# ♦ Implementing a Management System Architecture Framework

*William C. Goers and Michael R. Brenner*

*Any practical vision for the evolution of communications services must include a strategy for how networking vendors make it possible for service providers to manage their networks. While the Telecommunications Management Network (TMN) framework has proponents, the IP services community has shown little interest. Furthermore, operations systems developers have long attempted to produce the best framework, but the technology is outdated before it exists. This paper addresses both issues by presenting an application-driven model for integrated management. This model can be applied to either a "classic" framework orientation or a management application view. What is common between these two views are a management portal, common data models, multiple interface technologies, open and simple network element interfaces, and common operations, administration, and administration (OA&M) tools. These are the elements for which there needs to be a consistent set of interface definitions. They form the basis for the construction of next-generation management applications.*

## Introduction

Any practical vision for the evolution of communications services must include a strategy for how networking software and equipment vendors make it possible for service providers to manage those services and the networking equipment upon which they are delivered. Unfortunately, the fact that there is no one, consistent service and network management strategy (there are many mutually inconsistent strategies) across the communications industry makes establishing such a vision very difficult. This is one of the key issues facing both service providers and vendors of management system applications (including element and network management systems, services and engineering support systems, and other business systems), and it is a major barrier to fulfilling the business promise of next-generation networks.

This paper serves as a companion to the previous paper by Silverman, Brenner, and Shannon,[1] which proposes a strategy for network and service management and describes an architectural framework. This paper expands upon the architectural framework described in the previous paper by looking at the framework from two complementary points of view. It also delves deeper into implementation considerations and makes recommendations based upon the authors' view of industry experience and currently available development technologies.

## Needs and Drivers

As Silverman, Brenner, and Shannon state, "[The] needs [of today's network and service providers] are … business obvious— business and operational support systems that:

- *Encourage* new service creation,
- Are much easier to operate and maintain, and
- Require less integration expertise to insert into an operator's systems environment.

The rush of new network builds, insertion of new elements, and quick scaling of new Internet services has passed the point of most providers to have a working

**Panel 1. Abbreviations, Acronyms, and Terms**

ASP—application service provider
BSS—business support system
CLI—common line interface
CMIP—common management information protocol
COPS—common open policy service
CORBA*—Common Object Request Broker Architecture
CPU—central processing unit
DBMS—database management system
DIAMETER—extension of RADIUS
EAI—enterprise application integration
EJB*—Enterprise JavaBeans
FAB—fulfillment, assurance, billing
FCAPS—fault, configuration, accounting, performance, and security
GUI—graphical user interface
HTML—HyperText Markup Language
i—"integrated, intelligent, and innovative"
IDL—interface definition language
IETF—Internet Engineering Task Force
IIOP*—Internet Inter-Orb Protocol
IP—Internet protocol
ITU-T—International Telecommunication Union, Telecommunication Standardization Sector
J2EE*—Java* Platform 2, Enterprise Edition
JDBC*—Java* Database Connectivity
LAN—local area network
LDAP—lightweight directory access protocol
NE—network element
OA&M—operations, administration, and maintenance
ODBG—open database connectivity
ODSI—optical domain service interconnect
OEM—original equipment manufacturer
OSF—operation system function
OSS—operations support system
Q3—the TMN interface between an OSF and NE function/mediation function/QAF
QAF—Q adapter function
QoS—quality of service
RADIUS—remote authentication dial-in user service
SLA—service-level agreement
SNMP—simple network management protocol
TCP—transmission control protocol
TFTP—trivial file transfer protocol
TL1—Transaction Language 1
TMF—TeleManagement Forum
TMN—Telecommunications Management Network
TOM—Telecom Operations Map
XML—Extensible Markup Language

strategy for the underlying service and network management systems."[1]

Service providers are driven to constantly develop new services and deliver them to the marketplace. These services drive a need for innovation in network elements, features, and interfaces. They also drive a need for innovation in service and network management systems—not only because of the proliferation of network elements that expose new behavior, but also because of the proliferation of logical (overlay) networks and the services delivered over those virtual networks. Add to that the fact that these service providers must support their traditional services (which are supported by an embedded base of support systems) at the same time, and it is easy to see that management systems applications developers face major challenges.

From an implementation perspective, these high-level needs statements can be translated into essentially three basic drivers:

- *The characteristics*—Promote simplicity, flexibility, openness, and evolution.
- *The blueprint*—Establish a common architectural strategy and development approach that facilitates end-to-end integration of multivendor solutions and embedded systems.
- *The key components and tools*—Provide mechanisms for quickly filling gaps in service and network management functionality while avoiding inconsistent or duplicated efforts.

The venerable Telecommunications Management Network (TMN) model, based on ITU-T Recommendation M.3010,[2] still commands respect within the telecommunications sector. The many incumbent network service providers seek support for this model. However, the IP services industry and community of users have little interest in or respect for it. This is reflected in their approach to management standards,

which can be summarized as: include as much management intelligence as possible in the network elements; provide some external hooks to tap into that intelligence; and otherwise, for the most part, ignore the fact that services and networks need to be managed.

The TeleManagement Forum (TMF), relatively new on the scene, has been very active in the last few years in providing process-flow-oriented specifications using the Fulfillment, Assurance, Billing (FAB) and Telecom Operations Map (TOM)[3] models. Those specifications are more in accord with current industry trends. They attempt to "soften" the rigidity of the TMN model by explaining the functions needed and the relationship between those functions, but stopping short of endorsing protocols or layers. This leaves the door open to new software vendors that try to project a strategic vision around their own product set, crossing, bypassing, or otherwise ignoring traditional model boundaries and conventions and with little definition to the landscape. They seem to think their exaggerated claims will not be scrutinized. However, not all service providers are accepting those claims—and the result is an uneasy skepticism as these service providers seek value propositions in clearly defined areas.

What one finds in the market is therefore several trends, instead of a unifying strategy. One such trend is toward increasing intelligence and adaptability of management applications. Another trend embraces the need to integrate seemingly separate applications in response to business and operational requirements.

## The Network as an Ecosystem of Applications

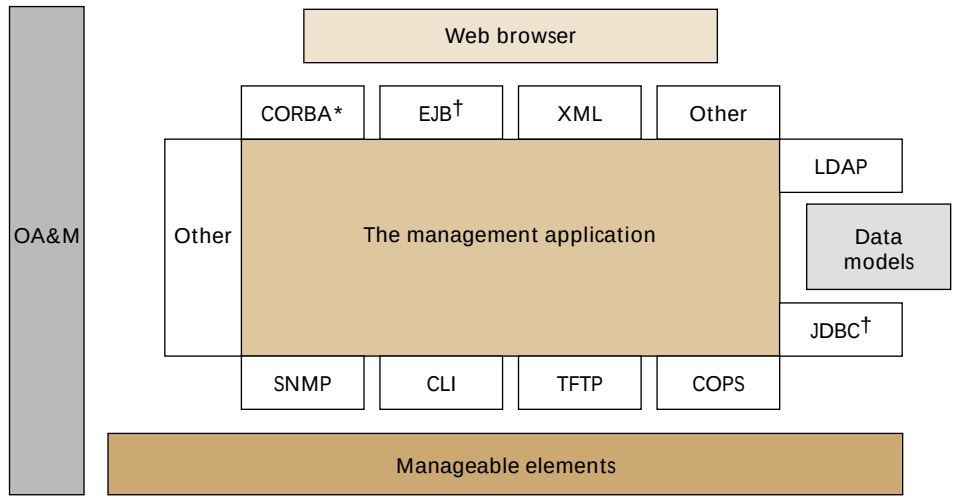
Sometime in the 1980s, Sun coined the phrase "The network is the computer," and it served the company well. In the 1990's, one of the database vendors (realizing the marketing leverage Sun had obtained from its slogan), came up with "The network is the database," and it served some of the database management system (DBMS) vendors well. Do the claims boasted by Sun and the database vendors still apply? They surely do. Not only do computer-based "hosts" generate the majority of the traffic on networks today, but most network elements are now based on computer technology (CPUs, interfaces, memory, storage). It is also true that the network would add much less business value without database servers, which maintain the "state" of the network, its users, and services.

However, it is time to recognize a new paradigm. The reality is that networks without management applications are little more than pre-programmed controllers, and that falls far short of what users expect today. It is also true that the networks customers are talking about today are mostly "virtual"—a result of software applications that configure logical networks over physically interconnected elements, often dynamically. Perhaps it is time to coin a new phrase and take the vision a step further by declaring "The network is the application." From a service and network management perspective, this could be interpreted as "the management application." From an end-user perspective, it would be the ubiquitous applications that connect today's user communities. The phrase is as accurate as its predecessors are, and also quite complementary to them.

## The Application as Driver of the Management Framework

Software vendors in the areas of operations support systems (OSSs)/business support systems (BSSs) have attempted now for a few decades to come up with the best framework, usually starting with a model in mind (TMN, for example). The framework usually pre-defines all the layers and interfaces between layers, as well as framework services and functions that belong in different layers. An implementation technology is then chosen for the framework and an organization that has nothing to do with the applications that need to use the framework gets tagged to develop the framework. The framework itself takes a year or two to develop. By then the technology is no longer the hottest, and the functions and services needed may no longer be the same—but developers do attempt for a while to force-fit applications into the framework to justify the investment.

Companies that have invested heavily in frameworks development have found out sooner or later that without substantial applications delivered as part of the framework they cannot survive as a business. A complex framework requires a lot of development, and the development technology is rapidly changing.




CLI – Command line interface
COPS – Common open policy service
CORBA* – Common Object Request Broker Architecture
EJB† – Enterprise JavaBeans†
JDBC† – Java† Database Connectivity
LDAP – Lighweight directory access protocol
OA&M – Operations, administration, and maintenance
SNMP – Simple network management protocol
TFTP – Trivial file transfer prototol
XML – Extensible Markup Language

*Registered trademark of Object Management Group, Inc.
†Trademark of Sun Microsystems, Inc.

***Figure 1.***
***The "Swiss army knife" application=framework.***

A framework customer has no incentive to deploy a framework developed with yesterday's technology and to start building applications around it. The exceptions are companies that focused on lightweight frameworks that made it very easy for third-party vendors to write compliant applications. The conclusion is that the framework architecture is driven by the management applications, and the implementation is driven by the technology—which can be captured as "The application is the framework." In other words, it makes no sense to promote a framework separate from the applications it supports. The applications are "carrying" the framework; they include it.

In a sense, the applications need to provide the information technology equivalent of a Swiss army knife—they need to include a set of technology "blades," which are adapters that allow them to exchange information with peer applications, to publish/subscribe data, or to dig into database schemas (see **Figure 1**). Practically, success demands that applications be ready to fit into a number of existing frameworks and to drive the evolution of those frameworks as they themselves evolve. Applications that will survive for a long time (becoming legacy applications) will have to evolve in order to be competitive. One way they will evolve is through adapters that will prolong the life of the application within the evolving ecosystem of applications.

## The Management Framework—Reconciling Two Views

Assuming an OSS vendor follows this suggested approach and thus establishes a management framework by contributing to an ecosystem of management applications, there are two key issues that have to be addressed:

1. What kinds of applications, interface technologies, and data stores are required, and how are they maintained, given the development technologies that exist at this point in time?

2. How should the answers to the first question be presented as a vision and strategy to the vendor's customers, both from a traditional perspective (consistent with the TMN model) and from a progressive perspective (emphasizing flexibility, simplicity, performance, and speed to market)?

The answer to the first question has to support/exploit the creation of multi-purpose (reusable) applications, common data models, simplified/common network element behavior, and common operation, administration, and maintenance (OA&M) capabilities. While it has not been mentioned until now, the OA&M practices that support any type of OSS/BSS deployment are critical elements. They directly support the interactions between the first three items and are fundamental to achieving one of the goals of the solution—simplicity.

The answer to the second question is perhaps the most important, because it is the quality of that answer that determines whether the implementation strategy will address the full and necessary scope of the problem.

This section explores these questions from the perspective of two representations. One representation, shown in **Figure 2**, is based upon the TMN/TOM model. It is a simplified representation of a management framework described at a high-level in the Silverman, Brenner, and Shannon paper.[1] This view consists of hierarchical layers and emphasizes the distinct needs for different types of applications and application interfaces, common data models, and separation (abstraction) of network element behavior. It is mostly useful for answering question 1, although it would be the basis for answering question 2 in the context of traditional services management. The other representation was fundamentally described in the previous section and is based upon the "Swiss army knife" application model shown in Figure 1.

This second representation will be explored through an illustrative example of a contrived service provisioning application ("i-Connect"). This is an application that interacts directly with the network elements, without having to go through the network management or element management layers. The power behind this view is its flexibility. It emphasizes the applications' ability to interface with other applications on many levels. It provides an example of how one would answer question 2 in the context of "progressive" next-generation services.

## Implementing a Service and Network Management Strategy

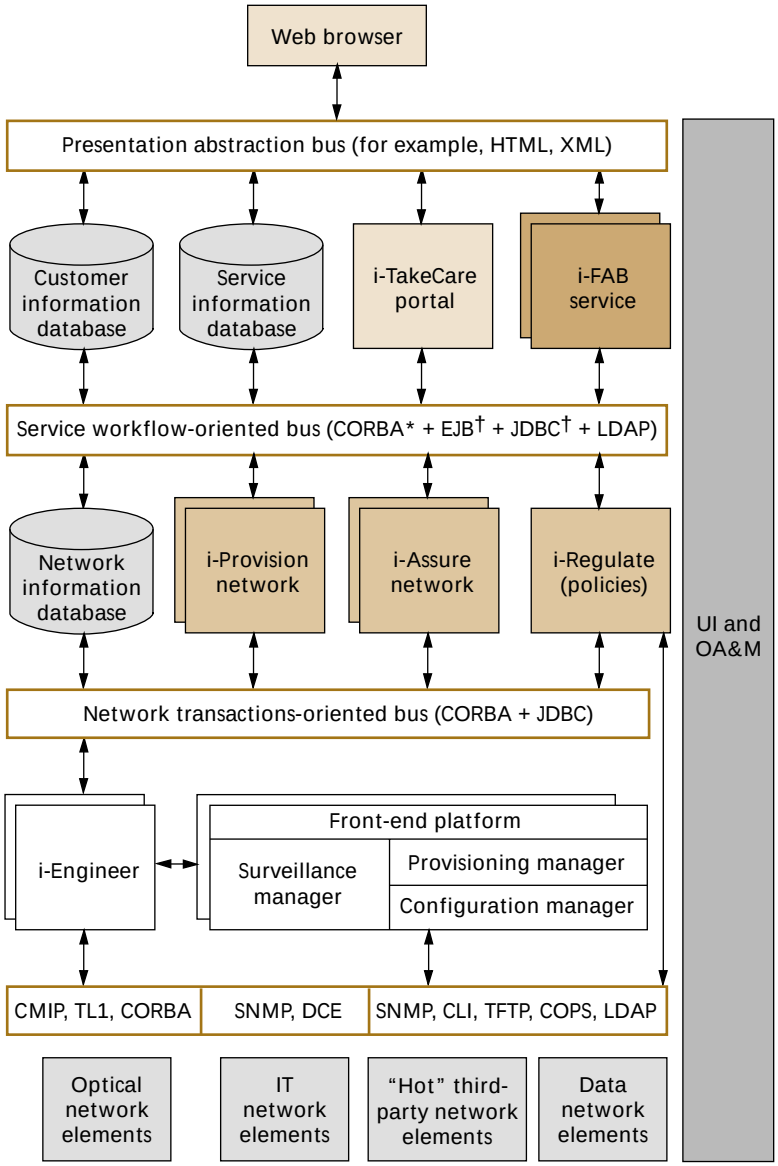
A key message of the section above (and in fact of this paper) is that these two views of the architectural framework are both consistent and necessary to a service and network management strategy. The sections below—"Key Framework Applications and Components" and "The Application Model View"—contain specific suggestions for the implementation of this strategy from both points of view.

### Key Framework Applications and Components

The implementation strategy must begin with what is most needed and expected to manage next-generation networks. It must address how the services that turn faster and larger revenues for customers are supported with regard to fulfillment (service ordering, configuring, provisioning, activation), assurance (service availability, performance, time interval to fulfill order, time interval to handle problems, compliance to service-level agreement), and billing. It also has to address how to integrate the legacy systems and other assets and how legacy operations will migrate to the new paradigm.

Instead of choosing a platform- or framework-based approach to provisioning OSSs, the vendor could use off-the-shelf management applications (internally or externally developed) and construct a Web-enabled OSS complex, fronted by a self-care management portal. This will allow the vendor to be in step with "new and cool" service provisioning, while still having to fix the underlying problems (which are everybody else's problems, too). The following subsections expand upon this basic strategy by identifying and defining the essential components.

**The management portal: gate to service fulfillment, assurance and billing.** The self-care management portal (shown in Figure 2 as "i-TakeCare") could be built in-house, acquired, or obtained though partnerships. It is not expected that a management solution vendor will make a lot of money by selling the




CLI – Command line interface
CMIP – Common management information protocol
COPS – Common open policy service
CORBA* – Common Object Request Broker Architecture
DCE – Distributed computing environment
EJB† – Enterprise JavaBeans†
FAB – Fulfillment, assurance, billing
HTML – HyperText Markup Language
i – "Integrated, intelligent, and innovative"
IT – Information technology
JDBC† – Java† Database Connectivity
LDAP – Lighweight directory access protocol
OA&M – Operations, administration, and maintenance
SNMP – Simple network management protocol
TFTP – Trivial file transfer prototol
TL1 – Translation Language 1
UI – User interface
XML – Extensible Markup Language

*Registered trademark of Object Management Group, Inc.
†Trademark of Sun Microsystems, Inc.

***Figure 2.***
***The framework (a TMN-like view).***

portal (however, it is expected that a vendor will lose money by not having one, because it will look like it is pushing yesterday's technology). The self-care portal is a user's porthole into the OSS/BSS solution (a *user* being defined as an end-customer, a sales person, an operator, or any other role player).

The portal will first authenticate and authorize the user, via a lightweight directory access protocol (LDAP) directory fronting pointers to other management applications, as well as customer, service, or network data to which the user is allowed access. Most of the data itself will probably not reside in the directory, but in DBMSs. Selective data, rarely modified but frequently accessed, may be cached in the directory itself; otherwise, the directory will simply contain pointers to the real data.

The portal will be accessed via a Web browser that is equipped with Java* applications necessary for presentation via the presentation abstraction bus built with Extensible Markup Language (XML) technology to allow flexible interfaces to other technologies. It will not only serve as a conduit to place and update orders as part of the fulfillment process. It will at the same time allow users, depending on their authorization, to check their activated services for availability and performance status, service level, and problem reporting, as well as billing reports for the services ordered.

**The application transaction bus.** An application transaction bus based on enterprise application integration (EAI) technology such as Enterprise JavaBeans* (EJB*) will provide us with an application platform that has high flexibility and scalability. The recommendation is to incorporate a workflow engine using this technology. A number of middleware vendors have solid workflow engines, including EJB-architectured "bus" technology. Alternatively, tools from other vendors could be used to develop the appropriate bus. Yet another approach would be to use the middleware and workflow engine a vendor already has in place and a future EJB bus around which future management applications may be built. The use of a Java-based architecture for the upper-layer of the OSS complex would provide any vendor with an application platform that has high flexibility and scalability. An application transaction bus built with EJB technology would allow plugging in of various service-related applications, such as service ordering, new service creation, service provisioning, compliance analysis, and billing. Some of those applications will require access to customer and services data, modeled by a published information model.

Access to the data will be through the same application transaction bus, using objects enabled with open database connectivity (ODBC) and/or Java Database Connectivity (JDBC*) adapters. It is extremely important to completely separate the data from the applications, to ensure that new applications that understand the published information models can access and use them without having to be dependent on other applications. That will allow the combination of applications into an integrated solution and the deployment of applications individually for small starts that do not require a complex solution. It will also allow product houses to make intelligent choices about what they need to build themselves, versus what they can reuse. The data models will have to be implemented in a reusable fashion, and the applications will have to exhibit sufficient independence. It is conceivable that in certain cases a group of applications together would meet the "sufficient independence" criteria.

The documentation accompanying applications should clearly articulate how applications work, individually or as a group, in providing a well-defined function. It is mandatory that the applications that are fronted by a presentation module (for example, a GUI) be developed in a fashion that will allow the same information that is collected or reported via the presentation module to be accepted as input or sent as output via the application bus. That will ensure that application-to-application interactions will be easy to achieve, allowing for a less painful flow-through process.

The truth is that this is a blueprint for an evolving framework; therefore, no application can fully predict where it will end up in the functional architecture. Thus, it would be unwise to not separate the presentation module from the algorithms of the application.

**The network and information technology infrastructure management.** Under the "layers" so far

described lie all the classical fault, configuration, accounting, performance, and security (FCAPS) systems known as element managers and network managers. Those tend to be systems that reflect (unfortunately, too much) the realities of the physical elements and the networks composed of physical elements from which they are trying to engineer, monitor, maintain, troubleshoot, upgrade, and collect information. This includes information needed for forecasting additional capacity, for being able to bill for the usage, or for providing data that can be compared against what was contracted. The traditional network includes network elements, but considering that the network is also "the computer" and "the database" and "the application," Lucent and other vendors are faced with the prospect of having to manage an increasingly complex solution with the addition of servers, storage solutions, and applications. Assuming that an end customer is using a service that traverses the Internet and accesses a page on a Web site, it is of little relevance to the customer if failure to receive a response is due to a failure in the "traditional network" (Internet or the LAN in the data center) or failure of an Apache Web server running on a Sun host.

In addition to the existing elements, new elements appear with an increasing frequency, and some of them expose new management behaviors. Most of the elements or network management systems, existing or planned, have very well identified targets. They do their job well for specific elements, and most of the time it is because the particularities of the elements force them into certain implementations. However, these systems need to expose a northbound interface (towards the service management applications) that is more consistent and standards compliant. That means it is unlikely to be changed very often, very reliable, and near real time. For a network management software vendor in particular, it is expected that this is an internal interface that allows a services application to talk to network and element management applications, with the purpose of sending in specific provisioning parameters and collecting well-defined alarms, performance reports, and usage counts (or pointers to data collected in files).

A bus that allows pair-wise interactions between applications via well-defined CORBA* interface definition languages (IDLs) is the industry standard. Stability for this type of pair-wise interaction (as opposed to the flexibility desired at the services applications layer) is derived from the fact that, once deployed, those applications will be used for managing long-term network infrastructure. The degree of change decreases as one moves from services to networks and elements. CORBA technology is also a popular choice with many OSS vendors (Lucent supports CORBA interfaces in the Inter-Domain Configuration Manager, Navis, and WaveStar™ product lines).

Initially several IDLs will exist; over time the expectation is that the number of different IDLs will remain contained and reused by new-emerging systems, rather than continuing to grow indefinitely. Of course, a minimal requirement is that all of the vendor's management applications exchanging information over the CORBA bus use the same CORBA middleware (not a mandatory requirement initially, since Internet Inter-Orb Protocol [IIOP*] can be used to exchange information between different CORBA implementations, but certainly a desired requirement in the long run, allowing the vendor to use development teams more efficiently). That will allow for reuse of network management or element management application "shells," simplifying the development and integration of new network applications in support of new network elements or the combination of different types of elements. Those "shells" need to be engineered around a well-defined strategy of specializing applications (for example, the "role-based" approach that is currently pursued by many vendors, including Lucent).

The "role-based" approach addresses the major human operations roles required for a network operator and supports those roles through focused applications. The "i-Engineer" application, shown in Figure 2, is focused on network engineering, including physical inventory, cards, ports, physical circuit installation and configuration, and facilities availability and performance reporting, as well as troubleshooting tools. The "i-Provision" application will support assignment of facilities; changes to network elements during service provisioning process; logical circuit modifications (cre-

ation, deletion, editing); and management of transactional workflow, including back-out, monitoring, and troubleshooting of logical circuits. An "i-Assure" application supports end-to-end circuit views, service performance monitoring, service alarm presentation, network-correlated faults by circuit, service, customer, or history, trend analysis, periodic reports, and service-level agreement (SLA) compliance reports.

One of the desirable goals of such a strategy is to produce "shells" of "i-Engineer," "i-Provision," "i-Assure." Those "shells" would serve several purposes. One would obviously be the one previously mentioned (reuse by another organization when the complete original application cannot be reused, because of the need for different algorithms in performing the task). A second purpose would be to use the "shells" to wrap any third-party vendor applications, or to allow third parties to quickly build appropriate management applications for "hot" new elements that the vendor may want to support. Last, but not least, the vendor could learn from and improve on something that at least one vendor does well today: when and if an opportunity exists to acquire another company that has a "hot" network element, one of the preconditions would be for the vendor to fit the acquired company's management applications into the vendor's pre-existing management "shells."

Becoming more systematic in approaching management interfaces and data exposed by network elements is critical to the implementation of this new vision for management. The network and element management "shells" described before will force conforming behavior of the elements that they manage.

While a large variety of possible interfaces are in effect today, simple network management protocol (SNMP) seems to be the "simple" preferred choice today for fault and performance measurements; trivial file transfer protocol (TFTP), the choice for hardware configuration; and command line interface (CLI), the choice for access control lists. There is no clear winner in policy management with respect to the protocol for communicating policies to the elements (COPS, DIAMETER, LDAP all vie for a place). However, it is relatively obvious that systems are moving away from CMIP/Q3 and TL1, and few elements are successfully implementing CORBA in the embedded software. Therefore, the recommendation is to go with SNMP, CLI, TFTP, and one of the policy communication protocols (the one that best fits the needs of the specific element).

The policy management applications ("i-Regulate") will have to support all de-facto standards.

**Data models.** Three main categories of data models are suggested here.

The *Customer Information Model* includes the data and the behavior of the processes interfacing with customer-specific information. That should include all the information needed to be captured for any category of users for any service offered by a service provider (customer organization, end users at the customer's location, sales force, operators), authentication and authorization of those customers for services offered, pointers to services active for each customer. All the customer-related information should be kept in one logical data model, accessible by any application that needs to access that information and is authorized to do so. It is expected that a well-defined and well-implemented customer information model can be reused by all integrated management solutions and should be designed and built only by one of the vendor's development organizations.

The *Service Information Model* includes the definition of all services offered by a service provider via a service catalog; for specific users, a repository of all services in use and their current state (order state, billing state, operational state) needs to be kept up to date. Each service defined should carry a service profile that would allow a selection of SLA classes, characterized by quality of service (QoS) attributes. The model has to allow the flexibility of mapping of specific SLA metrics into QoS classes, and both the type of QoS classes and the SLA metrics need to remain extensible. The service definition needs to allow for services inheritance and lateral dependencies on other services, as well as support the behavior needed to create new services. Basically the model has to support complex services. It also needs to include classes related to orders, mirroring the user's request. The user's request will be decomposed into several network order requests that can be supported by the network model. It is expected

that a well-defined and well-implemented Service Information Model, designed and built only by one of the vendor's development organizations, can be reused by all integrated management solutions.

The *Network Model* needs to address all types of technology-driven networks. The model will address network topology, managed elements, subnetworks, and traffic descriptors. Other common data to be represented in the network model includes physical inventory data, problem-handling data, performance data, capacity data, alarm-related data, and routing protocols data. Both inter-technology domain and individual domain network models are needed to complete the model. A large-scale deployment may need the implementation of all the classes of the model, while a smaller-scale or specific customer segment may only need a subset of the network model. It is expected that a well-defined and well-implemented network information model can be designed and implemented by one of the vendor's development organizations. This will allow for it to be reused across customer segments and scaling requirements, while facilitating other organizations to contribute new classes that handle specific aspects of their technology domain to the common data model.

**OA&M tools.** A number of requirements apply to the common use of OA&M tools. This is an area that is less glamorous than other areas for developers, and it is rarely a sellable feature. Therefore, it is frequently considered as an afterthought. Paradoxically, these are good reasons for dealing with this rarely and consistently—in other words, it makes a lot of sense to establish a set of reusable tools and procedures and enforce them rigorously in all management applications, including the integrated result of such applications. Accepting that as a guiding principle usually implies standardizing on certain hardware platforms, operating systems, and software tools, which seems to be the more difficult issue. While this is not something that can be resolved immediately, it is something that can be imposed as a requirement for new management applications and solutions—as part of the "shell" concept.

The other non-negotiable requirement should be for management applications not to be developed as a standalone system (more often than not, an OSS vendor's applications take control of the host they use as a platform, deliberately or accidentally). As a viable requirement, a vendor could decide that all new management applications to be developed first be implemented on a preferred hardware and software (operating system) platform (further porting being driven by business needs).

This policy will immediately facilitate other practices. Software packaging, installation, distribution and update procedures may easily be developed around a primary software choice, including tools provided by the same vendor. Standard startup and shutdown procedures should be the same, and a similar approach could be taken with respect to DBMS to be used initially—which would then simplify the choices for data backup/restore. A standard tracing package should be easy to adopt across applications, as long as one organization would volunteer to provide and maintain it on the platform of choice. It would most likely have to provide a C++ and Java library. Following such requirements would at least make possible, if not probable, co-residency of management applications on the same host and therefore reduce the number of "boxes" needed in the solution.

### The Application Model View

**Figure 3** illustrates how an optical connection provisioning application would drive the management framework. This only illustrates a subset of the service fulfillment process, and it does not illustrate at all the service assurance or billing aspects of a complete solution. The application ("i-Connect") would have XML, EJB/J2EE, CORBA, and JDBC "blades," in addition to a "blade" for signaling connections data to the optical network elements. The XML would be used to present user screens with dynamic content, rather than the static representation available through HyperText Markup Language (HTML). This would allow the Web pages and product catalog that would be presented via the "i-TakeCare" portal not to require revision every time the application changes. The XML interface would also be exposed (as an option) as a loosely coupled flow-through interface to an external OSS.

The portal would have an LDAP "blade" that would allow the connection request to be authenti-

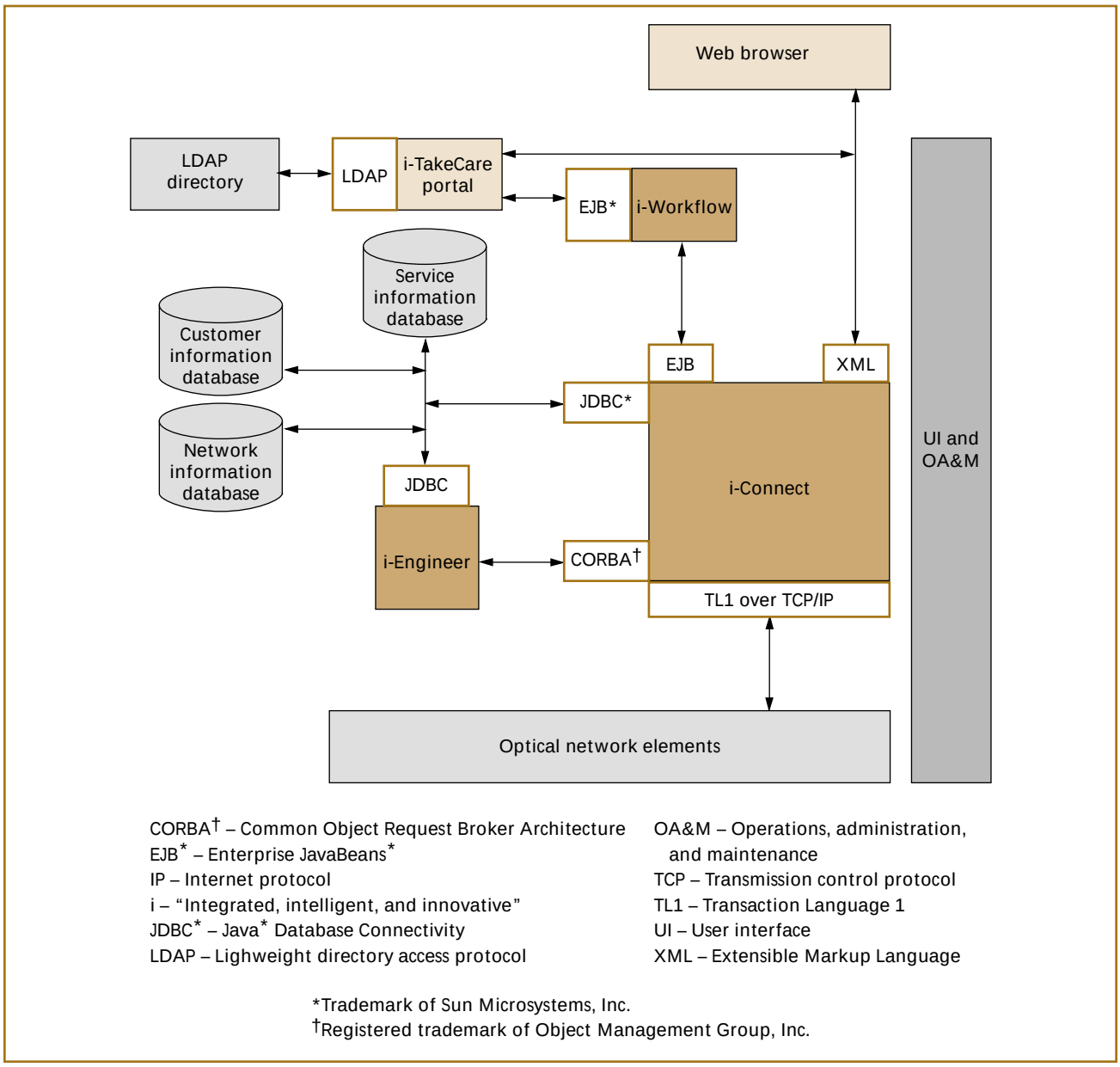



***Figure 3.***
***The framework (an application view).***

cated and authorized with the help of customer information previously stored in an LDAP directory. The portal would be implemented using EJB/J2EE technology, allowing it to interface to the "i-Distribute" engine. The interface to "i-Connect" (or the entire "i-Connect" application) could be implemented using EJB/J2EE technology. The "i-Connect" would interact with the service information database to extract information about the specifics of the service requested, such as class of service and SLA metrics associated with the class. This information would be processed by the application to determine QoS parameters needed to be signaled to the network elements. The "i-Connect" application would obtain physical/logical port and topology information from the network information database, either using direct access via its JDBC "blade" or using the "i-Engineer" application as a proxy process (a design choice). A third alternative

would be for the "i-Connect" application to obtain all the network-related data directly from the elements, without any need to access the network data repository. Using data entered or subsequently obtained and processed, the "i-Connect" application can now signal the connection directly to the optical elements via a to-be-defined "blade" (a standard—for example, optical domain service interconnect [ODSI]—or proprietary protocol over TCP/IP or SNMP or CORBA, depending on the protocol supported by the particular network elements).

## Approaching Policy-Based Management

Discussing how this implementation strategy relates to policy-based management is beyond the scope of this paper. However, we believe that the two views presented here and the work that is going on in the Internet Engineering Task Force (IETF) in policy management are consistent and complementary. To illustrate this belief, we offer the following observations.

Extending the "role-based" strategy beyond the handling of element and network services would make for a more powerful strategy. A policy management application that will handle QoS policies, security policies, and any other administrative (OA&M) policies for any type of network should be common to any solution beyond a small-scale customer. While the policies may differ by market segment or technology domain, how they are communicated and administered is not something that every organization needs to develop in a vacuum.

Policy management could be further extended as a complement to the "i-Distribute" application by associating rules/policies to handle routing of order information, provisioning information, and reporting information. Furthermore, policies would then be developed to automatically handle decomposition of a bundled service into objects that "i-Provision" types of applications can handle or to compose the results of "i-Assure" types of applications. The latter would then be fed into an analysis and diagnosis application that would weigh the composite assurance result against a contractual obligation represented by an SLA with a specific customer ("i-Comply").

A good example of a market segment where this type of application is becoming increasingly important is service providers offering services that cross several administrative domains and technologies (the so-called "CyberCarriers,"[4] ASP aggregators or services brokers). A typical service like this would be offering access from an enterprise or residence to applications hosted in an Internet data center. The packets exchanged between the user and an application hosted in a server in the Internet data center will traverse access equipment, possibly metro-rings, a core backbone, an Internet data center LAN infrastructure, servers, and storage—and those may be either owned or at least administered by different organizations.

An end-to-end service provider would have to break up an order for such a service into the appropriate service components (via an "i-FAB" service, with *F* for "fulfillment"), which then will be provisioned via "i-Provision" applications. The "i-Assure" applications would provide assurance for each of the components of the service. An "i-FAB" service will then collect all the assurances from the components of the service and create a composite assurance report that can be matched against the contractual obligation. These types of applications are really technology or service independent and therefore definitely reusable across different customer segments.

## Conclusions

This paper provides recommendations for the implementation of an architecture framework that promotes commonality and simplicity, while at the same time supporting the flexibility required to address the challenges service providers face in operating and leveraging their networks today. It includes a set of recommendations on how to accomplish such an integrated framework and justification for those recommendations. The recommendations take into account the need for a new vision, while at the same time providing an opportunity to capitalize on a vendor's best assets and successful strategies in place. If implemented, such a framework will provide product houses with the opportunity to reuse every other management application created as part of the framework, without forcing product houses into waiting for

a capability that has not been created yet (therefore encouraging them to develop it and extend the framework's value).

Certainly it is desirable to develop some guidelines/policies around such flexibility. The most important one would have to be that the management application comply with the evolving framework—to the point that it either is completely self-contained or it has been integrated with other applications that have passed that litmus test beforehand. The vendor's product houses that currently build management applications will have to relinquish some control, since they will have to rely on other partner organizations to contribute to the completeness of the solution. This price will be offset by the benefits associated with focusing resources on building specific applications that add the most value for the service providers and hence produce the greatest major for the software vendor.

The strategy will be well received regardless of whether the networking vendor leads with products or with an integrated end-to-end solution. When leading with products, the vendor will be able to offer role-based, focused applications, that either can be easier integrated into a customer's solution with multi-vendor products or can be scaled up to the complete vendor's management solution when the customer is ready. When leading with an integrated solution, this strategy would be much easier to communicate, sell, and deploy, because it addresses end-to-end integration, openness and flexibility and can keep pace with the innovations in technology.

***Trademarks**

CORBA and IIOP are registered trademarks of Object Management Group, Inc.

EJB, Enterprise JavaBeans, J2EE, Java, and JDBC are trademarks of Sun Microsystems, Inc.

**References**


1. K. S. Silverman, M. R. Brenner, and G. E. Shannon, "Toward a Vision for Network and Service Management," *Bell Labs Tech. J.*, Vol. 5, No. 4, Oct.–Dec. 2000, pp. 21–30.
2. International Telecommunication Union, Telecommunication Standardization Sector, "Principles for a Telecommunications Management Network (TMN)," Rec. M.3010, May 1996, <http://www.itu.int/ITU-T>.
3. TeleManagement Forum, "Telecom Operations Map," Document GB910, Version 2.1, Mar. 2000, <http://www.tmforum.org>.
4. M. R. Brenner, M. Chu, G. Gross, and M. Malek, "CyberCarrier Service and Network Management," *Bell Labs Tech. J.*, Vol. 5, No. 4, Oct.–Dec. 2000, pp. 44–62.



***(Manuscript approved March 2001)***



*WILLIAM C. GOERS is director of Offer Definition, Broadband Networks Group, at Lucent Technologies in Holmdel, New Jersey. He leads a group of networking engineers and architects in defining high-level solutions to help Lucent's customers meet their business objectives. These solutions, or "offers," are based on industry-leading technologies from across the full breadth of Lucent's product portfolio. Each focuses on providing value-added services to end-users. With many years of experience in communications technology, Mr. Goers has worked with compressed digital video conferencing control systems, satellite communications network control and monitoring systems, packet data communications systems, and digital cross connects. A member of IEEE, he holds M.S.E.E. and B.S.E.E degrees from Texas A&M University in College Station.*



*MICHAEL R. BRENNER is a technical manager in the Network Solutions Architecture Department at Lucent Technologies in Holmdel, New Jersey. He leads a group of architects who define service and network management and application architecture in support of Lucent's network-hosted services solutions. He joined Bell Labs to work on applied expert systems and OSSs after an extensive career in software design and development for real-time systems, medical imaging processing, and pioneering work in picture archiving and communication systems. Later, he developed and managed network management software for Lucent's ATM switches and for forward-looking technologies, such as mobile agents and policy management. Mr. Brenner holds B.S. and M.S. degrees in computer science from the Polytechnic Institute in Bucharest, Romania.* ◆