Introduction 1

With increasing global competition, corporations today are facing new challenges in bringing their products and services to market. The globalization of today's corporations, decentralization of corporate operations, and the accelerated rate of change in markets and competition are all factors representing both challenges and opportunities for today's businesses.

Modern companies are increasingly becoming information-based organizations, dependent upon the continuous flow of data for virtually every aspect of their operations. However their ability to handle data is breaking down because the volume of information is growing at a faster rate than their ability to process it. Further complicating matters is the difficulty that traditional monolithic applications have in representing an organization's business processes effectively — particularly as those processes or the organization itself changes.

Networked object technology such as Solaris™ NEO™ represent the best opportunity to meet the demands of agile organizations. Solaris NEO applications can effectively encapsulate business practices, policies, and tactics in modular components engineered to be re-engineered. Fast prototyping and updating capabilities encourage a development model that can design, test, and deploy a new application in less time and with less effort than traditional development methods. With Solaris NEO, legacy software can be encapsulated in objects, retaining its semantics and continuing to serve its users while it gracefully evolves into a fully networked application.

Software for Dynamic Enterprise Systems

The Solaris NEO product family fosters the rapid development and deployment of customized networked applications designed to interoperate with legacy data and software, with MS-Windows® desktops, and with other OMG-compliant object systems. Sun's leading multiprocessing (MP) and multithreading (MT) client-server technologies further provide an industrial strength platform with scalable performance to build networked object applications today.

Rapid Response to Change for a Competitive Advantage

Rapidly changing markets and business practices compel business to constantly improve policies and services. This translates into the need to create and modify software quickly. Unfortunately, traditional development techniques have slowed this necessary response to change. Solaris NEO enables custom, complex applications to be built, tested, enhanced, and deployed faster because they can be constructed from reusable components that have already been tested and proven.

To facilitate this streamlined approach to object application development, Solaris NEO provides a robust and complete infrastructure for shared networked services, including access to stored data, and a consistent, intuitive, rich graphical interface that facilitates the rapid development of new applications.

Object Technology Enables Application Flexibility

Object technology especially benefits businesses whose primary concern is the creation of custom applications. Using objects to abstract the logic, data, and networking, frees developers from dealing with time consuming implementation details and empowers them to concentrate on the application's semantics. Telecommunications, customer service, and financial industries are prime examples of businesses that can serve their customers better with custom applications that can be quickly changed to keep pace with market trends.

Reducing the Software Development and Maintenance Burden

The intrinsic inheritance of objects allows multiple levels of re-use, making it faster and easier over time to produce functions, programs, and entire systems from tested and optimized components. Built largely from existing object modules, new applications are easier to create, develop, deploy, modify, and maintain. Further, programming to a well defined interface definition that is independent of any specific language, operating system, or vendor allows true heterogeneous network interoperability.

Solaris NEO provides an extensive foundation of reusable components that form the basis for building new applications. Services to support the presentation and management of networked applications are built into its runtime environment, as are objects which can exploit Solaris' advanced multithreading capability. Building on these existing services encourages modification through a refinement development model that produces applications quickly, reducing the application development backlog that has troubled many MIS organizations.

Shared Services Enable the Networked Enterprise

Solaris NEO promotes an additional level of abstraction, viewing the application as components selected from three categories of objects that are readily combined for building flexible solutions. Separating the user interface and data access methods from shared business processes, practices, and policies, makes it easier to refine the application to meet changing requirements (see Figure 1). Because the user interface and back-end data are separated from the core application logic, changes are easier to implement in both the user interface and the application itself. Further, back-end data storage operations are now independent of the application, and hence can be designed to scale more easily and be upgraded when needed.

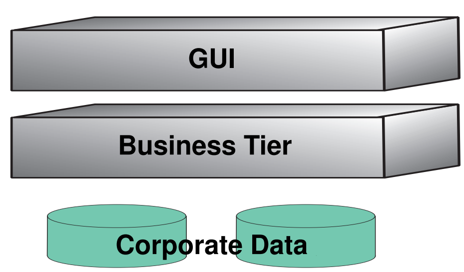

Figure 1 Three-Tier Computing Architecture

The Solaris NEO Operating Environment provides an excellent means to implement this architecture:

- *A consistent, visually rich user interface* for both general-purpose as well as custom applications provides an intuitive environment that helps leverage the user's time and productivity. Building on proven user interface objects helps facilitate fast prototyping and shortened development cycles.
- *Generic business objects may be developed first* followed by more specific sub-classes built from these original objects and inheriting functionality from their parents. These objects are then used to build custom applications precisely tailored to meet very specific needs. Separating business processes, practices, and policies from the rest of the application enables it to be refined dynamically to meet market and competitive demands, while at the same time delivering a superior quality product.
- *Data access may also be isolated* and is seen as the third element of this model. Whether using the built-in persistence offered by Solaris NEO, or the ability to access legacy data stored in Relational Database Management Systems (RDBMS) or Object Oriented Database Management Systems (OODBMS), objects may be built to handle any data requirement.

Multiple servers may be brought on-line dynamically, allowing the enterprise network to balance loads for peak periods. Reliability may be increased through distributed client-server replication management and tuned to any desired level of redundancy.

The NEO Product Family 2

A Comprehensive Solution

The built-in networking of Solaris is the foundation of SunSoft's object environment, and provides a transparent, high performance, distributed computing infrastructure. Industry-leading multiprocessing and multithreading technology ensures that Solaris NEO can scale to meet growing demands of enterprise-wide networked applications, while bundled Solstice NEO management tools allow total control of the object network from any location.

Advanced productivity tools available in Workshop NEO enable team software development with an object-oriented interface builder that speeds and simplifies the building of prototypes. Promotion of, and conformance to, industry standards further ensures connectivity with mainframes as well as with MS-Windows based desktops. Further, database connectivity and built-in persistence and encapsulation methods provide easy access to legacy data.

Figure 2 illustrates the five key dimensions of the NEO Product Family. Together, Solaris NEO, Solstice NEO, WorkShop NEO, and complementary MS-Windows and database connectivity technology, provide a comprehensive solution for enterprise application systems.

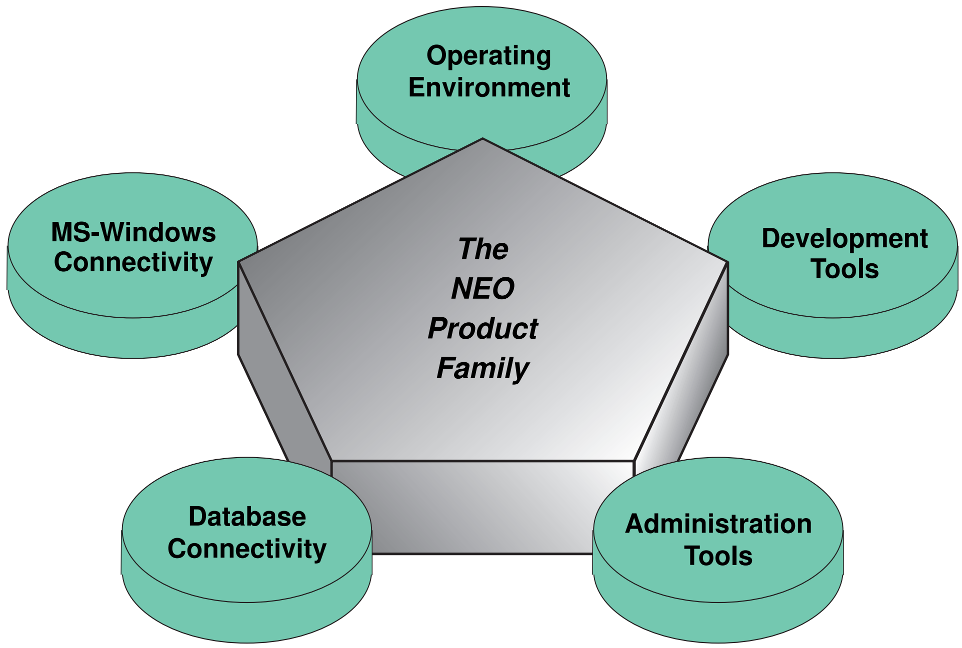

Figure 2 Solaris NEO Product Family

As illustrated in Figure 3, Solaris NEO delivers the means to implement and deploy a comprehensive set of services for the networked enterprise. With Solaris NEO, a three-tier computing architecture can be efficiently implemented that provides scalable, and flexible networked applications.

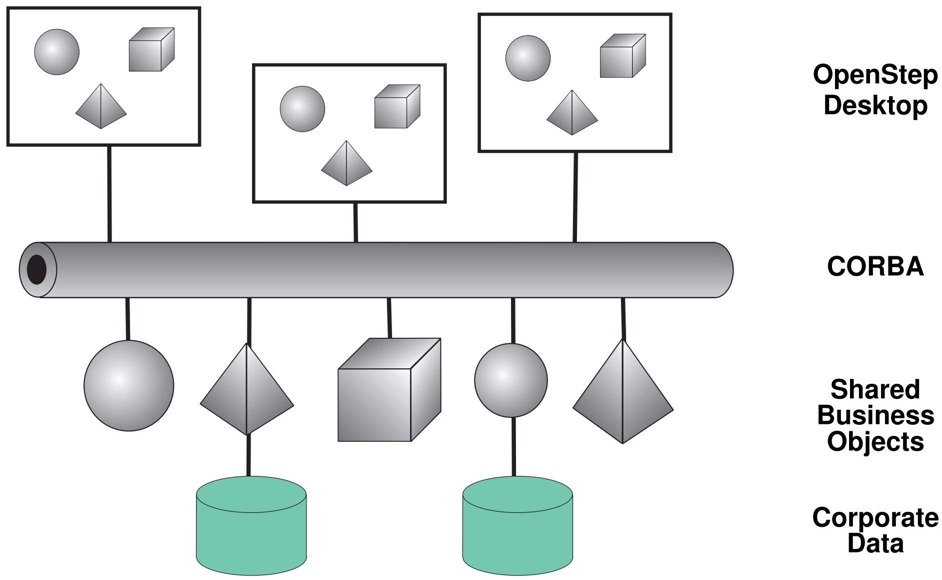

Figure 3 Solaris NEO components supports efficient three-tier computing

Standards for Interoperability

With a membership of over 500 software vendors, developers, and end users, the Object Management Group (OMG™) continues to develop the standards that will ensure a common architecture for heterogeneous, distributed, object-oriented applications. SunSoft was a founding member of the OMG, and has contributed many key specifications designed to ensure portability and interoperability between object system products from multiple vendors.

A major benefit of truly heterogeneous networked objects is the ability to integrate disparate platforms, allowing information resources to be shared across the network, independent of specific operating systems, languages, or implementation techniques. The cornerstone of this effort is the method by which interfaces to objects are defined, allowing behavior to be characterized without specifying the method used to implement it. Evidence of its commitment to both standards and object technology, SunSoft contributed the Interface Definition Language (IDL) to the OMG.

Through the availability of a standardized Inter-ORB Protocol (IOP), vendors can supply objects that interoperate in a heterogeneous environment. OMG's CORBA 2.0 standard Internet IOP (IIOP) is based on TCP/IP, and SunSoft provides a sample reference implementation free of charge, thus enabling vendors to be certain that their systems, objects and applications are interoperable.

Solaris NEO Operating Environment

Solaris NEO is a comprehensive operating environment that includes an intuitive, rich graphical desktop, a networked object infrastructure, an advanced runtime environment for shared services, and administration and management tools. Figure 4 illustrates the component structure of Solaris NEO. The shaded boxes represent shared runtime libraries.

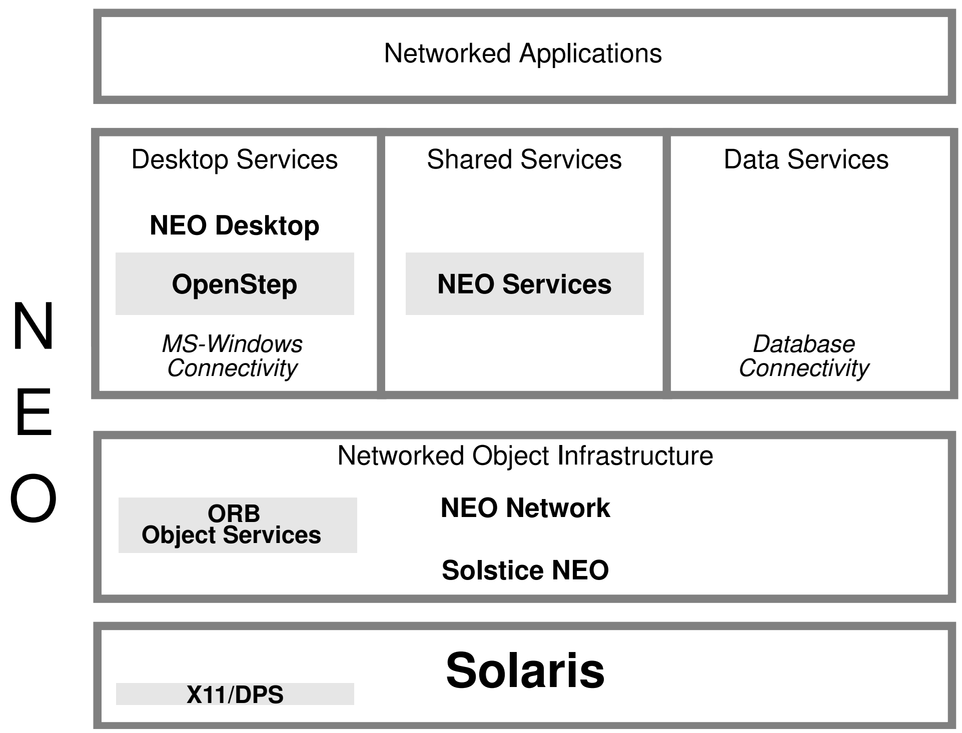

Figure 4 Solaris NEO components

NEO Desktop

The NEO Desktop is a user-friendly OpenStep-based application environment consisting of the following elements:

- Window Manager
- Desktop Applications
- Desktop Services

The ready-to-use desktop applications (accessories) build on capabilities available to all OpenStep-based applications. These capabilities facilitate application consistency and interoperability in areas such as fonts, color, and on-line hypertext-based help. The common desktop services are usable by any OpenStep application at runtime. These services, accessed programmatically via OpenStep frameworks, include pasteboard (for cut and paste), drag and drop, hyperlinks, spell checker, and print.

NEO Desktop provides the user with the same functionality as the NeXTSTEP Application Environment. In addition, NEO Desktop enables the continued use of the thousands of existing applications available for Solaris today. As an integral part of SunSoft's "universal desktop", this includes use of Common Desktop Environment (CDE) applications, MS-Windows applications using Wabi™, and Apple® Macintosh® applications running under the Macintosh Application Environment (MAE™). Interoperability such as drag-and-drop and cut-and-paste is supported between applications from all of these different environments. In addition to OpenStep applications, WABI, MAE, and CDE-Motif applications can be launched from the NEO Desktop. In turn, OpenStep applications can be launched from the CDE desktop.

NEO Desktop delivers a rich visual environment based on X11 with Display PostScript™ extensions and provides an intuitive, easy to use, high quality, multimedia user environment. Informative active icons, advanced drag and drop support, embedded multimedia capabilities, and integrated help facilities enhance the user experience and productivity. Interface consistency aids users in adapting to employing new applications quickly, and productivity is improved because the same functionality is available for all desktop applications. As an example, the NEO Desktop spell checker may be invoked from the text edit and electronic mail desktop applications, or any other custom application.

OpenStep

The OpenStep component of NEO consists of OpenStep-compliant development frameworks and associated shared runtime libraries. The OpenStep frameworks, supplied as Objective C class libraries, comprise the following:

- Graphical User Interface (GUI) Framework
- Application Framework
- Foundation Framework
- Enabling Framework

OpenStep frameworks provide powerful reusable application building blocks that can be used by developers in conjunction with the Workshop NEO Graphical Application Builder.

NEO Network

NEO Network is an OMG CORBA-compliant networked object infrastructure. It essentially provides an "operating system" for networked objects that is particularly suitable for pre-emptive multitasking, multithreading environments. NEO Network includes an Object Request Broker (ORB) and set of object services.

NEO Network's advanced technology is scalable, high performance and designed to form a solid platform for shared service computing. Networked administration and management facilities are built-in. In addition, NEO Network is architected to enable future system evolution.

NEO Services

NEO Services is key to enabling the shared service computing model. NEO Services is a comprehensive development framework and associated shared runtime libraries for networked objects and shared services. The NEO Services development framework is employed by the Workshop NEO Networked Object Constructor.

NEO Services automates and makes transparent functions that an application developer would normally need to write for areas including shared services, server availability, persistent object availability, concurrent requests, server management, and application installation. It enables full, simplified use of the NEO networked object infrastructure.

Completely unique to Solaris NEO, NEO Services is fully compatible with CORBA specifications, and provides the extended services required to facilitate rapid development of networked applications.

Solstice NEO

Management tools are key to the success of distributed computing environments. By providing network management from any location, enterprise management tools improve administrative capabilities and responsiveness while reducing costs.

Solstice NEO, bundled with Solaris NEO, complements this strategy and adds capabilities for the administration and management of networked applications and shared services to the Solstice product family. With Solstice NEO, resources can be monitored and controlled from any point on the network. System, workgroup, shared service, and application management capabilities are supported in the areas of:

- System Installation and Management
- Application Installation and Administration
- Workgroup and Shared Service Administration

A full set of tools provide for one-step system and application installation including incremental upgrades and there is extensive on-line help.

WorkShop NEO Development Environment

A rich environment for the development of applications is a major component of the NEO product family. SunSoft's award winning WorkShop has been extended to include tools unique to NEO. WorkShop NEO is an integrated development environment that includes tools for building networked objects and shared services and tools for building custom applications including GUI front-ends.

Figure 5 illustrates the component structure of WorkShop NEO. For additional details, consult the *WorkShop NEO Development Environment Product Overview* whitepaper.

NEOworks™

The NEOworks component of WorkShop NEO consists of:

- CORBA networked object development tools and the NEO Services development framework
- OpenStep graphical application development tools and the OpenStep-compliant GUI, Application, Foundation, and Enabling Frameworks

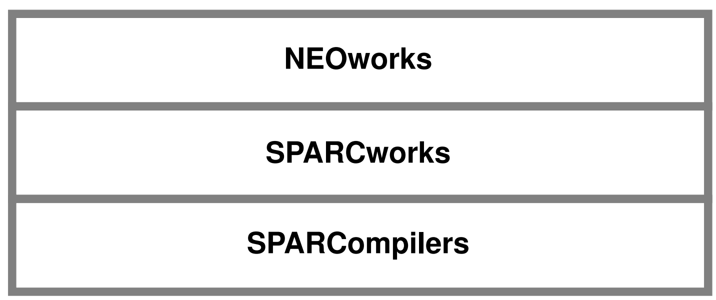

Figure 5 WorkShop NEO components

Tools for Constructing Networked Objects

Included in NEOworks are the Networked Object Constructor, IDL compiler, Networked Object Debugger, and NEO Services Development Framework. Coupled with the OpenStep graphical application development tools and frameworks, and SunSoft's SPARCompilers and SPARCworks tools, NEOworks enables developers to rapidly design, implement, test, and deploy shared services and complete networked applications.

Tools for Building Graphical Applications

SunSoft's NEOworks OpenStep development tools leverage the industry-leading NeXTSTEP™ object development environment. As part of NEOworks, they deliver a competitive edge to enterprises relying on custom software applications. Familiar tools such as Interface Builder, Project Builder, and Icon Builder facilitate the prototyping and deployment of graphical front-end applications with less effort and at lower cost.

SPARCworks and SPARCompilers

WorkShop NEO also includes advanced SPARCworks developer productivity tools and SPARCompilers. This includes tools for C, C++ and Objective C programming, tools for multithreaded programming, and tools to support team development.

MS-Windows Connectivity

Today's enterprise environment includes MS-Windows desktops, mainframes with legacy databases, as well as workgroup LANs. A successful software environment must provide access to and seamlessly integrate these technologies into a coherent conduit for information exchange. SunSoft's own experience in deploying heterogeneous environments has provided us with the needed vision to begin forging the standards and relationships with key technology leaders to ensure this transparent connectivity.

Complementary technology from IONA Technologies, in combination with Solaris NEO, provides connectivity with MS-Windows desktops including interoperability with OLE and the underlying Component Object Model (COM). This technology, based on IONA's Orbix™ product, allows MS-Windows applications to access NEO objects and shared services and act as networked enterprise application front-ends. Network communication is based on OMG's CORBA 2.0 Interoperability specifications (see Figure 6).

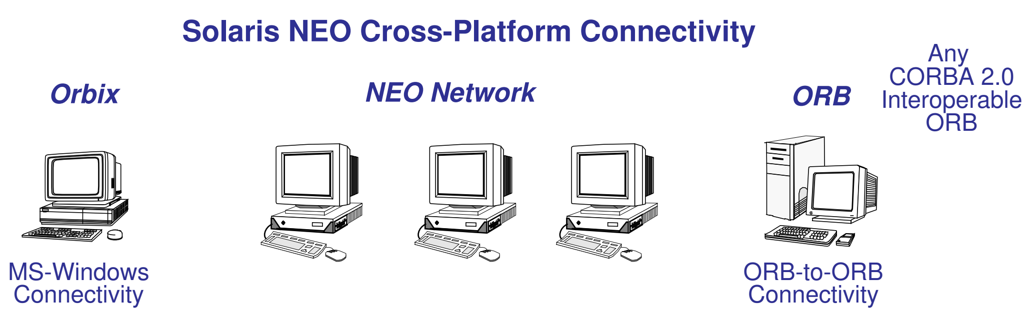

Figure 6 Solaris NEO cross-platform connectivity

Database Connectivity

A wide range of options are available in NEO to meet data storage, and database access and integration needs. The built-in Solaris NEO persistence support, based on technology licensed from Object Design, Inc., provides a mechanism for applications with simple object data storage requirements. For more complex needs, application developers can employ existing Relational Database Management Systems (RDBMS), or Object Oriented Database Management Systems (OODBMS).

Complementary technology from Persistence™ Software, in combination with Solaris NEO, provides mechanisms for automatically mapping objects to relational tables, ensuring data integrity by enforcing object constraints and controlling relational transactions, as well as scaling to multiprocessor and multiple database implementations.

As a founder of the Object Database Management Group (ODMG), SunSoft is promoting standards for interoperability between OODBMS providers. The ODMG-93 specification defines interfaces to object databases that ensure heterogeneous interoperability, object portability and reuse, as well as concurrent access.

NEO Desktop 3

The NEO Desktop is a user-friendly OpenStep-based application environment consisting of the following elements:

- Window Manager
- Desktop Applications
- Desktop Services

The ready-to-use desktop applications (accessories) build on OpenStep capabilities available to all OpenStep-based applications. These capabilities facilitate application consistency and interoperability in areas such as fonts, color, and on-line help.

The common desktop services are usable by any OpenStep application at runtime. These services, accessed programmatically via OpenStep frameworks, include pasteboard (for cut and paste), drag and drop, hyperlinks, spell checker, and print.

The NEO Desktop also provides interoperability with other ICCCM-compliant application environments. WABI, MAE, CDE-Motif and OpenStep applications can be launched from the NEO Desktop. In turn, OpenStep-based applications can be launched from the CDE desktop.

Window Manager

The NEO Desktop offers a NeXTSTEP/OpenStep look and feel that supports the powerful active window model, icon paradigm, and drag and drop direct manipulation for icons and other GUI elements.

The NEO Desktop Window Manager is a X11-based ICCCM-compliant window manager. This provides consistency and interoperability in the areas of:

- Common window hierarchy
- Key and mouse mappings
- Application launch (double click)
- Cut and paste (simple text)
- Drag and drop

Desktop Applications

NEO Desktop includes a broad set of ready-to-use desktop applications built on OpenStep's capabilities. Described below, these applications include Workspace Manager, Text Edit, Multimedia Mail, User Preferences, PostScript Preview, and Terminal.

Workspace Manager

The NEO Desktop Workspace Manager is a customizable graphical user environment for managing and using network-wide documents and applications. It includes an application dock which manages applications in icon form, and from which frequently used applications can be launched. It also includes a file viewer to navigate the network with list, browser, and iconic views and to open files in their associated applications.

Figure 7 illustrates the NEO Desktop Workspace Manager.

Text Edit

Edit is a mouse-based text editor for creating and editing ASCII and Rich Text Format (RTF) files. It handles both TIFF and EPS images, which together with other data types, may be embedded in a document using drag and drop.

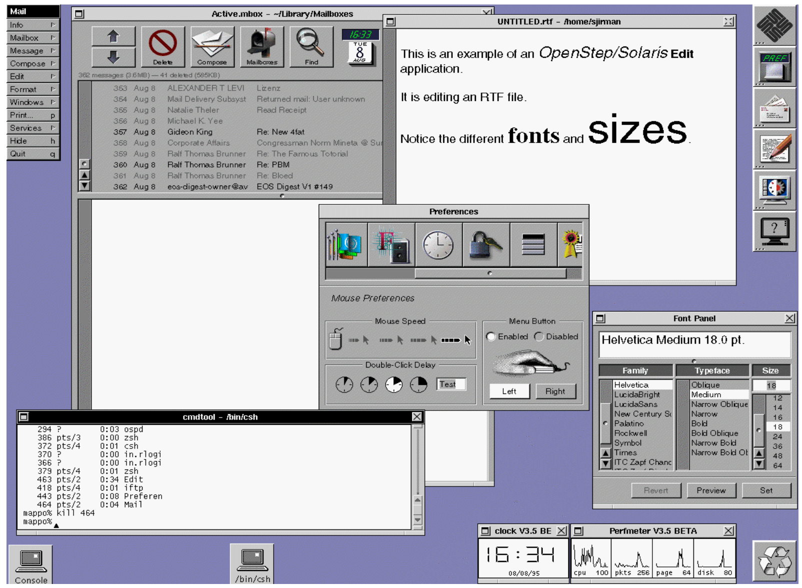

Figure 7 NEO Desktop Workspace Manager

Multimedia Mail

Mail is an electronic multimedia mail tool compatible with UNIX mail. It supports MIME formats and Rich Text Format (RTF) text and file attachments (including Sun attachments). Images and other data types can be embedded in a message using drag and drop.

User Preferences

Preferences provides a graphical interface to change personal NEO Desktop and workspace preferences. Customizable preferences include screen background color, system font types and sizes, language preferences, key mappings, and mouse button placement.

PostScript Preview

Preview provides the ability to display PostScript files. It may also be used to preview TIFF files by first converting them into PostScript.

Terminal

Terminal provides a UNIX-style terminal capability integrated into the OpenStep-based application environment.

Desktop Services

NEO Desktop includes a flexible set of common desktop services usable by any application at runtime. Described in the following sections, these services include pasteboard, drag and drop, hyperlinks, spell checker, and print.

Pasteboard

The Pasteboard Service enables cut and paste between documents of mutually understood data types.

Drag and Drop

The Drag and Drop Service enables the direct manipulation of icons and other GUI elements, with associated context-specific behavior.

Hyperlinks

The Hyperlinks Service enables data (text and other data types) in different documents to be linked together and the links navigated. Documents can originate from different applications.

Spell Checker

The Spell Checker Service enables applications to access a 100,000 word Houghton Mifflin dictionary (in 16+ languages). The dictionary can be updated to learn new words.

Print

The Print Service manages access to system printing resources.

OpenStep 4

The OpenStep component of NEO consists of OpenStep-compliant development frameworks and associated shared libraries that make up the OpenStep runtime system in Solaris NEO.

The OpenStep frameworks, supplied as Objective C class libraries, comprise the following:

- Graphical User Interface (GUI) Framework
- Application Framework
- Foundation Framework
- Enabling Framework

OpenStep Runtime System

The OpenStep runtime system is most easily understood in terms of the development frameworks it supports. The following sections describe these frameworks. OpenStep frameworks provide powerful reusable application building blocks that can be used by developers in conjunction with the NEOworks Graphical Application Builder. The frameworks are extensible by subclassing or delegation (via hooks to dynamically add new components) and feature the following capabilities:

- Rich data-type support including images and multi-attributed text suitable for supporting multimedia documents
- Same imaging model for screen and hardcopy (via Display PostScript)
- Common fonts and printing management

- Consistent user help model
- Abstractions for cross-platform program portability
- Support for Sun "Level 3" internationalization for end-user, developer, and administrator

The AppKit class library, SunSoft's implementation of the OpenStep GUI and Application Frameworks, is based on the X11 Display PostScript (DPS) capability supplied as part of Solaris. AppKit provides a rich set of standard objects useful for all applications, with a wide variety of customization options. Included is a complete set of user interface objects and controls, as well as objects that support data sharing and inter-application communication.

AppKit is similar in overall function to the CDE-Motif toolkit. It incorporates DPS rendering within top level X Windows and supports Alpha compositing. It provides extremely sophisticated shading and transparency effects beyond the simple 3D shading provided by most X Windows toolkits. Other extended features include support for lower level data types available in the OpenStep Foundation Framework (implemented as the FoundationKit class library).

OpenStep GUI Framework

The OpenStep GUI Framework provides reusable, customizable objects for building and using windows, panels, structural views, control views, images, colors, text, and fonts. Table 1 lists the GUI Framework objects and their classes.

Objects	**Classes**
Window	Window, Screen, Event
Panel	Panel, Menu, ActionCell, MenuCell, PopUpList,
Structural View	View, Box, SplitView, ScrollView, ClipView
Control View	Control, Cell, BrowserCell, Browser, Button, PopUpButton, ButtonCell, Slider, SliderCell, Scroller
Image	Image, ImageRep, BitmapImageRep, EPSImageRep, CustomImageRep, CachedImageRep, Cursor
Color	Color, ColorList, ColorPicker, ColorPanel, ColorWell
Text	Text, CStringtext, TextField, TextFieldCell
Font	Font, FontManager, FontPanel

Table 1: OpenStep GUI Framework objects and their classes

Image classes support both TIFF and EPS formats. Predefined window panels for incorporation in applications are provided for selecting colors and PostScript fonts, and controlling font size, weight, and other properties.

OpenStep Application Framework

The OpenStep Application Framework provides reusable, customizable objects for managing and manipulating workspaces, forms, help, filing, printing, spelling, data exchange (for cut and paste, and drag and drop), and data links (hyperlinks). Predefined window panels for incorporation in applications are provided for help (enabling consistent, context-sensitive, hypertext linked, on-line help), opening and saving files, printing documents, creating and navigating hyperlinks, and for checking spelling and building user dictionaries. Table 2 lists the Application Framework objects and their classes.

Objects	**Classes**
Workspace	Workspace
Form	Matrix, Form, FormCell
Help	HelpPanel
File	SavePanel, OpenPanel
Print	PrintInfo, PrintOperation, Printer, PrintPanel, PageLayout
Spell	SpellChecker, SpellServer, SpellPanel
Data Exchange	Pasteboard
Data Link (Hyperlink)	DataLink, DataLink Manager, DataLink Panel, Selection

Table 2: OpenStep Application Framework objects and their classes

OpenStep Foundation Framework

The OpenStep Foundation Framework provides reusable, customizable objects that support basic program functions. This includes operation processing, creating and accessing basic structured and stored data, thread control, and obtaining information about the program runtime environment. Table 3 lists the Foundation Framework objects and their classes.

Objects	Classes
Operation Processing	Object, Invocation, MethodSignature, Exception, AssertionHandler RunLoop Notification, NotificationCenter, NotificationQueue Proxy, DistantObject
Structured Data	Array, MutableArray, Number, Value, Enumerator String, Scanner, MutableString Set, MutableSet, CountedSet Data, MutableData, Serializer, Deserializer CharacterSet, MutableCharacterSet
Stored Data	BTreeBlock, BTreeCursor ByteStore, ByteStoreFile Dictionary, MutableDictionary Coder, Archiver, Unarchive Date, CalandarDate TimeZone, TimeZoneDetail
Thread Control	Thread, Lock, Connection, RecursiveLock, ConditionLock
Program Environment	ProcessInfo, Timer, UserDefaults, Bundle, AutoReleasePool

Table 3: OpenStep Foundation Framework objects and their classes

OpenStep Enabling Framework

The OpenStep Enabling Framework provides reusable, customizable objects that enable the integration of networked objects.

Callback Facility

The Callback Facility supports the ability for OpenStep-based Objective C programs to call networked objects and properly handle the return results through the use of callback objects. A callback object is a limited form of networked object.

AppKit Synchronizer

NEO application code can be multithreaded, but most libraries are not MT-safe, including the AppKit. Only a single thread in a process may safely access the AppKit at any one time. On its own, AppKit cannot properly handle multiple threads arising from separate NEO Network ORB requests from the same process. The AppKit Synchronizer can be used by applications to serialize the request processing. This is sufficient for simple updates.

NEO Network 5

The NEO Network component of Solaris NEO is an OMG CORBA-compliant networked object infrastructure consisting of an Object Request Broker (ORB) and set of object services. NEO Network is mature, tested technology. The NEO Early Developers Release (EDR) was first distributed in June 1993, followed by EDR-2 in May of 1994.

NEO Network provides a robust, networked object infrastructure that enables object-to-object communication. It is the traffic controller and message router between objects and provides application developers with the services necessary to create systems of cooperating objects, shared services, and integrated networked applications.

The design center of NEO Network can be summarized as follows:

- *Standards based*
- *Scalable*
- *High performance*
- *Multithreading*
- *Shared service computing*
- *System evolution*
- *Developer productivity* (through built in support for WorkShop NEO)
- *Networked administration and management* (through built in support for Solstice NEO)

NEO Network supplies the mechanisms required for objects to transparently communicate and interact with each other across the network. Traditional techniques require an intimate knowledge of networking to distribute an application. Eliminating the need for complex programming, NEO Network considerably reduces the knowledge and effort required to develop and maintain networked applications.

Informal Definitions

The following informal definitions may assist in understanding the features and capabilities of NEO Network:

Networked objects have OMG IDL interfaces and are accessed using an *object reference.*

Networked services (or simply services) are collections of networked objects that provide coherent chunks of shared functionality for clients.

Factories are services that create objects.

Server programs are executables that contain the implementations for one or more object types.

Server processes (or simply servers) are server programs running in a Solaris process.

Object instances (or simply objects) are specific instantiations of an object type with its own state.

Introduction to NEO Network

Leadership of OMG CORBA Standards

SunSoft has authored or coauthored all the core OMG CORBA specifications to date, and 9 of the 11 CORBAservices. The front-end of SunSoft's OMG IDL compiler has served as the industry's reference implementation since SunSoft made the source code for it freely available on the OMG Server in 1992. In March 1995, with the goal of ensuring that all ORB vendors can interoperate using the recently adopted OMG CORBA 2.0 Universal Networked Objects specification, SunSoft supplied in source form and free of licensing charges, a portable reference implementation of the mandatory inter-ORB protocol for networked ORBs — the Internet IOP (IIOP).

Scalable Technology

NEO Network supports a wide variety of objects (large and small, persistent and transient, local and remote) and accommodates large numbers of objects and computers. There is no hard limit to the number of concurrent requests. Additionally, there are no restrictions on the number of interfaces or implementations per server process. There is no cost to adding a computer to the system in that there are no required central points nor knowledge required among existing computers.

NEO Network supports flexible activation of server processes, object implementations, and objects. Objects and server processes can be deactivated manually or automatically by setting an idle timeout. Queued requests are sent to a newly activated server process. Client programs notice nothing other than a slight delay.

All NEO object references are automatically persistent across server process activations which means that client programs do not need to maintain any additional information about objects in order to access them. NEO Network also provides a form of fine-grained object — *subobjects* — which are particularly useful for the efficient support for large numbers of objects stored in databases. An unlimited number of subobjects can share the normal memory and resources required for a single object reference.

High Performance Through Optimization

Multithreading (MT) was architected into NEO Network from the beginning. MT is required by the shared service computing model to allow effective sharing of services and information resources and for scalability, particularly to support high performance server-based applications.

Overhead in the NEO Network ORB has also been minimized. Once communication is established between the client and the server process, the ORB gets out of the way. NEO Network features same-process and inter-process optimization. Networking code is bypassed for same-process objects and shared memory is used. Extensive caching is done to optimize such things as location lookup, connection reuse, and object adapter information.

Subobjects allow faster access to fine-grain objects. From the client program point of view, object references for subobjects are indistinguishable from normal objects.

Multithreading Throughout

NEO Network uses mutlithreading extensively internally and allows both clients and server programs to be fully multithreaded. A thread is automatically created for each incoming request. Clients may spawn threads and issue concurrent requests. Additional supporting APIs for object implementations are also provided (e.g., to quiesce threads). Proper use of MT enables simpler and safer programming.

Platform for Shared Service Computing

NEO Network fully supports the shared service computing model. In NEO Network, server processes can support many objects, objects of different interfaces, and different implementations of an interface. This flexibility is necessary for packaging, security, and performance reasons.

A rich set of object services is included that form building blocks for applications composed of networked objects. These services also form the basis for application interoperability and integration. Generic tools and frameworks can be developed to use and manipulate objects and services that conform to the standardized object service interfaces — without any other special additional knowledge about their functionality. Sun "Level 4" internationalization is also supported.

Architected for System Evolution

NEO Network has been architected with an number of features that facilitates future system evolution and, in particular. the introduction of SunSoft's Spring technology such as "DOORS", "M-Tables" and "subcontracts". DOORS, a high speed IPC mechanism, will provide significantly improved same-process and inter-process performance. M-Tables will provide capabilities beyond current dynamic dispatch mechanisms. Subcontract provides a general capability that can be used to support such things as dynamic protocol stack selection.

NEO Network's use of and support for opaque object references is crucial for making transparent changes in such areas as security and replication support and for providing dynamic system upgrades. The internal architecture of NEO Network is based on virtual layering techniques that allow future transparent introduction of capabilities such as object and service replication and high availability. In addition, all NEO Network protocols and file formats are versioned. The system is designed to be dynamically upgradable.

Security is a key aspect of the Spring operating system. SunSoft is working within OMG to define a standardized security solution which we plan to support in a future release of NEO Network.

NEO Network ORB

Networked Object Model

The NEO Network Object Request Broker (ORB) implements and supports the object-oriented paradigm applied to networked objects. This includes the encapsulation of behavior and state (whereby implementation details are hidden), interface inheritance (interfaces can be extended — implementations can be customized), and polymorphism (operation names are scoped to the object definition so that generic framework code can work on application-specific objects).

The ORB provides a uniform way of defining and accessing objects in the same process, on the same computer, or across a network. Interfaces to objects are separated from their implementation. This allows the same mechanisms to be used to access all services — everything becomes an object.

Access to networked objects is by means of object references. Object references are opaque data structures generated by the ORB which are used to identify the target object of a request. They contain enough information for the ORB to efficiently find the object. Unlike RPC mechanisms, this is a dynamic process — object location and communication are not hard-wired.

OMG Interface Definition Language

The OMG Interface Definition Language (IDL) provides a standardized way to define the interfaces to networked objects. IDL is a strongly typed declarative language based on C++ syntax. Strong typing is essential for building large, robust, long-lived systems. The IDL definition is the contract between the implementor of an object and the client.

IDL Compiler

NEO Network includes an IDL compiler that is architected as a single front-end and back-ends supporting the C and C++ language mappings and for loading information into the ORB Interface Repository. Upwards-compatible SunSoft extensions include interface versioning and support for additional data types including 64-bit ints, long dbls, wchars, and wstrings.

Programming Language Mappings

IDL language mappings provide a standardized way to access networked objects using the developer's programming language of choice in a style that is natural to that language. Operations on networked objects are invoked in a way that is consistent with the particular programming language being used. For example, a networked object is accessed in C++ using a normal C++ object method invocation.

NEO Network supports C and C++ mappings including full **Any** support (all CORBA data types). In addition, C++ interfaces are provided for interpreting **TypeCodes**.

ORB Operations

The NEO Network ORB *core* provides the location and transport mechanisms necessary to deliver requests from a client to an object wherever the server program and process for the object's implementation reside. The current location of the object on the network (including on the same computer) is determined at runtime when the object is first accessed.

The remote transport is based on TLI. Local transport is based on shared memory. Client cancellation is supported for both remote and local transport. Retry and rebind exceptions are handled, allowing clients to resubmit requests if a server process crashes or the ORB's Basic Object Adaptor (BOA) becomes unavailable.

The NEO Network ORB provides the basic `CORBA::Object` and `CORBA::ORB` APIs. These provide operations for such things as stringify/destringify object references and translations functions for **TypeCode** and **Any**. The *Environment interface* handles exception data. The *Principal interface* supplies a client invoker's ID.

The NEO Network ORB also provides built-in administration interfaces including those for BOA database inspection, consistency checking, backup and restore, and for obtaining server process runtime addressing information for debugging purposes.

Basic Object Adapter

The Basic Object Adapter (BOA) is the part of the ORB that is the conduit between the object implementation, server process, and the ORB core. It allows the ORB to locate, activate, and invoke operations on a networked object. The NEO BOA supports a wide variety of common styles of object implementations.

The BOA causes the activation and deactivation when needed of server processes, object implementations and instances, and dispatches calls to methods through "skeleton" code. Multiple activation models are also supported.

The BOA is also responsible for generating and interpreting objects references and the maintenance of object reference information including up to 1KByte of associated persistent Reference Data. NEO object references are automatically persistent across server process activations.

In addition to the standardized BOA API, the NEO BOA provides upwards-compatible extensions that include support for subobjects and the ability to block new requests to enable MT coordination for object-critical sections.

Interface Repository

The Interface Repository is a fully distributed database integrated with the NEO Network ORB. Interface definitions are located where the objects are located so that there are no bottlenecks. Fully logical interface IDs are employed. APIs are provided to allow the Interface Repository database to be loaded, updated, backed-up, and restored.

The Interface Repository supports full type safety and internationalization and enables runtime type checking. Distributed application evolution is provided via major-minor interface version numbering and interface inheritance (and programmatic use of narrowing).

Dynamic Type Support

The Dynamic Invocation Interface (DII) of the ORB enables the runtime discovery of installed interfaces and the dynamic construction of operation invocations. DII is particularly useful for generic object browsers.

The Dynamic Skeleton Interface (DSI) enables the delivery of requests to an object implementation that does not have compile-time knowledge of the type of the object it is implementing. This allows all requests on one or more object types to share the same invocation code. This code can use narrowing with runtime type checking using the Interface Repository. DSI is useful for building inter-ORB bridges, debugging, interposing of objects, implementing objects with interpreters and scripting languages, and dynamically generating implementations.

Inter-ORB Communication

SunSoft took a leadership position and has been very active in contributing to the development of a standard inter-ORB protocol at OMG. SunSoft contributed bridging and protocol technology to the Universal Networked Objects (UNO) specification that was adopted by OMG in 1994.

SunSoft has supplied, in source form and free of licensing charges, a portable reference implementation of an engine for the CORBA 2.0 mandatory inter-ORB protocol (the Internet IOP) for networked ORBs. (The IIOP is the TCP/IP transport mapping of the CORBA 2.0 General IOP or GIOP.) This software formed the basis for the multivendor interoperability demonstration at the 1995 ObjectWorld, in San Francisco.

The Internet IOP Engine does the work to generate, receive, and handle the CORBA 2.0 standard protocol and is composed of four parts: a CDR (Common Data Representation) marshaling engine, a **TypeCode** interpreter, an Internet IOP Engine framework, and modules that enable the engine framework and the CDR marshalling engine to send, receive, and dispatch Internet IOP messages.

The Internet IOP software is written in C++ and is highly portable. The software has been compiled using: SPARCworks (for SPARC) and PROworks (for x86) C++ 4.0.1, Borland C++ 4.5, GNU C++ 2.6.3, and Visual C++ 2.0 on the following operating system platforms: Solaris 2.4 (both SPARC and Intel platforms), SunOS 4.1 on SPARC, Linux (1.1.47 and later) on 486 hardware., NEXTSTEP 3.2, Windows NT and Windows 3.5. To retrieve the software, send an electronic mail message with the subject *help* to *iiop-bridging@omg.org* and the mail server will respond with instructions, or use anonymous FTP to connect to the *ftp.omg.org* server.

NEO Network Object Services

The NEO Network Object Services are building blocks for applications composed of networked objects. They also form the basis for application interoperability and integration by enabling application components to be designed and implemented separately yet work together in useful ways.

As a particular example of this, the NEO Network Object Services allow generic tools and application frameworks to use and manipulate independently-developed off-the-shelf objects and services. Such tools and frameworks need not have any other special additional knowledge about the objects and services other than that they conform to the standardized object service interfaces.

NEO Network provides implementations of a rich set of generally useful services:

- Naming Service
- Event Service
- Property Service
- Relationship Service
- LifeCycle Service

Other OMG CORBAservices planned for future releases include:

- Externalization Service
- Transactions Service
- Query Service
- Security Service

Naming Service

The NEO Naming Service provides a standardized way of storing and retrieving object references by name. It provides the ability to bind a name to an object reference relative to a *naming context*. A naming context is an object that contains a set of name bindings in which each name is unique. To r*esolve a name* is to determine the object associated with the name in a given context.

Because naming contexts can be named in other naming contexts, the Naming Service supports hierarchical naming schemes for objects and naming contexts. Graphs of naming contexts can be supported in a federated fashion. This scalable design allows the distributed, heterogeneous implementation and administration of names and naming contexts.

A canonical representation for compound names is used such that no particular name syntax is mandated. This, and the provision of a name "kind" attribute, facilitates application localization and places minimal constraints on higher-level name policies.

Event Service

The NEO Event Service provides capabilities for asynchronous event notification between event producers and consumers that can be configured together in a very flexible and powerful manner. Event channels decouple suppliers and consumers, and support multiple suppliers and multiple consumers. Suppliers can generate events without knowing the identities of the consumers. Conversely, consumers can receive events without knowing the identities of the suppliers.

Push-style and pull-style delivery models and event *fan-in* (collect) and *fan-out* (multicast) are supported. Event content is IDL type **Any**. Typed event channels extend basic event channels to support typed interaction. Extended interfaces also allow event channels to be chained or piped together by third-parties without the involvement or knowledge of the suppliers or consumers.

NEO Network includes two Event Service implementations with different qualities-of-service: (1) fully persistent, and (2) transient events, persistent connections. Their design is scalable and is particularly suitable for distributed environments. There is no centralized server or dependency on any global service.

Property Service

The NEO Property Service is a simple, extensible service that enables properties to be dynamically associated with any object independent of its static IDL interface attributes. Properties can be added, modified, deleted and retrieved without the involvement of the associated object.

A *PropertySet* is a "first class" networked object that maintains a set of key-value pairs. The keys are strings and values are IDL type **Any**. The PropertySet interface defines operations to create properties, get and set property values, delete properties, enumerate the properties in the set, and destroy the property set. PropertySet objects can easily be shared with other applications.

Relationship Service

Based on the entity-relationship model, the NEO Relationship Service provides a standardized way of linking networked objects. One-to-one, one-to-many, and many-to-many binary relationships are supported.

Two common kinds of relationships are predefined: *containment* and *reference*. Examples of these are a document that contains an image and a table, and a document that references an appendix. The service is extensible and developers can define additional kinds of relationships.

The service defines two new types of objects: *relationships* and *roles*. A role object represents an object in an relationship. A relationship object is created by passing a set of roles to a relationship factory. Type and cardinality constraints can be expressed and checked and referential maintenance is supported.

Relationships are established in a way that does not require the involvement of objects being related. Because relationships are first class objects themselves, third parties (e.g., utility tools and object browsers) can traverse and manipulate the connections between the objects. Navigation performance and availability is comparable to the direct use of object references; role objects can be collocated with their objects and need not depend on a centralized repository of relationship information. As such, navigating a relationship can be a local operation.

The Relationship Service also supports traversals of graphs of related objects and thus provides the basis for compound operations.

LifeCycle Service

The NEO LifeCycle Service establishes conventions for creating, deleting, copying and moving objects. Because objects can be networked, the service accommodates life-cycle operations on objects in different locations.

The client program's model of creation is defined in terms of *factory* objects. A factory is an object that creates another object. As with any object, factories have well-defined IDL interfaces. The LifeCycle Service also includes the definition of an interface for a generic factory. This allows for the definition of standard creation services.

Compound life-cycle operations address copying, moving and deleting objects that are connected to other objects by relationships. Operations are propagated in one of three ways: *none*, *shallow*, and *deep*. Each type of relationship defines the required propagation behavior (e.g., copying of contained objects but not referenced objects).

NEO Services 6

NEO Services is a comprehensive development framework and associated shared libraries for networked objects and shared services. The development framework is employed by the NEOworks Networked Object Constructor. The shared libraries make up the NEO Services runtime system in Solaris NEO.

Unique to NEO, NEO Services is fully compatible with CORBA and enables full, simplified use of the NEO Network object infrastructure. It automates and makes transparent functions that an application developer would normally need to write in the areas of:

- Workgroup Support
- Shared Service Finder
- Server Availability
- Persistent Object Availability
- Data Store Manager
- Concurrent Requests
- Implementation Support
- Server Management
- Application Installation

Table 4 summarizes the functions of NEO Services.

Function	Description
Workgroup Support	Establishes and implements standardized workgroup/computer naming policies, and allows access to shared services
Shared Service Finder	Registers and finds services based on NEO Network Naming Service, and enables relocation of services without changing or recompiling client code
Server Availability	Simplifies server activation, provides transparent management of object implementations grouped in a server program, and handles housekeeping functions
Persistent Object Availability	Provides transparent management of persistent objects, manages application-independent ORB-related housekeeping, automates support for life-cycle create and destroy and object instance and implementation activation
Data Store Manager	Provides transparent persistence for object state
Concurrent Requests	Provides transparent management of concurrent requests to multithreaded object implementations, and supports multiple locking policies
Implementation Support	Provides functions that simplify and make easier development of object implementations
Server Management	Provides server and application management functions and transparent support for management objects installed as part of applications
Application Installation	Provides transparent support for application installation

Table 4: NEO Services functions

NEO Services Runtime System

Workgroup Support

The NEO Workgroup Support establishes and implements standardized workgroup and computer resource naming policies. The workgroup is the unit for administration, management and sharing in NEO. A workgroup is a collection of computers that can include workstations and server machines. Each user may select which of their own resources may be shared by the workgroup, and which workgroup services are to be used instead of local resources. A user can have a private part of the workgroup resources reserved for their development or other purposes.

Workgroup Support allows access to shared services by enabling services to be registered in a well known place and found using the NEO Services Shared Service Finder. In this way, the need for applications to have a detailed knowledge of the system name space is eliminated.

Shared Service Finder

The NEO Shared Service Finder employs a federated approach to registering and finding services based on the NEO Naming Service. Building on the Workgroup Support resource naming policies, it uses a federation of predefined and application-specific naming contexts.

Services (i.e., named objects) are registered at install time or runtime in a well known place. Applications can then dynamically find a service that is available to run in an appropriate server process. A service can be local to a computer or shared by a workgroup. A mode can be selected that controls whether a deployed service or a version under development is found.

The Shared Service Finder enables the relocation of services without needing to modify or recompile client code. In this way, service requests can be distributed to the most appropriate resource in a workgroup and dynamic load balancing is facilitated.

Server Availability

The NEO Server Availability builds on and simplifies NEO Network ORB server process activation. It provides the transparent management of the availability of object implementations grouped in a server program and takes care of application-independent ORB-related housekeeping functions that the server program developer would otherwise need to provide. With NEO Services, server program developers need only write minimal application-specific “server availability” code if needed.

Server Availability automates server process startup on arrival of a request for any object in a server program as well as server process shutdown after a period of inactivity. The timeout (i.e., maximum idle period) is configurable per server process. The process waits for all objects in the server process to be deactivated before shutdown occurs. The wait interval and shutdown retry cycle time are configurable.

Because server availability is handled in a standardized way, the system can automatically manage and recover resources such as memory. In conjunction with the NEO Services Persistent Object Availability, the use of system resources is minimized.

Persistent Object Availability

The NEO Persistent Object Availability builds on and simplifies NEO Network ORB object activation. It provides transparent server-side management of the availability of persistent objects and takes care of application-independent ORB-related housekeeping functions that the object developer would otherwise need to provide. With NEO Services, object developers need only write minimal application-specific "persistent object availability" code if needed.

Persistent Object Availability automates support for object life-cycle create and destroy, and object instance and implementation activation. It automates the activation of the object implementation and instance on the arrival of a request. A *servant* C++ object implementing the operations of the object's interface, transient data and other functions is automatically instantiated when an object is activated, and destroyed when the object is deactivated.

Persistent Object Availability also automates the deactivation of objects after a configurable period of inactivity. The timeout (i.e., maximum idle period) is configurable per object implementation. It transparently handles thread quiescing and waits for all pending operations to complete using a configurable wait and retry cycle time.

Optional transparent persistence of an object's state is provided based on the NEO Data Store Manager. This provides automatic atomic update at timed intervals and before deactivation of the object. The time interval is configurable per implementation. In addition, an infrastructure is provided to support custom persistence. All the hooks necessary to incorporate custom persistence mechanisms are provided to address cases where more control is needed over performance and the handling of data formats and legacy data.

Data Store Manager

Used in conjunction with Persistent Object Availability, the NEO Data Store Manager provides transparent persistence for the state of an object. It supports the model that allows networked objects to be implemented by code organized as servant objects.

Based on technology provided by Object Design, Inc., the Data Store Manager is designed to efficiently support development models in which applications are composed of many fine-grain objects comparable in size and complexity to typical C++ objects. Data objects can contain pointers to other data objects, allowing users to create persistent representations of complex, linked data structures in a natural manner.

A subset of IDL, called the Data Definition Language (DDL), is used to define the persistent state in terms of "data objects". The IDL language binding approach is used to provide client bindings for a wide range of architectures and programming languages. Stored data maintains the same level of type safety as that provided by IDL.

Applications can create "data stores" (the unit of storage) in any part of the file system to which they have access. Transparent to clients, data objects are cached in local memory. Access to individual attribute values is essentially at the speed of native programming language calls. Modifying the persistent state of an object is done by simply modifying the C++ state of the object in memory — the rest is automatic and transparent.

Concurrent access to clusters within a data store is supported, making it ideally suited for supporting NEO multithreaded server programs. Automatic atomic updates provide a two-phase commit protocol, in anticipation of distributed transactions involving multiple services. Either all the changes made within an update are recorded, or none of the changes are recorded. A server process can be atomically updating several clusters with different schema definitions concurrently.

Based on the OMG CORBAservices Persistent Object Service specification, the Data Store Manager is a upwards-compatible subset of the object database specification developed by the Object Database Management Group (ODMG). An object implementation can therefore be easily upgraded as application requirements evolve.

Concurrent Requests

The NEO Network ORB spawns a new thread for each incoming request to a server process. NEO Concurrent Requests provide transparent management of concurrent requests to multithreaded object implementations, ensuring the integrity of shared data. This frees developers from the implementation details of multithreaded programming, allowing them to focus on the application, while gaining the significant performance advantages and scalability of Solaris' advanced multithreading technology.

Three different locking policies are supported: *mutex* (default), *reader-writer*, and *fine grain*. The locking policy is selectable on a per object implementation basis. Scoped locks are supported: with mutex and reader-writer locking, locks are automatically released when out of scope of the locking variable, or when the variable is destroyed.

- *Mutex Locking*

  Only one request at a time is allowed for each object (i.e., only one method in the object implementation is active at any time): a single thread gains exclusive access to an object's persistent state. Other objects (instances as well as implementations) in a server process can be servicing requests simultaneously.

- *Reader-writer Locking*

  Operations that only read shared data (persistent and transient) are distinguished from those that (potentially) may write or change data. Each operation in an object's interface is defined as either reader or writer. This locking policy enables multiple concurrent reader operations but writer operations gain exclusive use.

- *"Fine grain" Locking*

  Higher level locking can be turned off and developers can implement custom fine grain locking policies. Shared data can be protected at the thread mutex level within a method.

Implementation Support

The NEO Implementation Support provides functions that simplify and make easier the development of object implementations. This includes:

- *Servant Support*: constructor, destructors, locking, and deactivation

- *Smart Object References*: automatic memory deallocation for smart object references when they go out of scope or upon assignment
- *Reference Data Manipulation*: simplified manipulation and management of reference data associated with object references
- *Exception Handling and Message Text*: including an internationalized message text catalog, extensible by developers
- *Object Tracing*: standardized, automated way of tracing and logging; controlled on a per-server process basis
- *Message Logging*: predefined logging macros (in addition to trace messages) and runtime configurable control of message source, message destination, and output format on a per-server process basis
- *Utility Support*: ease-of-use support for custom persistence, subobjects, and other NEO Network and NEO Services features

Server Management

Individual custom applications can readily capitalize on the advanced built-in administration and management capabilities of Solaris NEO. The NEO Server Management provides server and application management functions and transparent support for management objects that are installed as part of an application. Management objects are automatically generated by the NEOworks Networked Object Constructor and incorporated into the server program along with application objects.

Server Management completely takes care of server management and application management functions that a server program developer would otherwise need to provide. It enables the status of computers, server processes, and objects to be interactively monitored, administered, and managed by Solstice NEO tools.

With NEO Services, server processes can be activated and shutdown manually, the current status of a server process can be queried, object tracing and message logging can be turned on or off, and the persistent state (transparent and custom persistence) of objects in the server process can be backed-up and restored without any programming by the application developer.

Application Installation

NEO Application Installation provides transparent support for the application installation process. Application installation code is automatically generated by the NEOworks Networked Object Constructor and accessed and controlled by Solstice NEO tools. Application Installation completely takes care of installation functions that the application developer would otherwise need to provide.

Installation steps that are automated include the registration of server programs with the NEO Network ORB, the installation of IDL information associated with server programs in the NEO Interface Repository, and the registration of shared services for access via the NEO Shared Service Finder.

The NEO Application Installation builds on the easy-to-use SVR4 packaging concept, providing a familiar and reliable method for installing and deinstalling software packages. Applications and objects are installed and their availability automatically registered, allowing seamless upgrading with minimal impact on users.

Solstice NEO 7

Solstice NEO extends Sun's strategy of using the network to manage the network itself by using objects to manage objects. Solstice NEO adds capabilities for the administration and management of networked applications and shared services to the Solstice product family. With Solstice NEO, resources can be monitored and controlled from any point on the network, improving responsiveness and reducing costs. System, workgroup, shared service, and application management capabilities are supported.

A full set of administration and management tools provide for one-step system and application installation, full backup and restore, and extensive on-line help. Both command line and graphical user interface tools are provided to facilitate both an automated, scripted method, or a graphical, interactive browsing approach. Computers can be upgraded incrementally so that the entire network need not be affected at once. Implementations can be evolved in a way that allows multiple versions to coexist in the same server program, and multiple server programs to coexist as separate processes.

Solstice NEO provides a comprehensive set of functions through a suite of three tools: *NEOadmin*, *ServerViewer*, and *NameViewer*. These tools are briefly described below, and the functionality they provide is described in the sections that follow.

Advanced Administration and Management Tools

neoadmin

The *neoadmin* utility is an interactive command line tool implemented as a Tool Command Language (Tcl) shell. Tcl is similar to the Bourne shell, but more complete. Scripts may be written to automate system administration functions. Extensive on-line help is included in *neoadmin.*

ServerViewer

The ServerViewer is an intuitive and powerful graphical tool with a user interface style similar to the NEXTSTEP file browser. ServerViewer provides centralized monitoring and control functions. Views may be filtered to isolate a particular computer or workgroup.

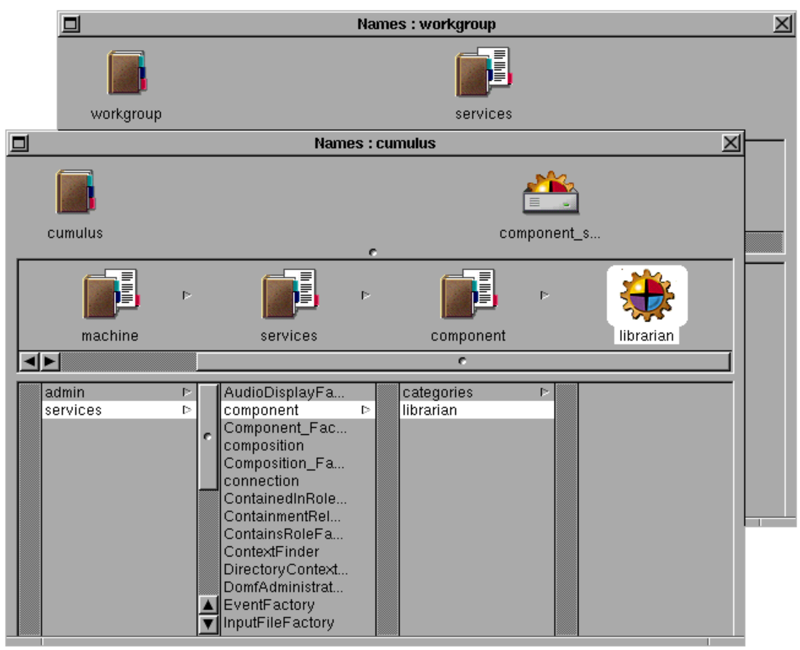

Figure 6 ServerViewer performs basic system management tasks such as controlling and displaying the status of object servers and processes, controlling tracing and logging facilities, and administering workgroups.

NameViewer

NameViewer is a graphical browser that allows the interactive traversal of hierarchical naming contexts. Naming contexts may also created, removed, and their contents listed.

System Installation and Management

System Installation

Solstice NEO enables one-step installation of Solaris NEO. When Solaris NEO is installed on a computer that computer automatically becomes accessible from other computers. Server programs are installed using SVR4 installation packages. Clients require no special installation. Solaris NEO can be installed on a computer using NFS mounted files as well as from a CD-ROM.

Interface Repository Administration

Interface information can be interactively traversed and displayed. Interfaces can also be shown individually or to any specified depth.

Naming Service Administration

Hierarchical naming contexts (including predefined workgroup naming contexts) can be interactively traversed and listed in browser or list formats. Naming contexts can also be created or removed.

Message Logging Control

Message logging can be toggled on and off per server process. Message source, destination (i.e., file, console, syslog, or custom handler), and output format can be changed and log files can be viewed.

Workgroup and Shared Service Administration

Workgroup Administration

Workgroup Administration enables workgroups to be created and maintained. This includes adding and removing a computer from a workgroup, listing the names of computers in a workgroup, and querying their status.

Workgroup Data Backup and Restore

Backup of all the data associated with a workgroup is supported, and may be carried out while the system is running. To ensure a consistent snapshot, a hold is put on running server processes (preventing further request processing), the data is saved, and the server processes are then released. Data includes server process data, object state (transparent and custom persistence), internal NEO Network ORB data (BOA and Interface Repository databases), Server Configuration Files, and the NEO Network ORB control file.

Backups are run as a background process on each computer. A two phase process handles problems arising from the distributed nature of the data and ensures the successful completion of the backup or restore. The status of backups can be queried and backups can be removed (i.e., archived).

Shared Service Administration

Shared Service Administration allows services to be registered, exported into a workgroup for sharing, and deregistered. The available services in a workgroup or on computer can be listed. A service can be removed from a specific computer or from each computer in a workgroup, and can be relocated by re-registering it in another workgroup or on a different computer.

Application Installation and Management

Application Installation

SVR4 installation packages are automatically generated by NEOworks Networked Object Constructor. The generated package contains all the required headers, libraries, stubs, skeletons, IDL files, Interface Repository files, server programs, client programs, exception message catalogs, user started server programs, and shell scripts.

These packages are installed on a computer using the Solaris visual *Software Manager* tool. Application upgrading and deinstallation are also supported.

Automated install-time steps include the registration of server programs with the NEO Network ORB, the installation of IDL information associated with server programs into the NEO Interface Repository, and the registration of shared services for access via the NEO Shared Service Finder. Optional custom scripts are also supported to "populate" naming contexts with required objects. These scripts are run once an application is installed and the associated server programs registered.

Server Monitoring and Management

The names of servers can be listed and their status queried. This includes what is installed, verification that server programs are correctly registered, what is currently running, and how many objects are activated.

Servers processes can be put on hold (e.g., for upgrades), released, shutdown, and restarted (e.g., to restart a hung server). The Server Configuration File can also be displayed which contains trace, logging, and server shutdown information.

Object Trace Control

Object tracing can be selectively toggled on and off per interface (e.g., method entry and exit). Trace output destination can also be changed.

References

The Common Object Request Broker: Architecture and Specification, Object Management Group, 1994.

NEO Programming Guide, Beta Version, SunSoft, Inc., May 1995.

NEO Systems Management Guide, Beta Version, SunSoft, Inc., May 1995.

NEO Tutorial, Beta Version, SunSoft, Inc., May 1995.

NEO System Installation, Beta Version, SunSoft, Inc., May 1995.

NEO Programming Interfaces Reference, Beta Version, SunSoft, Inc., May 1995.

Solaris NEO Product Family, SunSoft, Inc., May 1995.

Workshop NEO Development Environment, Product Overview, SunSoft, Inc., January 1996.

Solaris OpenWindows: OpenWindows V3 Collection: Release Reports and White Papers, Part Number 91021-0, SunSoft Inc.

Solaris SunOS 5.0: SunOS 5.0 Multithreading and Real-Time, Part Number 91025-0, SunSoft Inc.

Solaris ONC: Design and Implementation of Transport-Independent RPC, Part Number 91028-0, SunSoft Inc.

Solaris SunOS: SunOS 5.0 Release Report, Part Number 91023-0, SunSoft Inc.

The ToolTalk Service, Part Number 91022-002, SunSoft Inc.

Introduction to the ToolTalk Service, Part Number 91031-002, SunSoft Inc.

Project DOE: Distributed Objects Everywhere, Part Number 91035-0, SunSoft Inc.

The ToolTalk Service: An Inter-Operability Solution, Part Number ISBN 013-088717-X. SunSoft Press/Prentice Hall, Englewood Cliffs, NJ.

ToolTalk and Open Protocols: Inter-Application Communication, Part Number ISBN 013-031055-7, SunSoft Press/Prentice Hall, Englewood Cliffs, NJ (June 1993).

An Overview of the Spring System, James G. Mitchell, et al, SunSoft, Inc., 1994