Service Design and Inventory System—An Object-Oriented Reusable Software Asset

Joseph O. Bergholm
J. Michael Davis
Behzad Nadji
Peter D. Ting

Competition and the fast pace of technological evolution in today's global telecommunications industry are placing unique demands on the flexibility of operations support software systems. The industry must be capable of rapidly introducing new services, technologies, and organizational structures. Networks must be capable of being partitioned for various applications and administered based on complex ownership relationships among the various network components. User permissions must be readily adaptable to reflect various combinations of services, network partitions, and work functions. To meet the time constraints of the market, telecommunications providers require the ability to configure systems to meet their needs without relying on traditional software development intervals and external software development resources. Traditional software development methodologies generally do not provide the timeliness and flexibility required. The *Service Design and Inventory (SDI)* system—also known as the *Attribute Design Database System (ADDS)*—has achieved a high degree of reusability and customer-configurable adaptability through a unique application of object-oriented technology.

Introduction

The *Service Design and Inventory (SDI)* system is an exceptionally adaptable object-oriented software asset that has achieved the integration of a wide range of functionality and a high degree of software reuse across diverse customer applications. SDI supports several application functions within the network management layer of the telecommunications management network (TMN) architecture (see Figure 1). Within the network management layer, SDI acts as the database of record for all network-related information, including the following:

- *Network locational information*, including geographic coordinates of structures containing network equipment (customer offices, access serving offices, central offices, and outside plant locations), as well as physical locations of equipment within an office;
- *Physical network components*, including equipment and transmission media (for example, fiber, coaxial, radio, and microwave), equipment port terminations, and cabling connections within and between equipment; and
- *Network path connections*, including high-line and lower levels of transport facilities and service-providing circuits.

Much of the application functionality within the network design and network inventory management modules of SDI revolves around the management of this highly interrelated network information, including tools supporting network office and bay installation, installation and cabling of equipment and facilities, and circuit design and assembly.

In addition to these network management functions, SDI also serves as the appli-

cation interface to the service management and element management layers of the TMN. The SDI order management module controls the acceptance and processing of requests for changes in the network originating from planning organizations, as well as customer service organizations. After a request has been processed in the network management layer (for example, a circuit has been designed and assembled), SDI will then communicate the design information to the element management layer to support physical implementation of the design against the appropriate physical network elements. The SDI gateway module supports these types of interfaces to upstream and downstream applications, as well as interfaces to applications in the network management layer for portions of the network that may not be controlled and inventoried via the SDI application.

The final SDI module depicted in Figure 1 is the rule management module. This module allows an SDI customer to tune the application of SDI to support individual processes, services, and technologies. SDI includes the following customizable aspects:

- *Service descriptions*, supporting routing, design, and assembly rules for facilities and circuits providing different types of service, as well as acceptable options and settings;
- *Processing control schedules and user permissions*, allowing a customer to outline the specific control steps and user permissions for various portions of the design and assembly process, as well as permissions management for other modules of SDI;
- *Equipment profiles*, allowing templates of different equipment configurations to be modeled to support automated equipment configuration, as well as validation processes;
- *Subnetwork definition*, allowing various portions of the network to be subdivided based on technology, network hierarchy, organization, or other category to allow separate definition and control of the target subnetworks; and
- *Feature (field definition) control*, allowing customization of labels, values, help, and error messages to support multiple language delivery.

Panel 1. Abbreviations, Acronyms, and Terms

ADDS—Attribute Design Database System, an alternate name for SDI
CCP—the Customer Care Platform of AT&T Business Communications Services
ITU—International Telecommunications Union
OOA—object-oriented approach
PDH—plesiochronous digital hierarchy
SDH—synchronous digital hierarchy
SDI—Service Design and Inventory system
SONET—synchronous optical network
STM 1—synchronous transfer mode level 1, the STM transport facility with a rate of 155 Mb/s
TMN—telecommunications management network
UIM/X*—User Interface Management System for X Windows software

SDI Development Methodologies

The following subsections detail the four development methodologies of the SDI system: object-oriented analysis, separation of application-specific and core domains, data gateway, and subnetworks.

Object-oriented Analysis Methodology. To support initial analysis and requirements definition activities, members of a team having a wide range of experience with existing network provisioning approaches were assigned to the SDI project. The team members based their initial system specification on an internally generated set of scenarios having the following constraints:

- The specific application domain would be fixed within the network management layer of the TMN.
- The application would handle design and assemble functions for all network hierarchical layers and service types that might potentially be targeted for support.
- The application would handle all existing equipment technologies that might potentially be targeted for support.
- The application would take into account the complex competitive environment in the domain of telecommunications. Finding an entire network that is owned and operated by one telecommunications company or organization is becoming increasingly rare. Thus, the application would have to handle the sharing of inventory use between multiple network management applications.

This open-ended set of constraints forced a dis-

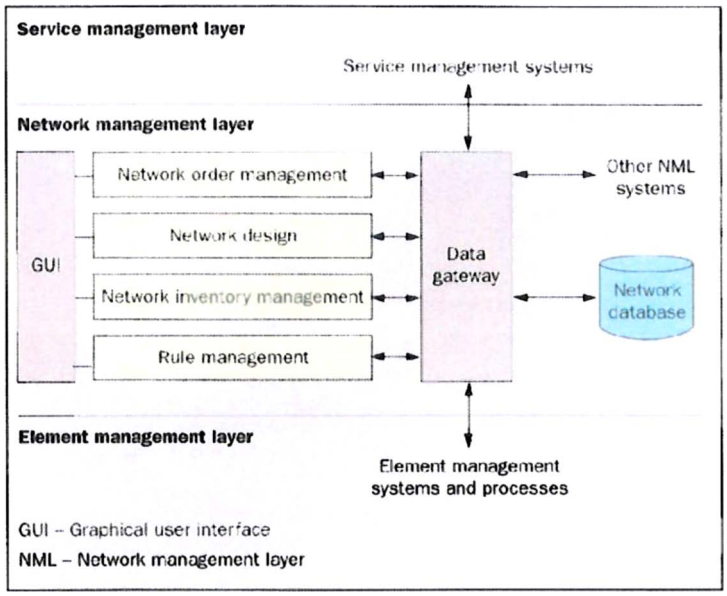

Figure 1. The SDI system is an exceptionally adaptable object-oriented software asset that has achieved the integration of a wide range of functionality and a high degree of software reuse across diverse customer applications. SDI supports several application functions within the network management layer of the telecommunications management network architecture, as shown in the drawing. Within the network management layer, SDI acts as the database of record for all network-related information.

tinctly different approach from that employed for traditional one-off software projects in which development is often based on detailed requirements defined by a single end-user. To maximize software reuse between potential application scenarios, the management of the SDI project adopted the following two design strategies:

- Recognize the similarities between scenarios supporting application functions and engineer around them.
- Separate the similarities found (core aspects) from the application-specific aspects in terms of both data and functionality.

To support this strategy for dissecting the application domain, the team decided to take an object-oriented approach (OOA) to the analysis. In the initial stages of the OOA, a set of primary object classes is typically defined. Then, an inheritance hierarchy is developed based on "typing" information that allows categorization of the class. Initially, typing is rather arbitrary and becomes a sort of classification exercise. True typing does not emerge until scenarios are found that actually result in different attributes and behaviors between different object types.

In the SDI project, the more that was learned about the primary target object classes (link, node, equipment), the fewer significant differences emerged between the class subtypes. Over the course of the analysis, the class hierarchy became flatter and flatter, or less and less specialized. The team slowly discovered that within the target application domain (network management), differences in object behavior resulted in different relationships between objects rather than fundamental differences in the objects themselves. In other words, one type of object (such as a particular link type) would be defined primarily in terms of the types of objects to which it could be related (node types, other link types). The primary determinants of these relationships were found to be the *values* of the attributes rather than new or different *sets* of attributes.

To illustrate this point with the link example, a path or aggregate link can be composed of a number of component links. The allowable types of component links that can constitute this relationship are based on various attribute values on both the aggregate and compositional links (some examples include bandwidth, mileage, restoration priority, transmission media, frame format, and line format). A DS0 circuit, for example, can only be composed of DS1 time slots or other component links meeting the attribute requirements of the DS0 aggregate

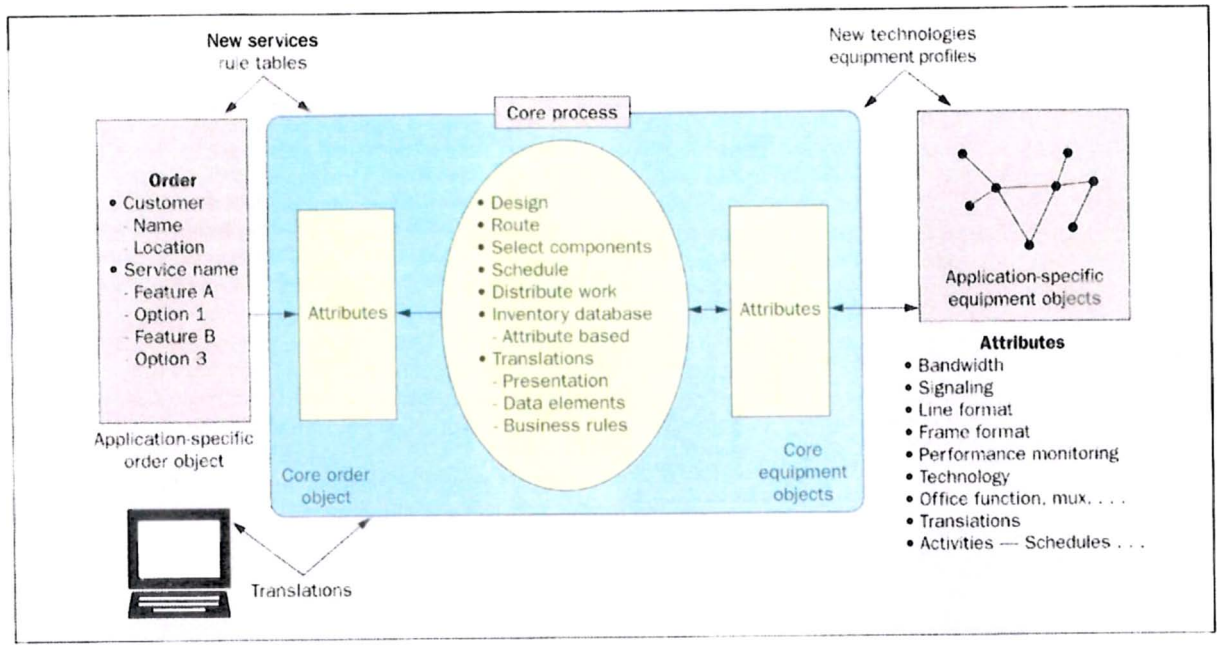

Figure 2. Core objects using application-specific rule data make SDI an adaptable system. The drawing illustrates the separation of application-specific rules from core operations. Using circuit provisioning as an example, the provisioning process can be depicted as being driven by a customer order for a particular type of service, which is defined by various attributes describing that particular type.

link. Although these links differ in the values of their attributes, they all contain the same basic attribute set.

Separation of Domains. Based on the results of the preliminary analysis, it was determined that virtually all the basic network management application functionality could be generalized to a set of primary superordinate object classes containing generalized network management operations without the need to develop specialized classes (see Figure 2). Specialization *was* needed, however, to support the definition of rules for defining which types of objects could be associated with other types of objects in the network. Of course, depending on the network, these rules could differ from application to application based on differences in operations processes, services, and equipment technologies.

To meet the goal of being able to handle these different application scenarios flexibly, the team had to develop a way to allow the flexible definition of these differences into the application. Toward this end, the team began to develop sets of interrelated rules as data structures in the database that would support object typing (attribute value) information needed to drive the operations supporting the core object classes. Whenever a core operation is triggered, it accesses the appropriate typing information to determine the constraints for establishing relationships to other network objects.

Figure 2 illustrates this separation of application-specific rules from core operations. Using circuit provisioning as an example, the provisioning process can be depicted as being driven by a customer order (request) for a particular type of service (type of circuit or link through the network), which is defined by various attributes describing that particular type.

Generally, the service type (or link type definition) is unique to a specific application in both format and jargon. Similarly, each telecommunications provider's

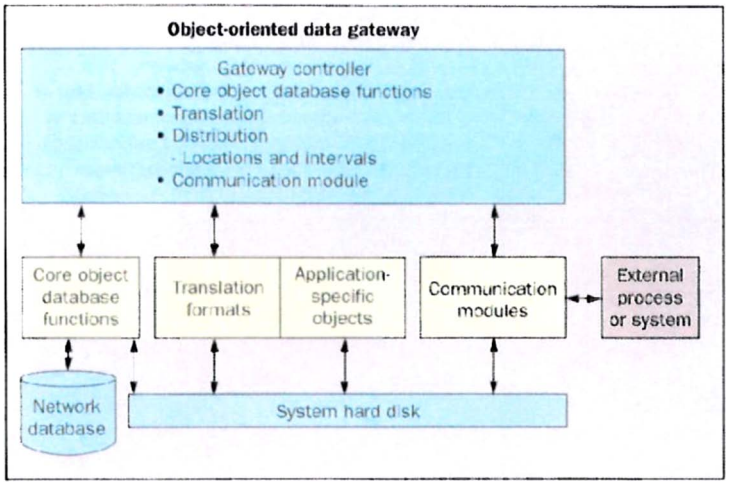

Figure 3. In the SDI data gateway architecture, core data objects, such as service orders, are translated into the formats of application-specific objects either required or supplied by external processes or systems. These external objects are either imported from or exported to the external systems using the appropriate communication module. The gateway supports both multiple translation formats and communication modules.

network is unique in terms of the specific combination of vendor's equipment and design strategies. Thus, the link or service types and network element types can be viewed as application-specific object type definition rules.

Despite the variations in local type definitions, however, all services (as requested by orders) can be viewed in terms of their requirements on the fundamental attributes of a transmission network. These attributes, as illustrated in Figure 2, include such properties as bandwidth, signaling, line format, and technology. In SDI, therefore, the application-specific specialization or typing information is captured separately. Then, it is *instantiated* to core objects prior to processing. Thus, specialization is achieved not through the development of specialized object classes but via the instantiation of specialized attribute values against core, generalized super classes.

The link object class is a good example of using modeling generalization and attributes in the formulation of object classes and objects. Links represent physical or logical connections between two network termination points or equipment ports. They can have inventories (channels) of available capacity. They can have component links at the same level of the network hierarchy (aggregate links) or at different hierarchy levels (which provides a mapping between levels). Finally, links have such attributes as bandwidth, line formats, technology, restoration, and ownership.

By specifying the appropriate attributes under the control of the SDI rules, link objects can be instantiated representing, for example, any level of the bandwidth hierarchy, any channelization scheme (such as ITU, North American, PDH, SONET, SDH), and any technology (such as fiber, radio, satellite). By appropriately specifying the attributes in the SDI rules, therefore, link objects can be constructed using the same object class code to support the design of a wide range of service types.

This is a radical departure from many traditional system environments. Not only is code not reused, but multiple one-off systems are developed to deal with individual service types, bandwidth levels, or technologies.

SDI applies the same approach to other such generalized object classes as equipment, nodes, and orders. This insures a high degree of code reuse between applications. Furthermore, the rule tables can be defined by users without software development, illustrating the flexibility of the SDI software assets to accommodate changes in a user's environment. It also emphasizes the high level of control the SDI rule application management process places in the hands of a user.

Data Gateway. The SDI data gateway architecture is another example of how the system separates core from application-specific functionality. The gateway structure is designed to provide a flexible methodology for SDI to interface the SDI core capabilities effectively with other application-specific systems and processes. The gateway separates those parts of the code from the

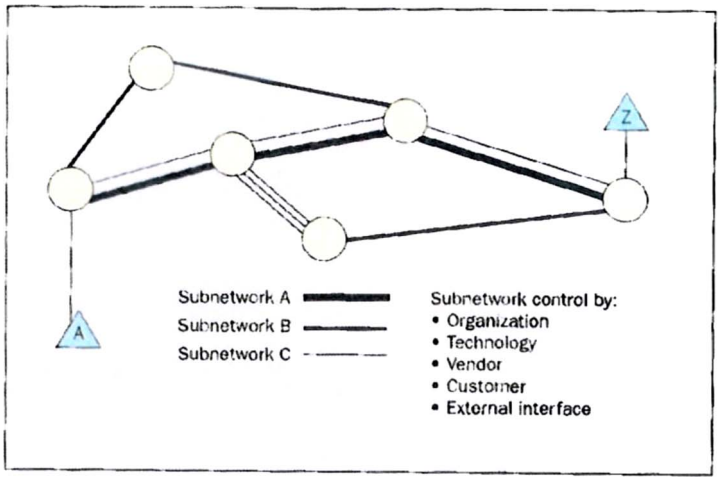

Figure 4. Subnetworks are defined at the link and port levels, and subnetwork attributes are key to management of the network. This drawing illustrates three subnetworks, two of which could represent the responsibilities of organizational regions. The third subnetwork represents a customer's facility between locations A and Z.

core code that may have to change from interface to interface. Such separation helps ensure that the changes can be made in the least complicated way and with the least possible impact on the core portion of the code.

In the SDI gateway as shown in Figure 3, core data objects, such as service orders, are translated into the formats of application-specific objects either required or supplied by external processes or systems. These external objects are either imported from or exported to the external systems using the appropriate communication module. The gateway supports both multiple translation formats and communication modules. This can be as simple as formatting a work order and sending it to a printer or as complex as a full Q3 interface with standards-managed object messages. The gateway controller is a table-driven process that, for a given transaction type, will choose the appropriate SDI core object, translation format, and communication module, as well as the appropriate distribution addresses and frequency (immediate or batch at a given time).

Subnetworks. The SDI network model was designed to allow a network to be divided into multiple subnetworks. This allowed meeting the objective requiring the ability to partition the network to support complex ownership interdependencies in the network management layer. Subnetworks can represent such factors as the network responsibilities of various organizations, the portions of the network leased from various providers, portions of the network dedicated to the use of a particular internal or external customer, or the portions of the network controlled by various network element controllers. Subnetworks can be defined at a granularity as specific and as low as port or link using tools in the inventory management module. Portions of the network can be "members" of more than one subnetwork.

The subnetwork attribute can be used as a key controlling element in many of the rule tables. When used with the user group and activity tables, the subnetwork attribute can determine, for example, which designer can view (or view and modify) different portions of the network. When the subnetwork is used with the gateway distribution table, it can cause the gateway to send commands to the appropriate network element controller or to send requests for more leased capacity to the appropriate telecommunications provider. With the activity description table, the subnetwork attribute determines such activities as which user groups will be assigned.

Figure 4 illustrates three subnetworks, two of which could represent the responsibilities of organizational regions. The third subnetwork represents a customer's facility between locations A and Z.

Figure 5 illustrates the application of subnetworks to identify a portion of the network obtained from an access provider between nodes A and D and the establishment of subnetworks at specific network hierarchy levels. For example, the STM 1 SDH network between

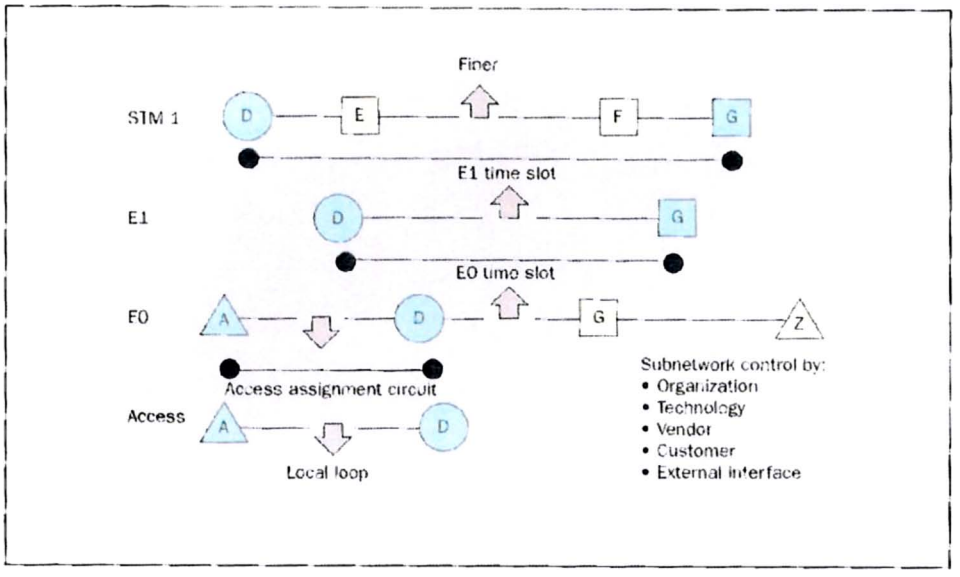

Figure 5. Subnetworks are applied to various levels of the network hierarchy. This illustration shows the application of subnetworks to identify a portion of the network obtained from an access provider between nodes A and D and the establishment of subnetworks at specific network hierarchy levels. The STM 1 SDH network between locations D and G may be the responsibility of a specific internal organization.

locations D and G may be the responsibility of a specific internal organization.

The use of the subnetwork attribute in the network definition and in the rule tables illustrates another capability that SDI places in the hands of a user to adapt the SDI software assets to changes in the business environment.

SDI Software and Hardware Architecture

SDI employs a logically three-tiered client/server architecture as depicted in Figure 6. Tier 1 normally consists of a workstation containing the user interface, some functional modules, and the database interface layers. Tier 2 consists of a local file server for the Tier-1 workstations and printers in the same cluster. The database and its server functions are located on a single Tier-3 server, also known as the *central server*.

Depending on the customer and the customer's geographical distribution of offices, two or three of the tiers can physically collapse into one. Thus, as shown in Figure 6, SDI can range from fully residing on a single workstation to a two-tiered client/server model on a local area network to a three-tiered architecture on a wide area network. This architecture is fully aligned with the AT&T Business Communications Services (AT&T-BCS) Customer Care Platform or CCP, which is currently globally deployed.

The software architecture of SDI, as shown in Figure 7, was designed using multiple layers. Where appropriate, an object-oriented approach was designed within layers to ensure ease of maintenance and porting, as well as data encapsulation.

The physical user-interface layer provides direct graphical interface to a user. It separates graphical user interface toolkit and platform dependence from the rest of the application. This strategy proved useful as SDI was ported from the OPEN LOOK* to the Motif* platform, only this layer had to be translated. This means that the switch to Motif* was made without affecting code other than the code at the physical UI layer. A third-party tool, the User Interface Management System for X Windows software (UIM/X*), is used to design the physical layer. The user-interface library layer, which functions as a set of C++ class objects and methods, is an object-oriented implementation toolkit (an independent window presentation). The control logic that drives the physical window layer through user-interface library objects resides in the abstract user-interface layer. Operations that manage the

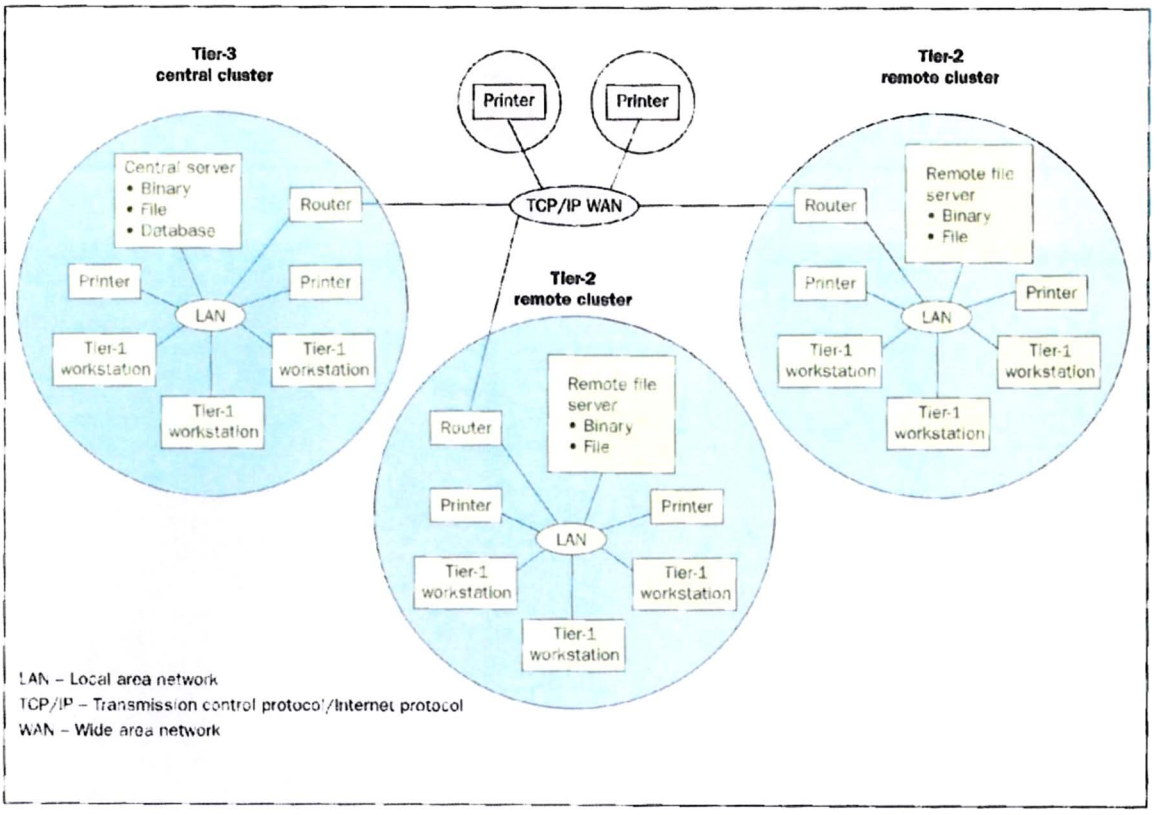

Figure 6. SDI employs a logically three-tiered client/server architecture as depicted in the drawing. Tier 1 normally consists of a workstation containing the user interface, some functional modules, and the database interface layers. Tier 2 consists of a local file server for the Tier-1 workstations and printers in the same cluster. The database and its server functions are located on a single Tier-3 server, also known as the central server.

physical window (such as zoom in/out and enable/disable buttons) are implemented in this layer. Finally, application-layer objects requiring visual presentation are grouped in the presentation layer.

The software architecture that maps the object-based modules into the relational database is illustrated in Figure 8. This drawing also depicts key characteristics of the SDI software architecture, as well as aspects of the modeling and development tools employed. The functional object-oriented modules[1], programmed in C++, interface to the ORACLE7* relational database management system through an object-oriented database interface layer, which is code generated using the specification-driven code generator MetaTool™. This layer provides the functional modules' independence from the database.

SDI object modeling drove the definition of the table structures in the relational database. Using Rumbaugh's techniques[1], the tables closely resemble the objects themselves. Denormalizing some tables in the database leads to improved performance for the database calls. Tuning the database for performance has been an important project activity.

To further improve performance and to separate the functional code from the details of the database realization, stored programs within the ORACLE system have also been employed. This is particularly important in light of the scaleable SDI hardware architecture depicted in Figure 6 in which the stored procedures reduce the amount of data that must be transmitted on the wide area networks between the central server and the clients.

SDI uses several third-party tools and libraries in its design. AT&T standard components libraries are a key part of SDI design. In addition, the use of AT&T BaseWorX™ system components, third-party map and table widgets, as well as hypertext have helped reduce the time required for development of the SDI system.

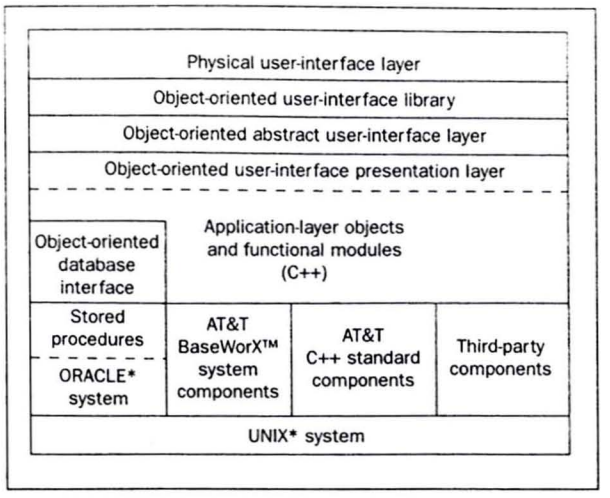

Figure 7. The software architecture of SDI, as shown in the chart, was designed using multiple layers. Where appropriate, an object-oriented approach was designed within layers to ensure ease of maintenance and porting, as well as data encapsulation.

SDI Deployment Activities

The deployment of the SDI system is based on a particular software release and includes the following three major activities:

- Characterization of the user's environment to obtain the information necessary to populate the appropriate tables and rules, ensuring that the core components will meet environmental needs;
- Assistance with the initial database population; and
- Identification and development of the necessary application-specific components.

With the rule-based structure of the SDI system, the capabilities required by a customer are partially realized both by the functionality of the code and by the populated rules. System testing of the code releases, therefore, only partially ensures that the customer's requirements will be realized.

Thus, a dual-cycle testing strategy has evolved from the SDI project. This strategy consists of through-system code testing followed by testing of the rules within the environment of the tested code. The rule testing generally takes the form of a regression of a key subset of the system tests combined with scenario testing based on specific customer application processes. After the entire testing cycle is completed, the system is deployed having an initial set of customer rules.

Rule management tools are available to enable an application administrator to modify the tables and rules to accommodate most changes in their environment, such as new services, technologies, and organizational responsibilities. To ensure the integrity of the rule data, however, the administrator must establish a test data environment for basic scenario testing before the introduction of new data structures into the operational environment.

Specific Deployment Activities. Currently, SDI has been deployed to four customers, including those in both the Spanish and English environments. The first four applications represent a wide variety of network configurations, service types, organizational structures, and work flows. In addition, the following basic technologies addressed in the customer networks varied widely:

- Transport capacity design and leased line circuit design for a network based on E5 fiber and E4 radio technologies;
- Trunk networks between customer PBXs, access switches, and backbone switches having leased transport at the E1, E3, or E4 bandwidth levels; and
- Transport capacity design involving an SDH-based backbone network initially supporting a switched trunk network but evolving to a broad set of voice and data services.

In addition to these fundamental differences in deployed technologies, the operations processes and user permissions management have also differed widely between the initial SDI customers. Despite the diverse nature of these applications, SDI has achieved virtually 100-percent software reuse across the deployments, discounting the software enhancements between releases for a product early in its development cycle. These achievements are a direct result of the design objectives adopted at the beginning of the project.

Conclusion

The definition of generalized, attribute-based object classes in SDI enables the same code to be applied to a wide range of user scenarios. To operate, these generalized object classes are specialized via the instantiation of attributes retrieved from user-populated rule tables.

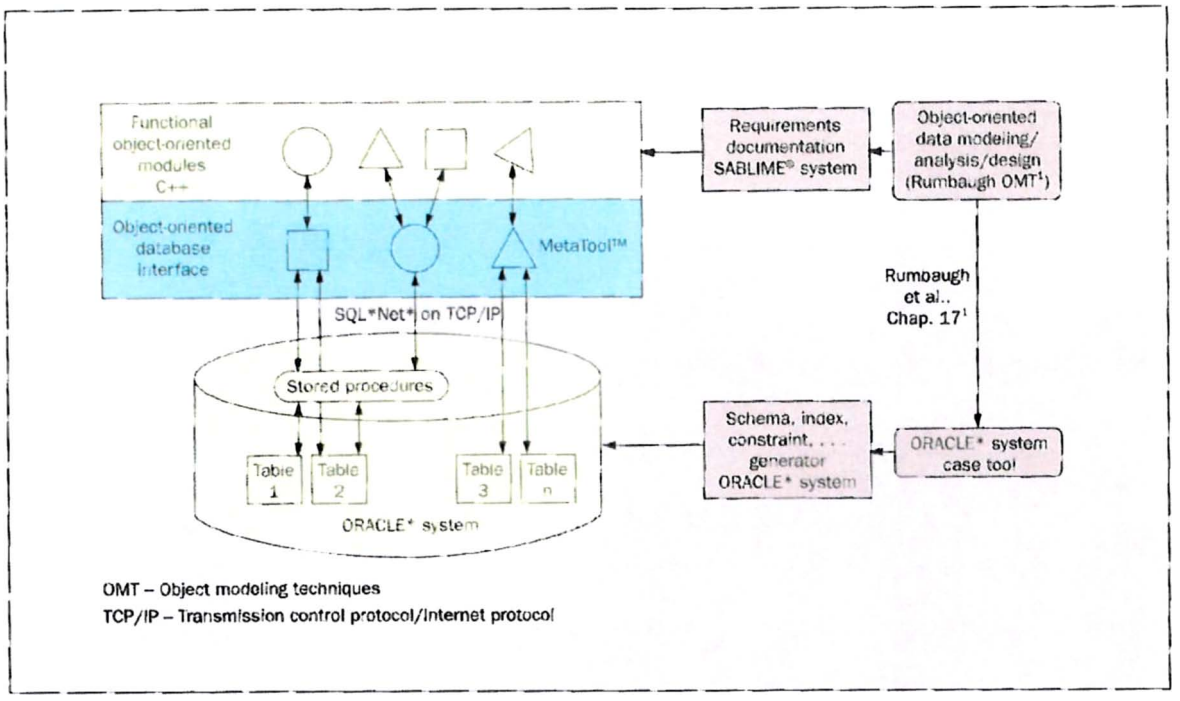

Figure 8. This drawing shows the mapping of objects in SDI to relational database tables. It also depicts key characteristics of the SDI software architecture, as well as aspects of the modeling and development tools employed. The functional object-oriented modules, programmed in C++, interface to the ORACLE7* relational database management system through an object-oriented database interface layer, which is code generated using the specification-driven code generator MetaTool™. This layer provides the functional modules' independence from the database.

Thus, not only is the same code reused, but the operation of the system can be configured by a user without additional code development. This arrangement reduces the time and expenditure necessary to customize SDI to meet application requirements, thereby reducing the time and cost required to introduce new technologies, services, and user-defined processes and permissions.

The adaptability of the object-oriented attribute and rule-based software has many user advantages in today's competitive and rapidly evolving telecommunications environment. The maintenance and generation of quality rules, however, is a responsibility that requires a thorough understanding of the user operational environment and SDI rule structures, as well as the discipline to test and verify the operation of the rules as the environmental needs evolve.

Acknowledgments

Many people have contributed to SDI. The leadership and vision of S. Y. Lee resulted in the initiation and support of this effort. Support from AT&T-BCS globalization fund providers Ed Sable and Don Wormley was instrumental in SDI's evolution in recent years. Prem Mehrotra, Debra Williams, Mohammed Mehio, Hai-Chen Tu, Joe Russell, Steve Lessard, Cheryl Histand, and Rod Pease were involved since SDI's initial design and architecture development. Currently, a team of nearly 50 dedicated people from AT&T-BCS and AT&T Network Systems are actively contributing to SDI engineering, development, testing, customer relations, and support.

***Trademarks**

Motif is a registered trademark of Hewlett-Packard Inc.

OPEN LOOK is a registered trademark of Sun Microsystems Inc.

ORACLE, ORACLE7, and SQL*Net are registered trademarks of ORACLE Corp.

UIM/X is a registered trademark of Visual Edge Software, Ltd.

Reference

1. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling And Design*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

(Manuscript approved February 1996)

'oseph O. Bergholm *is a technical manager in the Globalization, Provisioning, and Technology Planning and Systems Department at AT&T Business Communications Systems in Middletown, New Jersey. Mr. Bergholm is responsible for planning, systems engineering, and 'ustomer deployment management in support of the Service)esign and Inventory System (SDI). He was graduated from he University of Pennsylvania in Philadelphia with B.S. and 1.S. degrees in electrical engineering, as well as a Ph.D. in ;ystems engineering and operations research. Mr. Bergholm, ı recipient of the AT&T Eagle Award for SDI system technology ınd a patent holder in digital transmission technology, joined he company in 1966.*

. Michael Davis *is a district manager in the Globalization, Provisioning, and Technology Planning and Systems Department at AT&T Business Communications Services in Middletown, New Jersey. He manages and supervises all Service Design and Inventory System (SDI) engineering rocesses. Mr. Davis, who joined AT&T in 1988, has a !.S. degree from Miami University in Oxford, Ohio. He also as an M.S. degree in cognitive science (artificial intelligence) 'om the University of Cincinnati in Ohio, where he is also a octoral candidate.*

Behzad Nadji *is a technical manager in the Globalization, Provisioning, and Technology Planning and Systems Department at AT&T Business Communications Services in Middletown, New Jersey. He is responsible for the development of network inventory and provisioning systems. He is currently working on the Service Design and Inventory (SDI) system, as well as other system development projects. Mr. Nadji has an M.S. in electrical engineering from the Electronics University of Tehran in Iran. He also has M.S., E.E.E. (engineer's), and Ph.D. degrees in electrical engineering from the University of Southern California in Los Angeles. He joined AT&T in 1988.*

Peter D. Ting *is a technical manager in the Provisioning Systems Development Department at AT&T Business Communications Services in Middletown, New Jersey. He manages an organization responsible for designing and developing software systems that automate telecommunications service provisioning processes. He has a B.S. degree from the National Taiwan University in Taipei, an M.S. from Brown University in Providence, Rhode Island, and a Ph.D. from the University of California at Berkeley, all in physics. Mr. Ting, an AT&T Bell Laboratories Fellow, joined the company in 1973.*