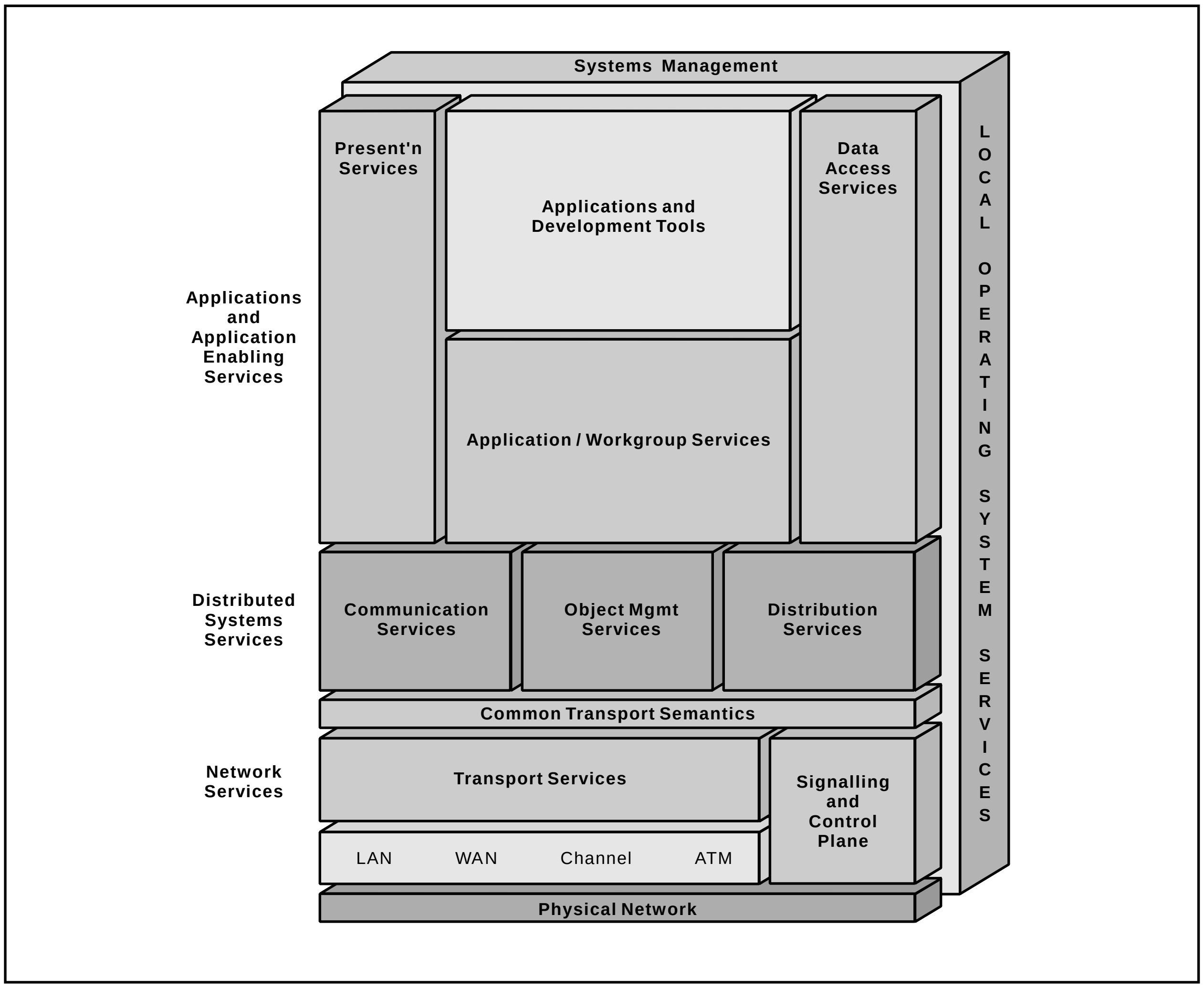
Open Blueprint

Application Development in the Open Blueprint



Systems Management
Present'n Services
Applications and Development Tools
Data Access Services
Applications and Application Enabling Services
Application / Workgroup Services
LOCAL OPERATING SYSTEM SERVICES
Distributed Systems Services
Communication Services
Object Mgmt Services
Distribution Services
Common Transport Semantics
Network Services
Transport Services
Signalling and Control Plane
LAN
WAN
Channel
ATM
Physical Network

GC23-3909-01

Open Blueprint

IBM

Application Development in the Open Blueprint

GC23-3909-01

About This Paper

Open, distributed computing of all forms, including client/server and network computing, is the model that is driving the rapid evolution of information technology today. The Open Blueprint structure is IBM's industry-leading architectural framework for distributed computing in a multivendor, heterogeneous environment. This paper describes the Application Development in the Open Blueprint component of the Open Blueprint and its relationships with other Open Blueprint components.

The Open Blueprint structure continues to accommodate advances in technology and incorporate emerging standards and protocols as information technology needs and capabilities evolve. For example, the structure now incorporates digital library, object-oriented and mobile technologies, and support for internet-enabled applications. Thus, this document is a snapshot at a particular point in time. The Open Blueprint structure will continue to evolve as new technologies emerge.

This paper is one in a series of papers available in the *Open Blueprint Technical Reference Library* collection, SBOF-8702 (hardcopy) or SK2T-2478 (CD-ROM). The intent of this technical library is to provide detailed information about each Open Blueprint component. The authors of these papers are the developers and designers directly responsible for the components, so you might observe differences in style, scope, and format between this paper and others.

Readers who are less familiar with a particular component can refer to the referenced materials to gain basic background knowledge not included in the papers. For a general technical overview of the Open Blueprint, see the *Open Blueprint Technical Overview*, GC23-3808.

Who Should Read This Paper

This paper is intended for audiences requiring technical detail about application development in the Open Blueprint. These include:

- Customers who are planning technology or architecture investments
- Software vendors who are developing products to interoperate with other products that support the Open Blueprint
- Consultants and service providers who offer integration services to customers

Contents

Application Development in the Open Blueprint

Introduction

The Open Blueprint structure defines an application as the use of information technology to assist in the execution of a business process. Distributed, network applications execute in or exploit multiple systems to accomplish their functions.

The Open Blueprint structure provides the reference model for services which enable applications to use system components and infrastructure across a distributed, network environment. The Open Blueprint resource managers provide common run time services and facilities which deliver consistent and interoperable functions across an application's network topology. The richness of these resource managers' functions allows the developer to easily incorporate network capability into the application. Application development tools enable the exploitation of these facilities to deliver integrated, interoperable business solutions. During development, tools create application elements that access Open Blueprint resource managers through calls on the various resource manager Application Programming Interfaces (APIs). Since the APIs are based on standards, the application can be delivered into many different computing environments and can be protected from change in the underlying run time services and facilities. These services and facilities are available through multiple programming facilities and languages, both procedural and object-oriented.

This paper focuses on the delivery, through application development tools, of business solutions which use the Open Blueprint interfaces and protocols in their run time environment. It also addresses future capabilities of distributed tools enabled by the Open Blueprint structure and describes current IBM application development tools and tool suites which enable distributed development.

First, however, it is important to understand the challenges facing businesses deploying application solutions in distributed environments.

Distributed Application Needs

The demands for distributed, client/server, network applications are driven by global markets, advanced communications, rapidly changing business topologies, and highly-competitive industries. For example, the distributed, network needs of a hospital which must serve customers (patients) through complete services include not only access to information and processes under the control of the hospital but also the ability to rapidly access information from suppliers (like pharmacies) and external servicing organizations (like insurance companies) and to supply information to physicians and patients.

The applications which support this organization must be:

- Deployed and managed across a distributed business environment
- Able to rapidly access information across an inter-enterprise business network now extended across the Internet
- Flexible and changeable

These needs can be demonstrated by looking at a representative user execution environment. This might consist of a network including:

- Host transaction environments (such as OS/390, CICS, and DB2)
- Middle tier servers (such as UNIX and OS/2)
- Workstations (such as OS/2 and Windows)
- Network computers

For this environment, the software inventory might include multiple operating systems, various relational databases, several transaction monitors, and multiple transport network protocols. This business environment demands that applications operate in a "many-to-many" topology. From the development perspective, the expectation is that various tools can be used to mix and match capabilities in delivering applications which can share appropriate data and logic.

In using Open Blueprint services, applications in this environment can call on various resource managers through APIs and/or higher-level interfaces. A typical application might include a Graphical User Interface (GUI) to the user through a Web browser or workstation application, use of APIs for local and/or remote processing, and access to data on a remote mainframe or server. Applications, particularly mainframe applications, often execute under the auspices of a transaction monitor, like CICS. During execution, the transaction monitor maintains control of the applications and invokes the appropriate resource managers such as security, database, and communications services for the application. The applications deploy actions to process exceptions and handle errors. And all the actions may be executing in a cross enterprise environment created on the Internet. Not all of the applications in this environment require complex processing, however. In any business, there are also simple client/server applications such as remote data requests. Through its open and accessible services, the Open Blueprint structure provides the structure, interfaces, and protocols to enable execution of multiple distributed, network application models across heterogeneous environments.

Distributed Application Models

There are several ways to view and categorize distribution models. The major difference among these models is the function that enables the interactions between application elements. From an application perspective, it is useful to look at where these functions execute (locally or remotely) and how complex they are. Distributed applications range from simple queries to robust, complex transactions involved in doing business electronically on the Internet. The ideal run time environment for applications must support the broad range of application models across differing and changing topologies. The Open Blueprint structure defines such an environment. Its services support these models by providing access to functions and data with location transparency.

Client/Server: Developers creating classic, two tier client/server models gain significant benefits from easy-to-use graphical user interfaces (GUI) and access to remote data. There are several methods of implementation to address these needs through the Open Blueprint Presentation and Data Access Services.

GUI Presentation Services. *Remote presentation* applications are built using interfaces such as the High Level Language Application Programming Interface (HLLAPI) which enables a workstation element to access host applications by emulating the host user interface. *Distributed presentation* applications are built using interfaces such as Motif on the X-Window System. GUI presentation is further enabled by abstract APIs which allow the application to be generated and deployed for execution in multiple environments without changes to the application's definition. As an example, a graphical user interface generated with IBM OpenClass Library: User Interface Classes can deploy over either OS/2 PM or OSF/Motif interfaces transparently.

The rise of the Internet and the World Wide Web has produced a new model for distributed presentation using the Web browser as the universal, “thin” client. Using the Hypertext Markup Language (HTML) as the development base, user interfaces are created that can be deployed on a broad range of Web browsers, such as IBM Web Explorer, Netscape Navigator, or Microsoft Internet Explorer on platforms from network computing appliances to high function engineering workstations.

Query. Query applications, which generally focus on retrieving remote or distributed data, are enabled through the Data Access Services interfaces of the Open Blueprint structure. Specifically, Database and File resource managers are called through APIs which execute distributed data requests and return the requested data to the application at the end user's client workstation for subsequent processing by local logic.

Distributed Data Access. An extension to remote data read (query) capabilities is full data update capability from a client application. Distributed and remote data applications are enabled using interfaces such as SQL Call Level Interface (CLI) which allows data on a server to be accessed as though it were local. Thus, client applications can provide a high level of location transparency to the end user for data access and manipulation.

Distributed Logic: As network infrastructures grow within users' environments, business logic elements become candidates for distribution. The Open Blueprint Relational Database resource manager provides a simple form of distributed logic capability through stored procedures. The Open Blueprint Communication Services allow the application to split its logic through four procedural mechanisms:

- Conversations
- Remote Procedure Call
- Messaging and Queuing
- Hypertext Transfer Protocol (HTTP)

The Open Blueprint Object Request Broker (ORB) enables distributed business logic within object-oriented applications. The Open Blueprint Distribution Services provide functions including directory, security, time, and transaction management that are integrated with the Communication Services and the ORB and that shield the application from many of the complexities of the distributed, networked environment.

Stored Procedures. Stored procedures, residing within the database management system, provide the ability to execute predefined business logic attached to remote data. Stored procedures are accessed through a call to the Relational Database resource manager.

Conversations. Conversational applications operate in a synchronous model; both parts of the application must be active. The parts engage in a two-way dialog, and each part maintains state information about the progress of the conversation. Conversational applications are implemented through calls to the Open Blueprint Conversational resource manager which supports the Common Programming Interface for Communications (CPI-C) API.

Remote Procedure Call. Applications requiring access to distributed logic running outside the local environment can invoke distributed applications through a familiar procedure call model. The Open Blueprint Remote Procedure Call (RPC) resource manager enables the remote execution of the distributed logic for the calling application. Like the conversational model, RPC is a synchronous model; both distributed parts must be active. The calling program is responsible for maintaining execution state.

Messaging and Queuing. This model is used for applications where assured, once-only delivery of a message between application parts is required. It has the characteristic that delivery does not require availability of an immediate connection between the sender and the receiver. This asynchronous model is enabled through the Message Queue Interface (MQI) of the Open Blueprint Messaging and Queuing resource manager. The Messaging and Queuing resource manager ensures message delivery is completed even after a disconnection. This is extremely important to applications that support mobile users and other applications where any part of the application may not always be online or available. Another example might be where an end user needs to request information about a product but can do other work until that information comes back.

Hypertext Transfer Protocol (HTTP). This model is used for intranet and Internet applications to provide message-based interaction in a point-to-point, and possibly multi-threaded, environment. Uniform Resource Locators (URLs) provide the message envelope addressing; Hypertext Markup Language (HTML) provides a lingua franca for message content. The basic model is stateless, but the newer versions of HTTP attach state information to the URL envelope.

Distributed Business Objects Object technologies allow applications to be designed, constructed, assembled, and deployed using components. By reusing pre-built components (the software equivalent of integrated circuits) supporting discrete business and technical functions, applications may be quickly assembled or modified to support business information system needs.**:** The distribution of business objects provides applications total flexibility in deployment. The Open Blueprint structure includes Object Management Group (OMG)[1] Common Object Request Broker Architecture (CORBA)-compliant services for object management. Implemented through the Open Blueprint Object Request Broker (ORB), these services enable applications to deploy objects across the network topology with maximum flexibility. This support allows the application developer to focus on putting objects where they belong in the business model while still maintaining consistency across all applications using the same function.

The Open Blueprint structure also includes a Virtual Machine resource manager which supports distributed applications written in Java. These Java applications can interoperate with non-Java objects through the OMG CORBA Internet Inter-ORB Protocol (IIOP).

Distributed Solutions: The most complex solutions are those which allow both distributed data and distributed logic across multiple locations. The Open Blueprint enables this complex environment with interfaces which can provide transparency and interoperability across various topologies. This model allows users to deploy function where needed and then allows movement of that function for purposes of workload balancing, ownership, and performance.

Collaborative Applications: Collaboration occurs when two or more people participate in a business task that relies on the use of common, shared information. The last few years have seen a rapid expansion in the both the scope of collaboration and the number of people using collaborative technology. One factor accelerating this growth is the phenomenon of the Internet. The Internet is increasingly becoming the platform for collaborative activities between enterprises and other external organizations outside the enterprise intranet.

Key to basic collaboration is sharing information that is contained in collaboration document stores. Using basic document services, ad hoc and collaborative business solutions can be built quickly and deployed easily. The application enabling services available for building collaborative solutions are described in the *Open Blueprint Collaboration Resource Manager* component description paper.

Workflow Applications: The distributed applications built from these models just described may be combined into larger business solutions using workflow functions. Workflow concepts and functions are described in the *Open Blueprint Workflow Resource Manager* component description paper.

[1] The Open Blueprint structure bases its approach to distributed object computing on specifications from the Object Management Group (OMG). Founded in 1989 as a non-profit organization, the OMG is now comprised of over 600 of the most prominent companies in the information industry including IBM, Apple, AT&T, NCR, Hewlett-Packard, Digital, Borland, and Sun. The mission of the OMG is to develop a single architecture and set of specifications (based on commercially available object technology) to enable distributed application integration while assuring reusability, portability, and interoperability of components.

Building Applications for Distribution

While the specific decision about which distribution model the application requires depends on the business problem being solved, selection of the tools and/or tool suites for building that implementation is usually a technical decision. Certainly, some tools and tool suites have more natural capabilities for certain models. However, applications for almost any distribution model can be built with several tools and tool suites. In selecting development tools, current tools and current skills of the developers often weigh heavily in the decision.

Building applications for maximum distribution flexibility requires independence from topology and execution infrastructures. The various alternatives provided by today's tools enforce varying degrees of structure on the developer. These development tools are generally centered on the language environment—3GL, 4GL, OO, Scripting—providing various levels of abstraction to the developer. Higher levels of abstraction tend to be more business-centered, that is, they provide more definition in terms of business objects and allow the developer to focus more on solving the business problem as opposed to working with the internals of the specific technical implementation.

Third-Generation Language Development. Procedural 3GL environments allow developers to build application elements using direct access to various resource manager APIs. For example, a 3GL-generated transaction processing application executes presentation, data access, and transaction monitor services through APIs to Open Blueprint resource managers. Applications written with 3GLs can execute Communication Services explicitly, such as RPCs to partner application elements. The Open Blueprint structure provides additional transparency through the Common Transport Semantics which allows the application program to execute across multiple transport network protocols. The languages themselves, such as COBOL, FORTRAN, PL/I, and C/C++, adhere to X/Open and ANSI/ISO language and library standards to enable portability across operating environments.

Because 3GLs provide maximum control for specific service execution, they also require maximum knowledge of interfaces and protocols by the developer. When used in combination with object-oriented class structures, their flexibility can be channeled to use a predefined set of capabilities. The class libraries provide "wrappered" Communication Services or Data Access Services, such as CPI-C, RPC, or SQL CLI calls, which make the specification of the API easier for the programmer. With these class structures and visual building aids, the knowledge requirement is lessened so that more of the development effort can be focused on solving the business problem.

Fourth-Generation Language Development. 4GLs, or generators, by definition provide and often enforce a higher level of abstraction by providing consistent interfaces across multiple infrastructure implementations. Building the same transaction processing application with a 4GL generally means that interface invocation parameters are built with prompts or templates. Resource manager service invocation can be defined by the developer using simpler terms and statements, often with the specific implementation defined at generation time rather than during logic development. The generated code adheres to the 3GL language standards to provide portability across multiple environments. The Open Blueprint structure provides consistent interfaces such as Communication Services which can be used by 4GLs for portability and interoperability across multiple topologies.

Object-Oriented Language Development. Object-oriented development can move the focus to a business object perspective. Through OO development, objects representing business entities and processes are defined. These business objects are then decomposed into classes, frameworks and parts, depending on the specific level of abstraction, for use in building applications. These business objects, when constructed to reflect the way the business is run, are natural candidates for distribution across the business network. When built with standard object-oriented languages such as C++, Smalltalk, and Java, the business objects are portable and allow users to exploit reusability of common business functions. The use of the Open Blueprint Virtual Machine resource manager supporting the execution of Java objects extends portability from the source code to the executable level. Flexibility of business object distribution is facilitated by the Open Blueprint Object Management Services and distribution infrastructure. The Open Blueprint Object Request Broker provides OMG CORBA-compliant services which allow interoperability of objects across distributed implementations. Thus, an application can exist which exploits business objects anywhere in the distributed environment.

Scripting Languages. Scripting languages are specialized languages used to provide the "glue" to combine other applications, parts, and objects. They may also be used to develop applications. Generally, scripting languages are easy to use, interpretive, and can be implemented at any level of language environment. When combined with the parts capabilities of the Open Blueprint Compound Document resource manager, they provide significant benefit in bringing business objects together at the desktop and provide novel ways to deliver mobile computing. Scripting is ideal for developing software agents which can be distributed through the network to act on behalf of users or other computing resources. Scripting accesses Open Blueprint services through resource manager APIs such as the relational database API, workflow API, and human computer interaction APIs to easily assemble and deliver combinations of applications, or business objects, across an enterprise's application environment.

In language-based development environments such as the four just described, there are no clear demarcations which force complete dependence on any specific level of abstraction. Within any one, the developer can choose to write at a lower level. However, the direction for application development tools is to provide less and less focus on the lower level technology and more on productivity improvements which sharpen the focus on solving the business problem. With visual building interfaces, assistants, and pre-built components, applications can easily exploit existing and shared functions and objects. The Open Blueprint structure enhances this ability by providing the infrastructure of interfaces and protocols for portability and interoperability across heterogeneous environments.

Tool Requirements for Building Distributed Applications

Successful exploitation of the distributed environment happens when the applications for that environment are built using facilities specifically focused on distributed capabilities. Accurate implementation of the distributed business model is most successful when the translation to implementation is technically easy and natural.

For these reasons, the decision on the type of application distribution model must accurately reflect the required capabilities of the business application solution. Once this is determined, the desired development environment for building the solution helps determine what tools are required.

Tools which support distributed environments are inherently different from stand-alone or single system-centric tools. They must be able to build and test complete distributed applications. In other words, they must provide functions which facilitate the complete development activity—model, design, simulate and test—across multiple platforms. They must be capable of integrating existing functions over new platforms, provide access to existing data, separate the development environment from the deployment environment, and provide mechanisms for validation and verification of distributed applications.

Complex, mission critical applications are still largely custom designs. With increasing frequency, the design process used to develop many applications is one of iteration and successive prototype development, instead of the one-pass, waterfall approach. In this approach, selected elements of the application are successively prototyped. Each prototype is more functionally complete than the previous one. Tools that support the iterative prototyping process include visual builders for client interfaces in IBM's VisualAge and the rapid development and test capabilities in IBM's Visual Generator, which allow both client and server logic to be exercised in the development environment. This iterative cycle is important not only during development but also on a continual basis throughout the life cycle of the application. It provides a means for assuring continued optimization within the deployment environment.

Additional development paradigms, appropriate to object manipulation and usage between objects, are emerging. Frameworks, or collections of classes, are designed to work together to generally define an architecture for a part of an application. These frameworks provide the base classes which allow a developer to create and implement subclasses to complete the function needed for specific application implementations. Frameworks provide an overall structure for applications, while allowing for flexibility in how the applications can be distributed.

To make the transition to distributed object technologies, the Open Blueprint structure supports a new generation of application development tools using a new paradigm which involves the separation of the roles of developer and assembler. The tools are defined to be “language neutral” and support multiple programming languages like C++, Smalltalk, and Java. To be effective for large-scale projects, they also provide team-based programming and support for component libraries. To offer dramatically improved levels of productivity over traditional programming approaches, they support “visual building” for the graphical construction of applications by combining icons representing components. The combination of these capabilities, team-based, cross-language, and visual building, produces an integrated development environment for distributed object computing. IBM's VisualAge tool family is an example of this type of integrated development environment. The value of such tool sets lies in the development consistency they offer to the enterprise within a very wide range of tool and language choices.

In the era of network computing and the World Wide Web, applications are under increasing pressure to support deployment around the globe. For an application to meet the specific needs and preferences of a variety of users in different countries, it needs to be internationalized at build time for general worldwide use, and it needs to be localized for a particular language, culture, and character data encoding at run time. This allows any user, at any location, to use that application and have it provide culturally correct processing results. Internationalization is discussed in depth in the *Open Blueprint Internationalization Resource Manager* component description paper.

Tool Support and Directions

The structure of the Open Blueprint increases the ability to provide tools which support these requirements. Tools can provide the application developer with the capability of building business applications that can easily exploit any of the services provided by Open Blueprint resource managers. And, using Open Blueprint distributed services, development tools can provide the ability to transparently test distributed applications across multiple platforms. For example, tests that execute a test suite on a remote server can be run from the developer's workstation. The developer performs all test and debug activities at the workstation, sees the application run as if it were local, and is able to dynamically test (and fix, if needed) the code executing at the server. These facilities are provided by the tool itself exploiting the Communications Services of the Open Blueprint.

Developing a distributed application is for the most part the same as developing a non-distributed, or stand-alone, application. In the development of a stand-alone application, the developer has to contend with interfaces to the user, interfaces to system resources, use of data, and implementing the business logic.

Developing a distributed application requires that the application be divided into interacting elements that may execute on different systems. Each element can be developed as a non-distributed application, that is, with consideration for user interface, system resource interface, data, and business logic as appropriate. The new consideration added as a result of the application's being distributed is the interaction between the application elements.

Approaching the Development of Distributed Applications. It is the understanding of these interactions between the distributed elements that provides the challenge in developing distributed applications. In placing elements, the frequency, type, and volume of data communications between elements must be considered. These factors largely determine the performance of the distributed application. Visual tools will provide assistance in determining the distribution points within the application and with analysis of chosen distribution points from a performance perspective. In cases where a new application element is being attached to an existing element, the distributed design of the application may be defined by the split between old and new. Examples of this are new clients being added to existing transactions. For performance analysis, the existing application can be measured and those values used in place of estimated values.

Designing a Distributed Application. Without the facilities provided by Open Blueprint resource managers, designing and building distributed logic applications would be complex. Fortunately, the challenges of locating elements, managing security, and providing transport independence are all assisted by Open Blueprint Distributed Systems Services and Network Services. Partitioning the application into elements remains a challenge for the developer. Tools will assist in constructing the application as elements. The approach to design can start based on an existing application domain analysis or design which is then partitioned and placed on a network topology. It can also start from an existing network topology and existing application elements which are refined into a design for the new elements. Tools also allow for easy definition of interactions between distributed elements in a way that is independent of the actual services being used. The resulting distributed application model is shared among multiple tools.

Once the distributed application elements have been identified, they can be designed using traditional techniques and tools. These tools can include object domain analysis and design tools that lead to generation of source code.

When the application partitioning details have been designed, tools will analyze the application based on how elements are used, determine bottlenecks in the application's distributed structure, and make suggestions on how to place application elements in the target distributed environment.

Implementing a Distributed Application. After the design of the application is completed and analysis indicates that the bottlenecks have been eliminated, an application skeleton can be generated from the distributed application model. The skeleton will include the code required to initialize and initiate the element interactions and whatever additional business and presentation logic that can be generated. The strength of the application skeleton is that the element interaction logic can be generated for the application developer, not by the application developer.

The generated code skeleton is stored in a version and change management system. Code stored in the version control system is accessible at the developer's workbench. To complete the application, the developer completes the code using whatever editor or builder is desired. The developer typically completes business logic, exception handling, and access to services not specified in the design tool.

Application assembly uses the build facility. The build facility allows different versions of the application to be maintained and creates executables on multiple platforms. It uses location information from the distributed application model to build the application for the target execution platform(s). As required by the compilers, source will be moved to appropriate platforms for compile and link. Generated files will be moved back to the library. This includes files required for the browsers and other analysis tools.

Evaluating a Distributed Application. Once the application is completed, function and performance correctness must be verified. The first tool used is the debugger. Today's debugger allows for debug of stand-alone applications running on the same system as the debugger. For some platforms, applications can be debugged remotely using workstation-based tools. The next step is to allow debug of a distributed application. The distributed debugger uses a workstation debug tool to cooperatively track execution of the application on the client and on the server(s). This includes the ability to transfer control to distributed elements.

The developer will also need to trace the interaction between distributed application elements. These interactions, or events, are captured as a result of executing the application and recording the events as they occur. Tools analyze the records and visually show the interactions. This visualization allows the developer to see application interactions and detect points of failure and unexpected behavior.

Existing tools, already a part of the VisualAge for C++ family, for example, allow analysis of individual distributed elements when executed and analyzed as stand-alone elements.

Deploying a Distributed Application. Once the application is complete and verified to be correct and performing reasonably, the application must be deployed to production systems. This involves getting executables to the target systems, ensuring that target system resources are tailored properly, and tailoring the application to match the target system. To facilitate deployment to a broad set of target systems, the tools will generate scripts that will drive the tailoring of the resources at the target system. These scripts will be generated to match the specific application needs.

IBM Tools for Distributed Development

IBM provides a rich set of tool suites for developing distributed applications. These suites contain tools that are themselves distributed applications which use the Open Blueprint resource managers during development time. Since programming is typically done by teams working cooperatively, developers share common suite facilities which provide the ability for checkin and checkout of programs, data, documentation, and metadata including private developer views of that information, and merge/promote capabilities for team sharing. The major tool suites are:

3GL Tool Suites. IBM COBOL, PL/I, and C/C++ provide complete environments for developing distributed applications. Each of these development tool suites provides user interface building capabilities for the developer at the workstation and compatibility across multiple platforms. Various levels of transparent development are supported through visual assistants for access to common functions, such as relational database and communications service invocation. Visual RPG provides a visual programming environment for distributed applications in an AS/400 run time environment. It uses the distribution support of the Open Blueprint Communications Services to provide seamless distributed application development.

4GL Tool Suite. VisualAge Generator provides an iterative generator environment, including GUI building from pre-built components and a fourth-generation language. Team Connection includes both extended dictionary and library facilities for sharing data models across multiple tools and software configuration facilities that support versioning and build for applications. Team Connection DataAtlas offers data modeling and physical database design capabilities.

Object-Oriented Tools. VisualAge for C++ provides object-oriented application development capabilities through a visual interface with a consistent C/C++ language across a broad number of deployment platforms. It includes a rich set of class libraries for common functions and services. IBM's Smalltalk-based tools, including VisualAge Smalltalk, provide an interpretive, object-oriented programming environment. VisualAge for Java couples the development support strengths of VisualAge with the portability inherent with the Java language. The VisualAge for Teams programming environment is built on IBM Smalltalk and provides object-oriented parts construction and the ability to assemble those parts into applications through a GUI. This assembly is accomplished by connecting icons on the palettes with other icons and by specifying parameters for connections and logic in a visual manner. balance ":CIT" line 505

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AS/400
CICS
DB2
IBM
IBMLink
Open Blueprint
OS/2
VisualAge

The following terms are trademarks or service marks of other companies:

Apple	Apple Computer, Incorporated (name and logo)
AT&T	American Telephone and Telegraph Company (sm)
Borland	Borland International, Incorporated
C++	American Telephone and Telegraph Company, Incorporated
CORBA	Object Management Group, Incorporated
Digital	Digital Equipment Corporation
Hewlett-Packard	Hewlett-Packard Company
Java	Sun Microsystems, Incorporated
Microsoft	Microsoft Corporation
Motif	Open Software Foundation, Incorporated
Netscape	Netscape Communications Corporation (logo)
NCR	NCR Corporation
OSF/Motif	Open Software Foundation, Incorporated
Sun	Sun Microsystems, Incorporated
X/Open	X/Open Company limited

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Communicating Your Comments to IBM

If you especially like or dislike anything about this paper, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply. Feel free to comment on specific error or omissions, accuracy, organization, subject matter, or completeness of this paper.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

- If you prefer to send comments by FAX, use this number:

  United States and Canada: 1-800-227-5088.

- If you prefer to send comments electronically, use one of these ID's:
  - Internet: **USIB2HPD@VNET.IBM.COM**
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink: **CIBMORCF at RALVM13**

Make sure to include the following in your note:

- Title of this paper
- Page number or topic to which your comment applies

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC23-3909-01