htm"

[IK97c] Bruce H. Cottman, *Universal Data Access*, Java Developer Journal, April, 1997.

[IK97d] Bruce Cottman, *Component Corner*, Object Magazine, July, 1997.

[IK97e] Ted Morin, *CORBA Legacy Integration*, Distributed Object Computing, Spring, 1997 and
"http://developer.netscape.com/news/viewsource/morin_corba/morin_corba.html"
September, 1997.

[NI95] Oscar Nierstrasz and Theo Dirk Meijler, *Research Directions in Software Composition.* ACM Computing Surveys, V27, N2, June, 1995.

[OMG95a] Object Management Group: "The Common Object Request Broker: Architecture and Specification", Revision 2.0, Object Management Group, July 1995.

[OMG95b] Object Management Group: "Common Object Services Specifications, Volume I", Document No 95-3-1 Object Management Group, March 1995.

[Shapiro94] M. Shapiro: "A binding protocol for distributed shared objects", 14th International Conference on Distributed Computer Systems (ICDCS), Poznan, Poland, June 1994.

[Spring93] G. Hamilton, M. Powell, J. Mitchell: "Subcontract: a flexible base for distributed programming", Proceedings of the 14th Symposium on Operating Systems Principles, Asheville NC, December1993.

[TINA94] N. Natarajan, F. Dupuy, N. Singer, H. Christensen: "Computational Modeling Concepts", Document TB_NAT.002_3.1_4, TINA Consortium, December 1994.