The RACE Open Services Architecture project

by A. O. Oshisanwo
M. D. Chapman
M. Key
A. P. Mullery
J. Saint-Blancat

The specification and implementation of current telecommunication services tend to be intimately bound to a specific network architecture. Moreover, within the service software, interactions between the logical modules are not always explicit, accessible, or uniform, and tend to be optimized for a particular service. This is exemplified by the difficulty experienced in integrating equipment from multiple vendors, and has resulted in telecommunication systems that cannot rapidly exploit the advantages of new technology or respond to changing customer requirements. In addition, current telecommunication services tend not to be viewed as an integral whole, whereby user, control, and management aspects of a service are developed independently from one another. Separate development can lead to problems of inconsistency if shared data are not updated correctly. The RACE Open Services Architecture (ROSA) project was established to address these problems. This paper presents an overview of the approach taken in the ROSA project.

The objective of the RACE[1] Open Services Architecture (ROSA) project is the definition of an open architecture for integrated broadband communications (IBC) services.

To understand the need for an open service architecture, it is necessary to examine the current situation in telecommunication networks and services. Today's telecommunication networks are not flexible with respect to changes in service requirements or to the introduction of new services because of a number of factors, including:

- The lack of a network-wide architecture and thus the coexistence of node-bound architectures
- The limited capability for interworking between services
- The close coupling between service management and network management
- The complexity of the interworking of subnetworks based on different architectures

All of these factors make the introduction of service or its modification complex and expensive because of the redesign that is necessarily involved.

For example, the Intelligent Network (IN)[2] platform is a means by which network operators provide advanced telephony services such as Free-Phone and premium rate telephony. The IN platform is overlaid on the basic network architecture, extending the architecture for those services that require the added IN functionality. While

the basic principles and architecture of IN are being standardized, a number of vendors are offering platforms that have very limited interworking. As a result, it is currently not possible to construct a service out of components from the IN platforms of different vendors.

In the view of both the standardization bodies and relevant RACE functional specification projects, integrated broadband communication networks (IBCN) should support an extremely wide range of services. This view is supported by the adoption of asynchronous transfer mode (ATM), a flexible and service-independent transfer technique, as the future standard for the transport infrastructure. To support this view in the higher layers of the network, a service-independent organization of functions should be pursued as far as possible to achieve the maximum flexibility with respect to service evolution during the lifetime of the IBC system.

Within the above perspective, ROSA aims at providing an architecture for the specification, design, and implementation of "open" services. Such an architecture must support the smooth introduction of new services, the graceful evolution of existing services, interworking between existing and new services, and an increased independence of the architecture and services from new and evolving network technologies.

This paper describes ROSA, the principles on which it is based, and the service specification methodology that has been developed for effective use of the architecture components. The paper is based on the results reported in the ROSA deliverables.[3-7] Although the ROSA project ends in December 1992, the definition of the ROSA architecture is being continued in a companion RACE project called CASSIOPEIA.[8]

The scope of the ROSA project

The terms *open*, *service*, and *architecture* are now defined with the aim of characterizing the objectives, the field of applicability, and the meaning of the ROSA architecture.[4]

Open. The IBC environment will have to meet the need for rapid changes in software, equipment, and network structures due to advances in technology and to an ever-increasing demand for new types of services by users and operators. An open system of services is one in which such changes can be executed in the most efficient and cost-effective manner. Such flexibility can only be achieved by creating a system in which:

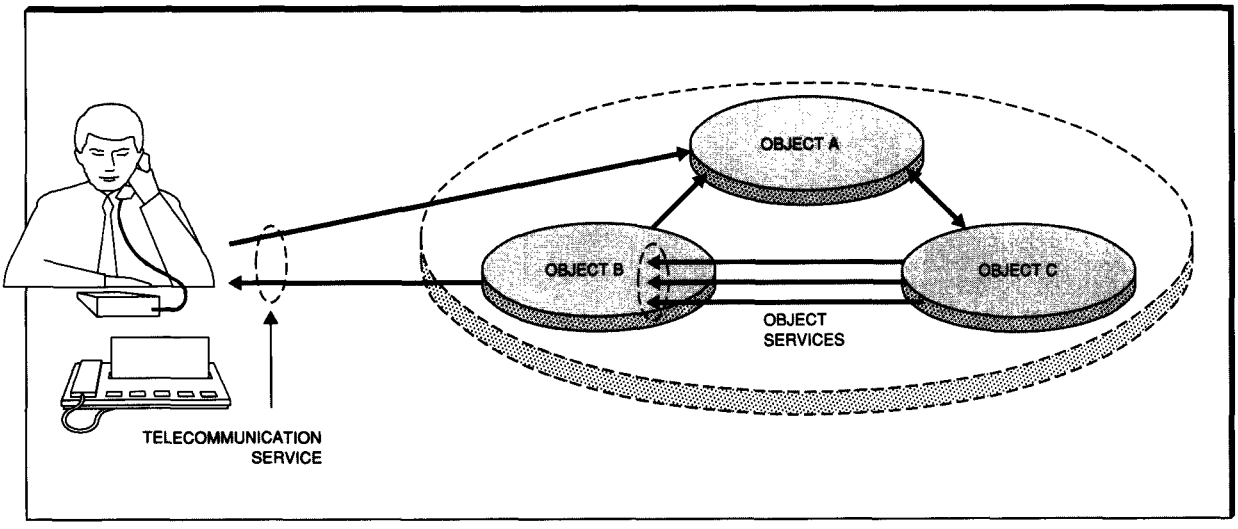
- Introduction of new services will be significantly faster than current techniques allow and will have a minimal impact on existing services.
- Elements of existing service specifications will be easy to reuse in new designs.
- Any change in implementation technology will not impact the existing services, provided that it is performed by respecting some predefined criteria.
- Interaction between services will be easier to identify and define than it is today.
- Service and network management will be treated in a way that is compatible with the principles laid down by the Telecommunications Management Network (TMN).[9]
- Service management and service core functionality will be treated in an integrated way.
- Service descriptions will be portable across various network infrastructures.

Thus, ROSA will be used to design and provide services that are open in *space*, *time*, and *technology*.

Service. In the ROSA approach, the term *service* is used in two different contexts, as shown in Figure 1. In an *enterprise* context, a service is defined as a meaningful set of capabilities provided by an existing or intended network to all who utilize it: customers, end users, network providers, and service providers. Each one sees a different perspective of the service. This usage of the term is congruent with the current notion of telecommunication services, i.e., basic communication services, advanced supplementary facilities, value-added services, distributed applications, telecommunication network management applications, etc.

ROSA uses an object model in which a service is defined as a single capability that can be invoked on an object instance of a given object type. This definition is the basis for the second context of the term service, where the term relates to the *computational* terminology adopted in the ROSA Object Model.

Figure 1 Different contexts of "service" in ROSA



ROSA has adopted this terminology for the following reasons. On the one hand, the meaning of the term "service" in the telecommunications world as a complete set of capabilities offered to users is too well established to try to change it. On the other hand, the term "service" as a single capability offered by an object is too convenient for one not to use it in an object modeling context.

A ROSA specification of a telecommunication system at the highest level of abstraction is given in terms of the "enterprise services" available to the different users. Such services are modeled as high-level objects.The high-level objects are defined in terms of objects of finer granularity and are depicted in Figure 1.

Refinement to objects of finer granularity is accomplished by decomposing the high-level object according to a number of criteria that ensure that the services are open. At the lowest level of abstraction, entities within the telecommunication network are defined as objects that can work together to produce the effects described as capabilities offered by higher-level objects. Thus, object orientation is consistently used to refine the specification of services at all levels of abstraction. Even the entities within a real telecommunication network are functionally represented as objects.

Architecture. Architecture generally means an observable style adopted to build systems of some complexity (for instance, buildings and computing systems). The adjective "observable" is fundamental in this definition, because an architecture deals with the external appearance of a system and its components down to a given level of granularity; it is not concerned with the internal realization. An architecture addresses a set of concepts, rules, and recipes (i.e., solutions, artifacts) to be used in design. A system designed by adopting the concepts and rules of a given architecture is said to conform to that architecture.

As an example, in the context of distributed computing systems, an architecture embodies basic concepts or elements such as record, file, directory, node, and process, among others. It also includes rules such as "a single file can be transparently split into different volumes but not on different nodes"; and "a single file can be transparently duplicated and consistently maintained in different volumes or nodes." In addition, it includes recipes such as naming schemes, information representation schemes, communication schemes, and exception handling. These concepts, rules, and recipes provide a characteristic structure to the applications that run on the system and may thus be deemed to be the system architecture.

The ROSA notion of architecture aims at supporting an open service scenario. ROSA can be characterized as containing:

- Concepts for specifying, designing, and implementing IBC services
- Rules for specializing, refining, and combining the components of the architecture
- Recipes that provide robust solutions to well-known problems and models for telecommunication service specification

The concepts, rules, and recipes of the architecture are embodied in a set of Reference Object Types.

Although ROSA will be used to design and implement services, the architecture is not biased to a specific set of services, a particular network technology, or a special target infrastructure. Issues and requirements that are common to the services to be offered on the IBC infrastructure are taken into account.

A key point in ROSA is its focus on telecommunication services and only subsequently on network entities. This focus is consistent with our view that a telecommunication system architecture can only be stable if we concentrate first on *what* a system is meant to provide rather than on *how* a system will be internally organized.

The principles of ROSA

ROSA advocates the specification of a telecommunication service as a set of interacting objects at several levels of abstraction. At the highest level of abstraction, a service can be viewed as a single object. This abstraction helps the designer to formally capture the requirements of each user who has a stake in the service.

The next level of abstraction emphasizes the logical partitioning of the functions needed to meet the requirements of the service, where the single object is now decomposed into a set of interacting objects. The rules and guidelines for the logical partitioning of functions are provided by ROSA. However, issues such as the physical distribution of the functions throughout a network and mechanisms to achieve communication and performance requirements are not of concern at this level. This allows services to be specified to a common and standard logical structure at a high level of abstraction, delaying the introduction of implementation-dependent details until later. By concentrating on the service being offered, we can more easily grasp the essentials of the service and see how it needs to interact with other services. Relationships between objects can be modeled in order to obtain service-level interoperability.

The ability to create logical service models using architectural concepts and object orientation gives ROSA its openness characteristics.

Implementation concerns such as distribution, communication, performance, etc. are introduced into the service specification at the lowest level of abstraction. With ROSA these concerns can be modeled. Focusing on these concerns at a low level of abstraction encourages service specifications that are independent of specific implementation constraints and promotes their applicability to a variety of network technologies.

When telecommunication services are modeled as a set of interacting objects, the objects provide a basis for defining standard components and interfaces. If the concepts and rules of the architecture are applied to the objects used to model telecommunication services, objects and the services they model will be open. It is the ability to create logical service models using architectural concepts coupled to object orientation that gives ROSA its characteristics of openness.

The rest of this section describes the basic techniques used in ROSA to define an architecture to promote open service definition. First, the *ROSA Object Model* (ROOM) is discussed. The ROOM is used both to describe services and to describe the components of the architecture in an object-oriented style. This is followed by the general principles of ROSA.

The ROSA Object Model. The ROOM has been designed so that telecommunication services can be described in an unambiguous and open way. The model consists of a set of concepts relating to abstraction, specialization, and composition, which are key features for constructing models. Here, a brief summary of the key features of the ROOM are presented; further details can be found in Reference 5.

The basic terms of the ROOM paradigm are objects since they are the one and only category of entities that can exhibit properties and behavior. An object is an abstraction of a problem domain entity used to model the characteristics of that entity. An object is described by an object type that defines the interface, internal behavior, and usage restrictions of objects of that type. Objects interact with each other, using a common interaction style in which objects invoke services on other objects and may receive responses to such invocations. The internal actions performed by an object to carry out a service are not visible to the requesting object. By default, all objects are concurrent in that many services may be processed at a time and services may be invoked in any order. However, in some cases it may be necessary to restrict the amount of concurrency allowed and impose sequences of service invocations. Such requirements are described in the usage restrictions of an object type.

The main characteristics of ROOM are:

- Encapsulation, whereby an object hides from its users its internal details, and its state can only be affected by invoking services on it, thus providing the ability to change the internal details without affecting users
- Abstraction, whereby specifications are written that focus on what a service offers, omitting details on how it is implemented
- Composition or decomposition, whereby an object can be composed of, or decomposed into, other objects, thus providing the ability to break down a problem domain
- Inheritance, whereby object types can be defined as specializations or extensions, or generalizations of other object types. A specialized or extended object type inherits all the properties of a general one and may then refine the properties in a consistent way. Multiple inheritance allows more than one general object type to be specialized by an object type.
- Reusability, whereby either specifications can be reused through use of inheritance or objects can be reused through composition, thus bringing down the cost of developing services and introducing them into a network

A notation has been defined, called COOLish,[5] that allows properties of objects to be expressed.

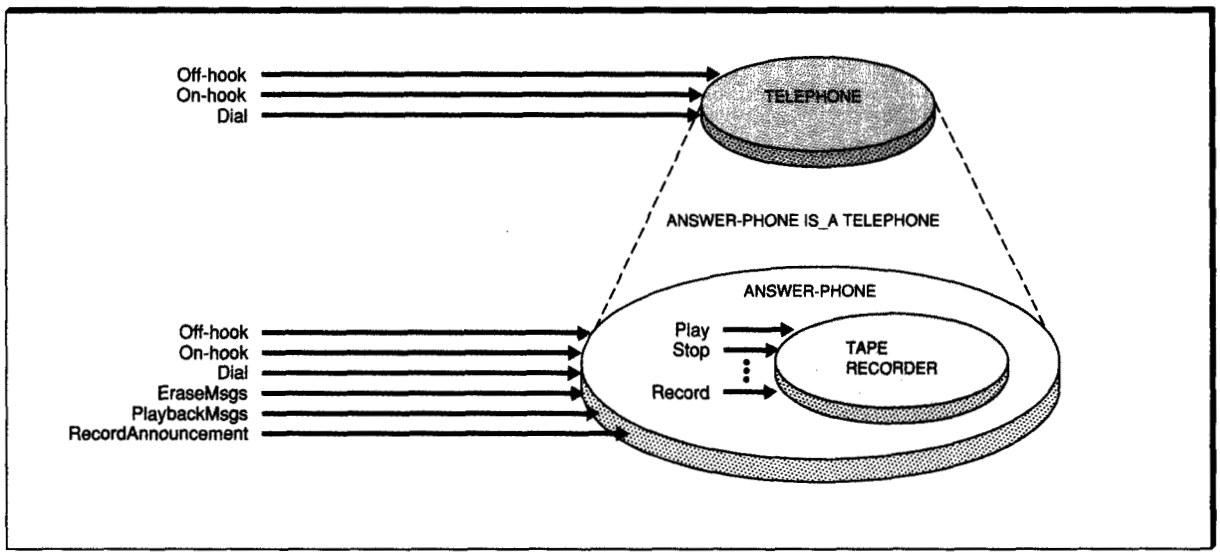
As an example to illustrate the use of object orientation, a model of an answer-phone is presented. An answer-phone is a device that behaves in a manner similar to a normal telephone, except that if the telephone is not answered after a predefined number of rings, the device picks up the call. On answering, it plays an announcement to the caller and records the caller's message.

To model this device, three concepts are defined: a telephone, a tape recorder, and an answer-phone. A telephone provides three services—off-hook, on-hook, and dial—that allow a user to make calls, answer calls, and disengage from calls. The answer-phone can be modeled as inheriting the properties of a telephone and being composed of a tape recorder (Figure 2). The answer-phone must customize the behavior of the telephone to cope with pickup, announcement playing, and message recording, resulting in specialization. Thus, the answer-phone provides the off-hook, on-hook, and dial services of a telephone but additionally provides services for recording an announcement and for the playback and erasure of messages left by a caller. The tape recorder is an encapsulated object, which means that the answer-phone does not know of its internal operation and may only instruct it to play, record, rewind, or fast-forward the tape.

Besides providing additional services to the telephone and changing the internal structure of the telephone, the answer-phone must refine the behavior of a telephone. The behavior of the answer-phone must be a consistent specialization of the telephone in that it must still behave in the same way as a telephone when the answering capability is not used. Formal methods are used in ROSA to ensure that such consistent extensions are made. If such a consistent extension is made, it can be said, in COOLish, that an answer-phone *is_a* telephone.

Architectural framework of ROSA. The purpose of an architectural framework is to appropriately

Figure 2 An object model of an answer-phone



structure an architecture such that it is logically organized in a relevant way for the problem domain being addressed. Such a structure should position architectural components relevant to one another, guide the selection of appropriate components, and help place boundaries on an architecture. Two key principles provide the means for defining a framework:[10] *viewpoints* and *aspects*. Rather than deal with a system as a whole, viewpoints can be used to provide different levels of abstraction of the problem of interest. By concentrating on one viewpoint at a time and ignoring the others, the designer is able to focus on the problem at a particular point in time. By combining all viewpoints, a complete system can be described. Aspects, in contrast, relate to a specific set of problems to be solved or characteristics to be exhibited. In general, aspects pervade the different viewpoints.

In ROSA, two viewpoints and six aspects have been identified. The two viewpoints are called the *service specification framework* (SSF) and the *resource specification framework* (RSF). The SSF is intended for a service designer who is concerned with specifying what a service provides, taking account of the requirements of end users, customers, service providers, network providers, and service managers. The RSF is intended for a system designer who is concerned with how a service can be provided by devising, specifying, and combining basic telecommunications and computing resources in order to ensure that a design is realizable. Thus, during service specification, the SSF is used to describe *what* is to be provided by the service, and the RSF is used to describe *how* the service should be provided.

The six aspects are *service modeling*, *access modeling*, *transport modeling*, *management modeling*, *object support modeling*, and *nonfunctional modeling*. Service modeling addresses the problem of providing information and functional concepts to define the core functionality of a service. Access modeling pertains to user identification, information presentation (both to and from a service), service selection, and end-user customization of the telecommunication system behavior. Transport modeling provides for exploring concerns for the definition and enablement of information transfer between separated entities. Management modeling concentrates on the concerns of telecommunication service and network management. Object support modeling involves those issues that relate to the specification of a service as a set of interacting objects in an open distributed information processing environment. Finally, nonfunctional modeling pro-

vides for the analysis of those issues that cannot be met by functional entities, for example, quality of service.

In general, the relationship between an aspect and the two viewpoints is that of a requirement or mechanism. Concepts within an aspect in the SSF provide an abstract, or computational view, which are used to express a functional or nonfunctional requirement that the service must exhibit. In the RSF, however, the same concepts provide a more detailed description, concentrating on an engineering view of the same problem. For example, in the SSF there is the concept of a *creator* that is capable of dynamically creating instances of objects. However, the SSF says nothing of the location or of the accessibility of the creator. Thus, by using the creator a designer is expressing the requirement that objects need to be created. In the RSF, the creator concept is refined, and a rule exists such that any object must have access to the creator. This access is achieved by insisting that each node in the network contains an instance of a creator object.

Domains. The ROOM provides simple mechanisms for defining composition relationships between object instances. Usually, it is required to define other, more complex, relationships between objects; such relationships are often specific to the problem being studied and, hence, not directly found in a general object model. The concept of *domain* can be used to describe such relationships.

A domain is a set of objects, where each is related by a characterizing relationship to a controlling object.[10] Every domain has a controlling object that knows the identity of the objects belonging to the domain.

Domains can be used to define functional, logical, and ownership relationships between objects. Respective examples are: objects providing management, objects within the United Kingdom, and objects owned by IBM.

Interaction, reference, and conformance points. Telecommunication services are specified by applying architectural concepts at various levels of abstraction using the ROOM. At the higher levels of abstraction, the ROOM objects model the logical partitioning of functionality and information that will be used to fulfill the requirements of the service under specification. The choice of object types, interfaces, and object interactions are oriented more toward the problem domain (telecommunication services) than to the resources on which the services will be implemented. Even so, some key objects, interfaces, and object interactions that must be observable in an implementation need to be identified.

At lower levels of abstraction, the object types in the service specification are refined to model the characteristics of the resources on which the service is implemented. The choice of the objects is oriented more toward modeling the functional blocks, interfaces, and interactions between the resources on which the services are implemented. Even at this level of abstraction, some key objects, interfaces, and object interactions that must be observable in the implementation will be identified.

The concepts of *interaction point*, *reference point*, and *conformance point* are basic principles of ROSA that can be used to characterize the observability of interactions between objects. The ROSA definitions of these concepts are congruent with those of the Open Distributed Processing (ODP) standardization initiative.[10]

An interaction point is a location at which objects interact. The location and the services invoked across the interaction point may not be observable in an implementation. A conformance point is an interaction point declared in a standard as a place at which behavior may be observed for the purposes of conformance testing. Thus the services invoked across a conformance point must be observable, and the service designer must include a set of criteria that can be used for conformance testing at this point. A reference point is an interaction point defined for testing to ensure that a specification is compliant with ROSA.

Transparencies. Transparency is the property of hiding the potential behavior of some parts of the system from a particular user.[10] Transparency results from the normal process of abstraction and is found in many areas. An example is the use of the telephone, during which an end user is unaware of the actual route of the connection to another party.

Designing telecommunication services means designing services for a distributed system. In dis-

tributed systems it is a major design issue whether or not to hide the properties of distribution. The term distribution transparency is therefore developed in ODP[11] to handle these properties. A number of transparencies are defined in ODP; the following are those explicitly addressed in ROSA: access, location, migration, configuration, liveness, failure, and replication.

One of the benefits of transparency is that it hides complexity and therefore simplifies service specification and the reuse of existing objects. However, full transparency may result in expensive implementation and overhead and, thus, poor performance. In ROSA a high degree of transparency is normally wanted. But a designer must be able to select the transparency needed in design and have full control of other aspects by turning off some transparencies. For example, although two end users involved in a telephone conversation are unaware of the route that their connection takes, it is important for the purposes of network management for this route to be known.

Conceptual models. Conceptual models formalize the key concepts, separations, and relationships that are crucial to open services and telecommunication systems. Conceptual models are used in the architecture to specify the rules of how its basic concepts should be combined during service specification. The components of the architecture are also categorized such that service designers who apply the appropriate conceptual models will easily find architectural components (concepts and object types) that they can use in their specifications.

Conceptual models exist at various levels of abstraction. At the highest level of abstraction, a conceptual model provides a logical partitioning of the concepts in the architecture, which can be used in service specifications. At lower levels of abstraction, conceptual models structure the relationships between the components that implement the requirements of the design. In ROSA, these components are modeled as objects. Thus, at lower levels of abstraction, conceptual models help service designers identify object types from the SSF and the RSF that can be used to design open services.

Three classes of conceptual models for telecommunication services can be identified. The models in each of the classes formalize one of the following:

- A telecommunication system as a set of services
- The component separations and relationships that are crucial to open services
- The administration domains into which components of a service can be distributed

The conceptual models in the first class are applicable to modeling the environment and allow users to select, initiate, and terminate services. The models in the second class are applicable to modeling the behavior of open services. Models in the third class are relevant for modeling the logical distribution of the components of services.

ROSA architecture

ROSA provides telecommunication services and system designers with techniques for specifying, modeling, implementing, and modifying IBC services. It consists of a set of concepts, rules, and recipes that can be used when modeling, specifying, designing, and implementing new IBC services. Such services are relatively independent of the technology on which they are implemented, are amenable to evolution, and interwork with other services in the IBC network. The architecture has been defined so that services can be specified and designed using the object-oriented paradigm.

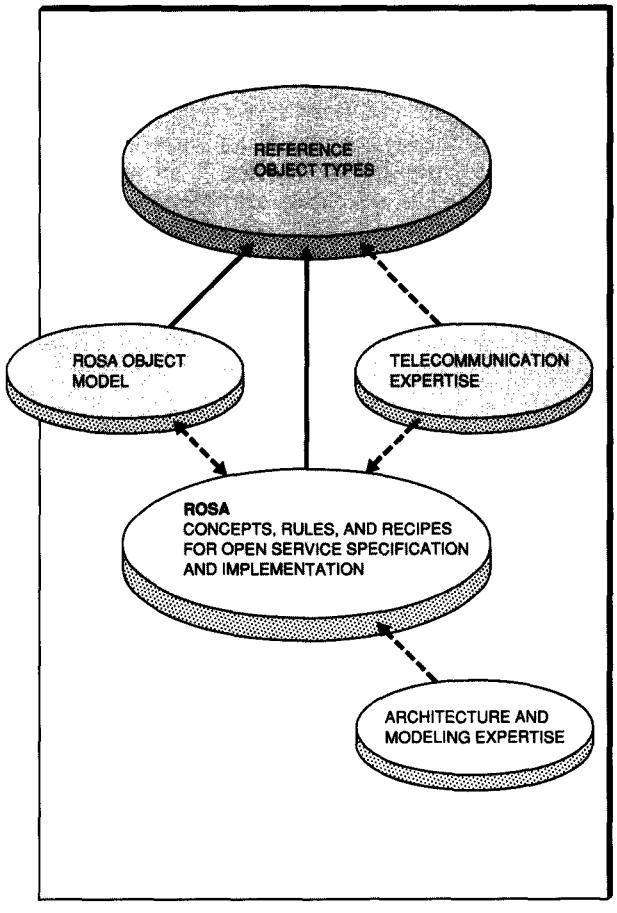
The core of the architecture has been defined by harvesting state-of-the-art research and expertise in distributed systems architecture, modeling, and advanced telecommunication systems architecture, as shown in Figure 3. The ODP standardization initiative,[12] together with the ANSA/ISA architecture,[13] has had an influence on the architecture in the areas of distributed system architecture. The telecommunications expertise has been drawn mainly from current state-of-the-art network architectures such as Intelligent Networks, Integrated Services Digital Networks, and Telecommunications Management Network. Since ROSA is intended for use in defining IBC services, the results from the RACE Functional Reference Model (FRM)[14] project and its associated Customer Service Function (CSF) task[15] have been used to focus the architecture on IBC services.

Service designers using ROSA are required to specify a telecommunication service using the ROOM. The concepts, rules, and recipes of the architecture must thus be capable of being used when constructing service models using ROOM. For this purpose, a set of *Reference Object Types* has been specified. These object types embody the core architecture and conform to the ROOM. Simple architectural concepts such as connections are expressed, logically, as single objects. The more complex concepts, such as entire telecommunication services that are best expressed logically as sets of interacting objects, are defined through structured specifications of object types. The structuring techniques provided by the ROOM are used for this purpose. Architectural rules constrain the permissible interaction between instances of objects derived from these object types. These rules are also followed by the service designer. The specifications of these object types, in COOLish,[5] are included in the architecture deliverable.

The rest of this section presents more details about the components found in both the SSF and RSF viewpoints of the architecture. In the last part of this section is a presentation of conceptual models defined in the architecture that act to structure use of the architecture when designing services. It has been recognized, though, that a designer requires more than just an architecture in defining a service, as certain activities, such as the capture of requirements cannot be supported in an architecture. ROSA has therefore developed a service specification methodology that can be used by designers for capturing requirements, for service analysis, and for service and system design, guiding the use of the architecture where relevant. This methodology is summarized in the subsequent section.

The service specification framework. The service specification framework (SSF) provides concepts for the modeling of the "telecommunication-oriented" aspects of a service. Telecommunication-oriented refers to the features that are found in everyday telecommunication services, such as calls, connections, charging, etc. A resultant specification using the SSF will prescribe what is offered to the users in terms of how it is logically provided by interacting components. Distribution of components around a network is, by default, not a concern in the SSF, although requirements on distribution may be expressed.

Figure 3 The ROSA definition process



Within the service modeling aspect of the SSF, the concept of service control is provided. Service control is responsible for coordinating the activities among the components of a service. In addition, concepts such as directory and charging are provided that help in the task of defining the core functionality of a service.

In access modeling, a user is represented in the system as a user agent, which acts on the user's behalf by coordinating activities to and from services. A user profile captures user-specific information, such as a billing address, whereas a logical terminal can be used to represent the capabilities of the user's terminal. Finally, a service profile can be used to capture a user's specific needs and to tailor a particular service. Together, these concepts provide support for

modeling the features required to access a service and negotiate service parameters.

The transport modeling aspect provides the concept of a trail, which is the abstraction of a multipoint, multimedia communication channel. This concept enables the transport of synchronized streams of end-user information to be modeled. A trail can be decomposed into many mono-media x-connections and synchronizer objects and contains features to synchronize and control connections.

To effectively manage a service, management modeling provides the concepts of a basic managed object and a basic log object. The use of the basic management object concept in services enables different services to be managed, by external management services, in the same consistent way. In addition, concepts to support service-specific management are also provided.

Object support modeling provides concepts for the modeling of the dynamic aspects of a service such as object creation, interobject interaction, and object deletion. A creator object provides for the creation and deletion of objects. Interaction between objects is achieved by first binding the two objects together. In order to support binding, a trader is defined that maintains a record of objects that have offered to provide a service. An object wishing to use a service can query the trader for the identity of such an object.

Finally in the SSF, nonfunctional modeling enables the designer to consider, for the service as a whole or for individual objects that compose the service, what the requirements are in terms of availability, dependability, and performance.

The above concepts in the SSF do not have any knowledge of the actual location of objects, i.e., location and access transparency is provided. In a multiprovider environment, where many organizations will be creating objects and may be interacting with objects in other organizations, there is a need for an organization to control its own environment. The concept of administration domain is provided to model such divisions and provides for the control of external access into the domain, in terms of who and on what objects. In addition, timing concepts are also provided to support the modeling of time-related aspects of a service.

The resource specification framework. The resource specification framework (RSF) provides concepts for the modeling of the "engineering-oriented" aspects of a service. Engineering-oriented refers to the requirements for the implementation of a service in a distributed environment. A resultant specification using the RSF describes the requirements a service will have on a system. The specification does not dictate how to implement, but states what needs to be taken into account when implementing. In addition, what is required, by a system, to support the service is also expressed.

The resource specification framework provides concepts for modeling "engineering-oriented" aspects of a service.

The primary concern when using the RSF is to identify which objects need to be implemented and to model the distribution of objects in a network of interconnected nodes in order to support requirements such as performance and reliability. The RSF will contain mechanisms to achieve some of the nonfunctional properties expressed using the SSF.

The service modeling aspect in the RSF supplies a catalog of resource-oriented components that are used when defining the core functionality of a service. One example is a conference bridge.

The concepts in the access modeling aspect provide a lower level of abstraction of access concerns than found in the SSF. Here, it is acknowledged that some functions are performed on the customer side of a network, and some on the operator side. A user agent can be decomposed into a network user agent, which furnishes a fixed point of control for the network to contact the user, and a customer user agent, which may not exist if the user is currently not using the network. The customer user agent may be mobile, and it provides the current contact point to reach the

user. Both entities are linked by an access channel that enables the network, via a fixed point, to access a possibly mobile user and vice versa.

To enable the more detailed modeling of a single x-connection, transport modeling provides for the decomposition of an x-connection into links that can transport media, connectors that join links, and terminators that model end points of links. This decomposition allows routing and gateway issues, for example, to be analyzed. Note that to include subnetworking, the links can be considered as x-connections and thus support the modeling of recursive network structures.

In management modeling, because of the network-level view taken by the RSF, network management concerns and their implications on a service need to be considered. Since access to TMN objects is provided (enabling fault reporting and alarm handling, for example), the resource management issues of a service can be defined. This access allows the link between service management and network management, with respect to a single service, to be explored.

Object support modeling in the RSF provides for the explicit modeling of, and analysis of, the distribution of objects within a network of nodes. A node supplies basic computing resources (processing, storage, and communications) to objects to enable them to execute. Access to resources is achieved by placing objects into capsules. Capsules are an abstraction of operating system processes. To express the fact that objects should always exist on the same node, for performance, scoping, or security reasons, the concept of cluster is given. A cluster may contain many objects and is a unit of distribution, failure, and storage. Objects within the same cluster always reside in the same capsule, and if one object fails or needs to be placed in storage, all other objects in the cluster will fail or will need to be placed in storage. As a consequence of this visible distribution, the creator and binding concepts found in the SSF are refined to cope with such an environment. On each node at least one creator object and binder object must exist and must always be available to the objects executing on the node.

Finally, the nonfunctional modeling aspect in the RSF provides mechanisms that may be used to solve some of the required nonfunctional properties expressed when the SSF was used. However, not all nonfunctional properties can be solved at this level, and new nonfunctional properties need to be explored as a result of refining the service description using the RSF. Thus, it is necessary to pass on some of these nonfunctional properties to the service implementor. The concepts of replication, storage, and cluster can be used as mechanisms to achieve nonfunctional properties. To achieve high availability, the replication of objects can be used such that if one disappears, a replicate can take its place. By use of storage, dependability can be increased. By use of both replication and storage, highly dependable and available objects can be defined. Performance properties may be partially solved by co-locating objects in clusters and nodes.

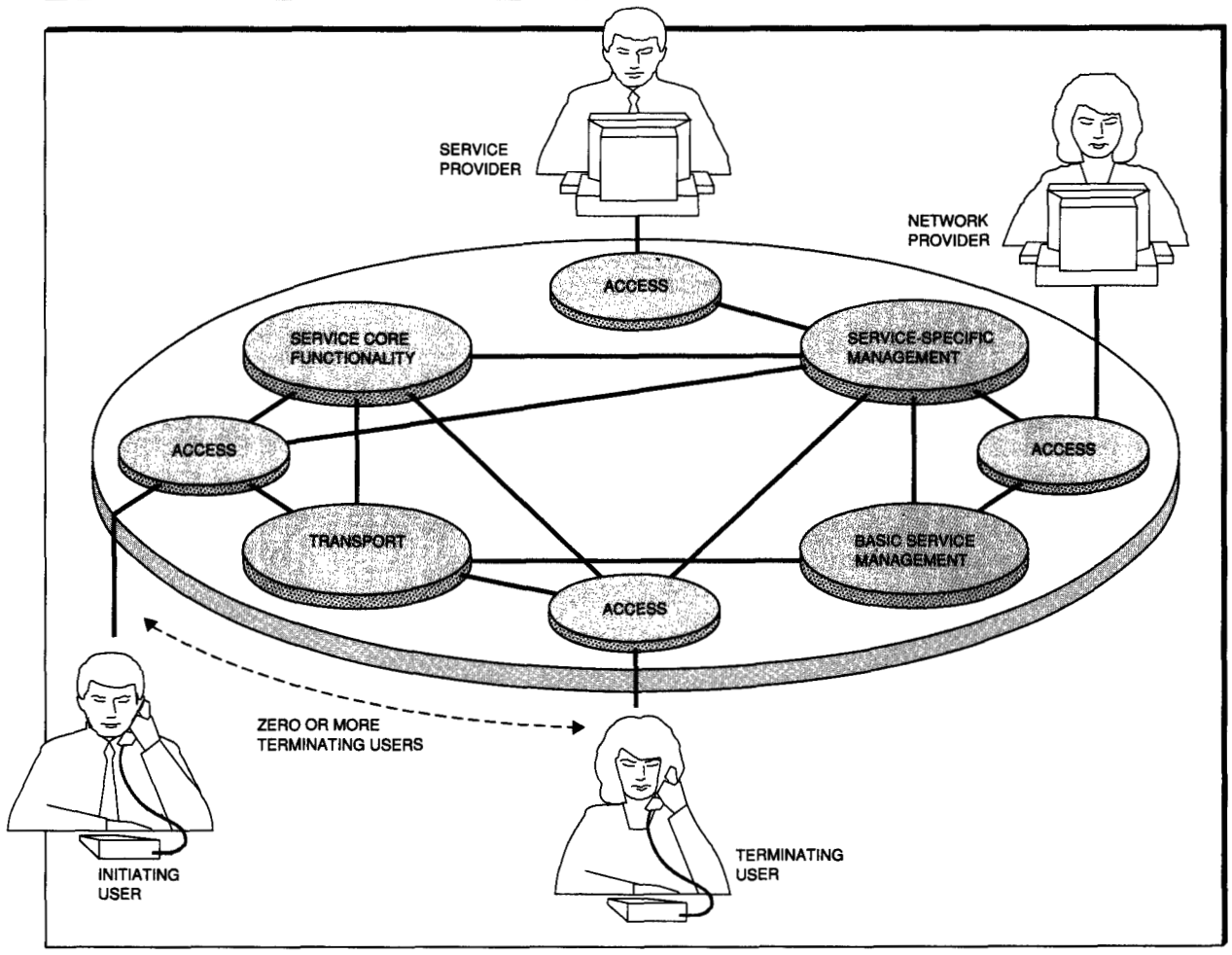
Conceptual models and the architecture. When designing a telecommunication service, from a functional point of view three views of that service can be taken. The first views a service as one of many services in a system. A user may be selecting, accessing, using, and disconnecting to and from many services at the same time. Thus, a service may be viewed in the context of other services. The second views a service in terms of its internal structure, where all the functions provided by the service can be observed. The third views the service in terms of a logical distribution of functionality among the various players involved. Thus, a service may be viewed in terms of who has the responsibility of providing which parts of the service.

Conceptual models have been defined in the architecture to accommodate the above three views for certain classes of services. Only the second form of conceptual model is presented here to illustrate the notion of conceptual models.

The conceptual model in Figure 4 identifies a number of important high-level logical concepts that are relevant to open services. Each of these concepts needs to be considered in a service specification by the service designer.

The *access* concept focuses on service access. The main issues addressed by this concept are the concept of user specialization of a service, supported by the *ServiceProfile* object type in the architecture, the concept of user screening, and the concept of presentation customization, sup-

Figure 4 ROSA service model for user-to-user services



ported by *TerminalProfile*. The service access concept offers components for modeling the behavior of the components required for access by each of the users associated with the service (i.e., end user, service provider, and network provider).

The concept of *transport* addresses the control and characteristics of the information transfer medium between the users of the service. The architecture provides the notion of the *trail* and *connection* to model this conceptual area of the service. Through these concepts, the quality of service parameters of the connection, the route, and the behavior of multimedia connections are controlled.

The concept of *service core functionality* is a high-level one that supports modeling of the functional part of a service. At lower levels, the concepts of conferencing and directories are defined in the architecture to be used in modeling the service core functionality. The components used in this concept are mainly service-dependent, as this is where all the unique capabilities of a service are defined. Therefore, not many components have been defined in the architecture to support service core functionality modeling.

Each service needs a management component. Some of the functions of the management are specific to the service. Other functions are generic to services. This difference is reflected in the split

between concepts required for *service-specific management* and *basic service management* in Figure 4. Service-specific management embraces the concepts required to manage a specific service. These include the charging policies, fault recovery, service suspension and resumption, service configuration, and service performance analysis of the service. In contrast, basic service management embraces the functionality required to monitor the resource usage by the service. The data gathered are used by the service-specific management components to perform their functions. Resource usage data are dependent on the service and could include bandwidth, duration, and the secondary storage usage.

This conceptual model is for a user-to-user service. A number of different conceptual models need to be developed for each class of services that will be specified using ROSA; a similar conceptual model is to be developed for service management services. Also, in each organization, the conceptual models used for specifying the logical implementation of services could be tailored to the modeling requirements of that organization. However, it is easier to relate components of services through their conceptual model than it is through the components that implement the service. Therefore, as long as definitions of conceptual service models are well documented and understood, it should be easy to relate the components of the services at all levels of abstraction. This will simplify the service interoperability problems.

To illustrate the relationship between conceptual models and the architecture, the transport modeling concept in Figure 4 is refined using SSF and RSF components. This refinement will also illustrate one of the possible relationships between the SSF and the RSF components.

The main component of the SSF when modeling a transport connection between parties is a *trail*. A trail is a pathway for multipoint and multimedia transfer of information, made available to the parties of a service. A trail instance is assumed to have the capabilities to transfer end-user information, control the way it is transferred, and inform its client objects of any control-related event occurring during this transfer. Trail instances are created with adequate properties with respect to medium, rate, and flow directions and by enlisting an initial number of parties, all of which can be dynamically modified.

The trail concept provides the designer with the ability to examine the requirements for the transfer of information. However, it is technically not feasible, at present, to buy such facilities. Thus, a trail must be provided by making it out of simpler, less abstract components. The concepts of *x-connection* and *synchronizer* are provided for this purpose.

An x-connection is a pathway for the mono-media transfer of information of type 'x.' By replacing the symbol 'x' with "video," "voice," "data," or "signaling," for example, all different connection types are included in the definition. Thus, an x-connection is an abstraction of all mono-media pathways and can become specialized by using a particular instance of a medium. An x-connection is assumed to be full-duplex. However, applying the principles of the ROOM, specializing x-connections to model half- and single-duplex communication channels can be done if desired.

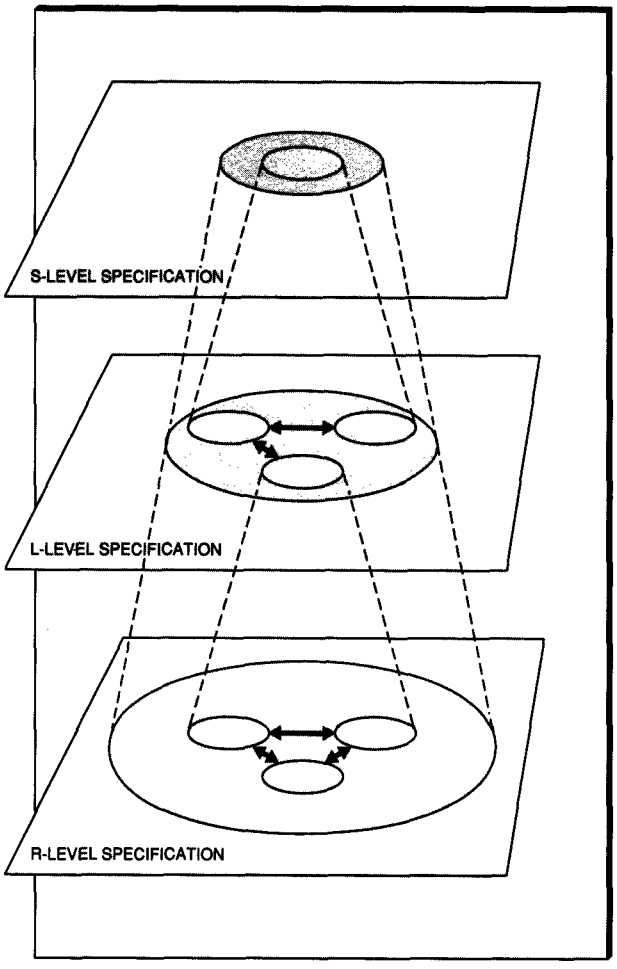
A synchronizer is an object that can receive information from more than one x-connection and will send information to respective x-connections, synchronizing the information in the process according to rules given to it. For example, a synchronizer object can be used to ensure that sound and video information transmitted on separate connections is sent with proper "lip-sync."

From a system perspective, an x-connection is still rather abstract. Thus the RSF concepts allow an x-connection to be decomposed into links, connectors, and terminators, permitting routing, connection failure, and internetworking issues, for example, to be considered.

The ROSA service specification methodology

The architecture must be backed by a methodology, that is, a set of guidelines, that assists the designer of a system of services throughout the overall development process. The methodology underlies most of the notions that constitute the architecture and is essential to the practical development of ROSA-conforming systems. Nevertheless, the architecture is distinct from the methodology, very much like mathematical axioms are

Figure 5 Refinement levels of ROSA specification



distinct from the intuitive reasoning underlying them.

The ROSA methodology provides an approach for service specification in IBC. The focus is on a strategy for service specification in the context of an open service architecture. It is based on the object paradigm that gives a powerful means to define and analyze the services of a system (network). The methodology addresses three principal activities:

- Service analysis
- Service specification
- Service implementation

Elements of the ROSA methodology. The ROSA methodology guides a service designer in using ROSA technology to create "realizable" service specifications. It supplies a framework for specifying the behavior of the objects that model the service through several levels of abstraction, until the service is expressed in terms of objects that are realizable or that describe the capabilities of the resource required in the network to support the service. It provides a strategy for refining usage-oriented service specifications, by which logical and physical aspects of the target environment are considered, to allow for the realization of services in heterogeneous networks with different logical architectures containing systems with different physical properties.

Object-oriented techniques are used in mapping service specifications at different levels of refinement and in incorporating properties from the target environment into service specifications. Services are described by object types that are composed of component object types modeling services of telecommunication and computer systems and resources. In the ROSA methodology it is assumed that these different categories of objects can be specified at different levels of abstraction using the ROOM.

The cone shown in Figure 5 represents a set, S(1) . . . S(M), of specifications of the same ROSA object where specification S(N + 1) refines specification S(N). The cone is thus a log of refinement steps applied to the object type.

Refinement levels of ROSA specifications. The discussion in the following subsections summarizes the ROSA service specification methodology. A fuller discussion of the methodology is presented in *The ROSA Handbook, Release One.*[7]

Service level specification of a ROSA object. A service level specification should make it possible to understand how to use the service repertoire of an object and how to reuse an object type in another object type. External requirements on the object regarding quality of service are also expressed in the object type. A service level specification of an object type is used by a designer of an application using the services that an object instance of that type provides. To develop a service level specification, service aspects (S-aspects) will be considered. S-aspects deal with services, attributes, and behavior of an object from the

user's point of view. Other S-aspects, for example, the constraints the service architecture may impose on the design, are for further study. A specification of an object that has taken all relevant S-aspects into consideration is called a service level (S-level) specification in the ROSA methodology.

Logical level specification of a ROSA object. To structure an object type into component object types that can be distributed on nodes in a network according to a logical (functional) network architecture, logical aspects (L-aspects) must be considered. The logical level (L-level) specification of an object is used to understand how it fits into a logical (functional) architecture. The designer will, in this specification, introduce aspects from a logical network architecture that will constrain the specification. A reference model is one way of expressing constraints in a logical architecture. ROSA, through the SSF, will provide object types that are structured to fit L-level reference models. A designer refines the S-level specification into a logical level specification, taking all relevant L-aspects into consideration.

Realization level specification of a ROSA object. To be able to implement an object type in a given target environment, physical aspects (R-aspects, for example, performance, reliability, and availability) from the target environment must be taken into consideration. The realization level (R-level) specification takes into account the requirements related to implementing a service in a target environment. A target environment is modeled by objects that represent the system and network capabilities that can be used by ROSA objects. In the realization specification, the designer refines the specification of one object type based on properties of a heterogeneous infrastructure (i.e., different target environments). By taking all relevant R-aspects into consideration, the designer refines the L-level specification into a realization level specification. A realization level specification is thus defined by taking all relevant S, L, and R aspects into consideration. It is a specification of an object type from which implementations in a ROSA-compliant target environment can be derived. The RSF will provide concepts, rules, and recipes that can be used at this level of abstraction.

Specification of non-ROSA objects. Specifications of objects that do not conform to ROSA can be reused in specifications of ROSA objects. Foreign object specifications (FOSs) are specifications of such object types. A FOS can be reused in ROSA object types but cannot be further refined in ROSA. A FOS conforms to the ROOM. A FOS specifies an object type for which, during the realization process, an implementation is assumed to exist. An activator object is a foreign object that is used in ROSA to model the behavior of human users.

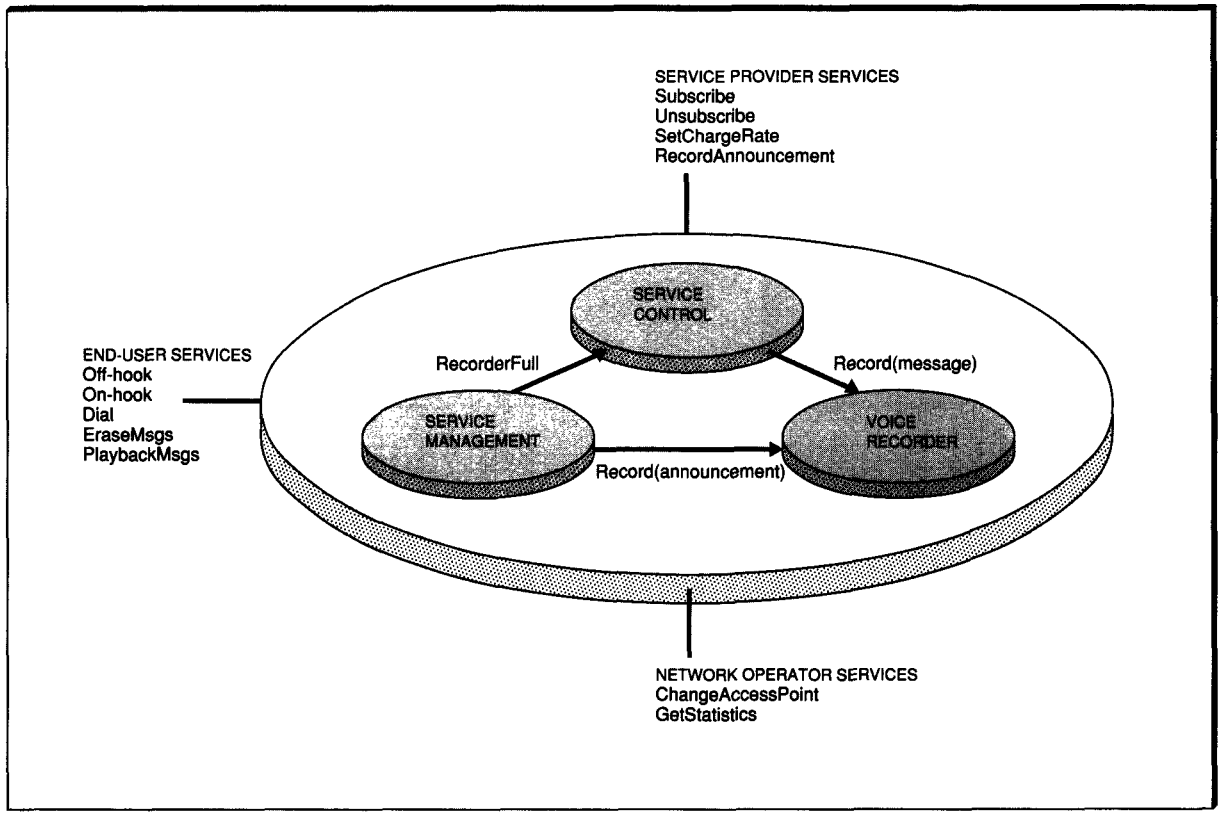
Example

To demonstrate the concepts presented in this paper, the methodology is exercised on a voice-messaging service. A voice-messaging service gives a caller the ability to leave a voice message if the called party does not answer within a predefined number of rings. This concept is an abstraction of both the answering phone and voice mail services, and either of these could be used as an implementation model. This demonstrates that use of high-level abstraction models supports a high degree of reuse, where the reuser is interested in what is provided and not how. Rather than describing a full model in this paper, only a very simplified design is presented.

The S-level description of the service is presented in Figure 6, which shows the services provided by voice messaging to its various users. These services should be defined as the result of a requirements analysis phase. Here, we only present a subset of the services required. In addition, the figure shows a very abstract internal decomposition of the voice mail service. Components at this level should provide enough information to show what the service does and what the main subjects of the service are. In this case the main subject is the voice recorder, and the main concepts to explain behavior can be captured in a service control object and service management object. The behavior of the service can be described by the internal behavior of each object and the interaction that takes place between them. Three possible interactions are shown:

1. Service control will instruct the voice recorder to start recording if no answer to the attempted call is detected.
2. Service management will inform service control if the voice recorder is full and not able to record more messages. The service control has to take this situation into account when deciding what to do.

Figure 6 An S-level description of a voice-messaging service

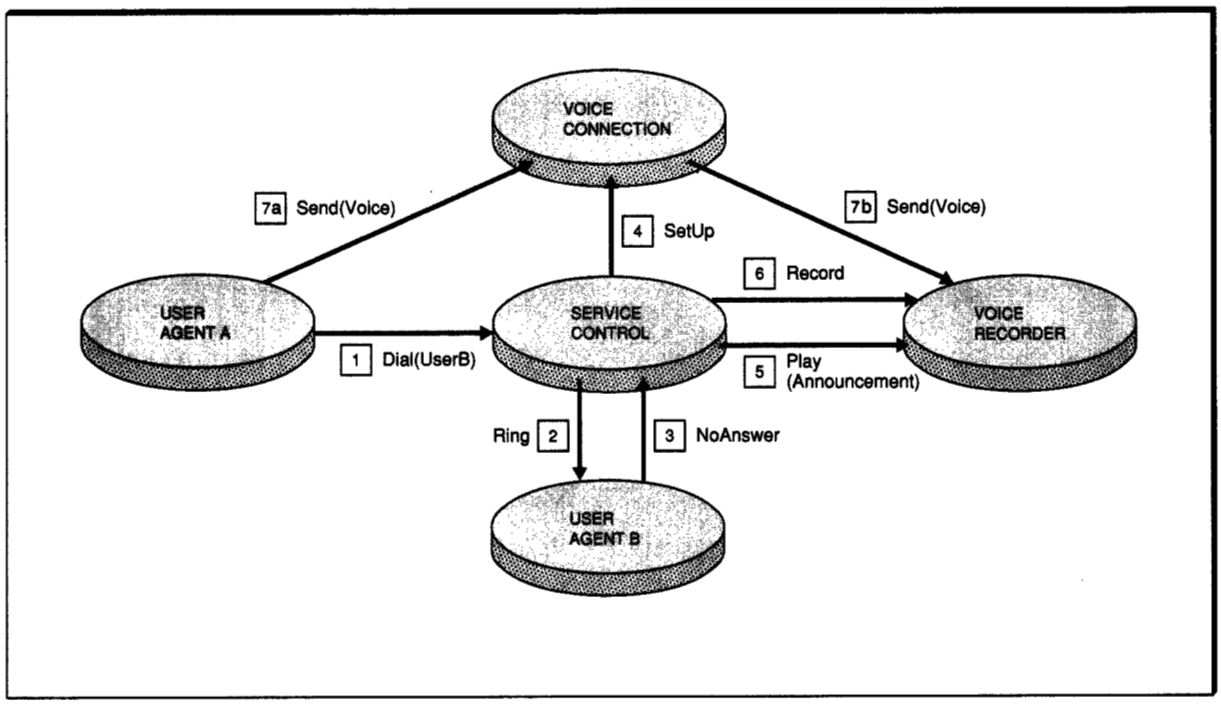


3. The service provider is responsible for providing the announcement message to be played back to a calling user. Thus, service management may instruct the voice recorder to record an announcement.

Having described, in abstract terms, the nature of a voice-messaging service, the L-level description provides a more detailed functional description. This refines the S-level description and enables a full functional description of the service to be developed, but without realization details, such as the distribution of the functions within a network of nodes. Figure 7 shows part of the L-level model; only internal structure is illustrated, and the linkages to the S-level services are not shown. Here, the concepts of user agent and voice connection from the SSF of ROSA are used. A sequence of eight interactions are shown that illustrate the events involved in recording a message. User agent A interacts with the service control to request the establishment of a connection to user agent B. User agent B will accept the request if its user picks up the phone. If this is not the case, user agent B informs the service control of no answer. The service control may then decide to establish a voice connection between user agent A and the voice recorder and interacts with the voice recorder to play the announcement and record any subsequent traffic from user agent A. At this level, nonfunctional issues should be considered. For this example we assume that the voice recorder is highly available. This assumption means that the probability of a voice recorder being unavailable because of node or equipment failure, for example, should be low.

Figure 8 shows a partial R-level description in which the allocation of objects to logical administration domains and to virtual nodes is considered. All objects must be allocated to a domain, but not all objects need be allocated to nodes.

Figure 7 A partial L-level description of a voice-messaging service



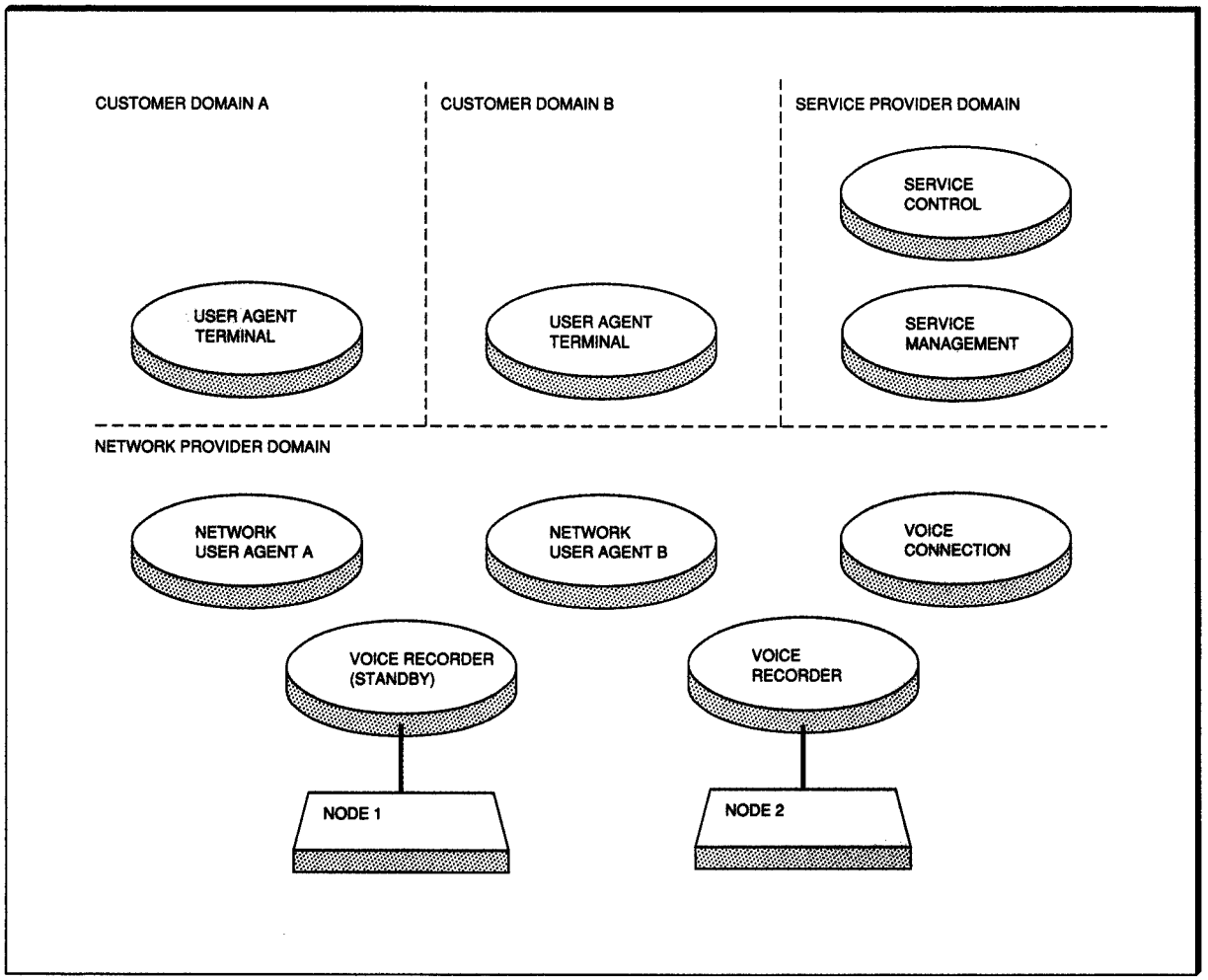
Allocation to nodes is only done to express either the co-location of objects or an explicit distribution of objects. At this level of design, objects not allocated to nodes can exist on any node in the allocated domain. With these allocations, precise requirements can be placed on the system in terms of who has to provide what. The network user agent and user agent terminal concepts of the RSF have been used here to illustrate the decomposition of a (SSF) user agent, which illustrates a division of responsibility. Also note that the voice recorder is replicated in this example and that copies should exist on nodes different from one another. This separate placement should satisfy the availability requirement; if one recorder becomes unavailable, service control can select the standby one. The grouping of objects to nodes and nodes to domain has been directed by a voice mail model. Here the network provider is responsible for providing voice recorder resources, and a service provider is responsible for providing the service control and service management. The allocation of objects could easily be changed to follow an answering phone implementation, in which the service control and voice recorder are co-located with the called user and in which the service provider and end user are one and the same.

Results and conclusions

The ROSA project is producing an architecture designed to be practical and compatible with the major architectural initiatives and the related RACE projects. ROSA expects to produce proposals for an open services architecture suitable for input to the international standardization process. The project has forged a relationship with the RACE Consensus Management project that will guide the ROSA standards proposals to standards bodies such as the European Telecommunication Standards Institute (ETSI), the Comité Consultatif International Télégraphique et Téléphonique (CCITT), and the International Organization for Standardization (ISO).

The development of an open services architecture for RACE is progressing through the use of a service specification framework and a resource specification framework. Some general telecommuni-

Figure 8 A partial R-level description of a voice-messaging service



cation objects have been defined and an attempt is now being made to use these objects for service specification under the control of the ROSA methodology and the ROOM. Rules are being defined for specializing, refining, and combining architectural entities to form object-based templates that model high-level telecommunication services.

An experimental case study will provide invaluable feedback about the integration requirements and the ROSA methodology for specifying services. Such a case study is currently in progress in the project, and its results will be reported in future ROSA deliverables.

An open services architecture as conceived in ROSA will require a service creation (object type creation) environment and an object management library facility before the ROSA approach can be effective. These two important areas are being addressed in the RACE projects SPECS, SCORE, BOOST, and ARISE.[8]

The object-oriented architectural approach to specifying generic service-related components is now being actively discussed in relevant RACE projects. ROSA has spearheaded this change of view. Three of the ROSA partners are now exploiting ROSA concepts and technology in a global

telecommunication system architecture initiative called Telecommunication Information Networking Architecture (TINA).

More recently, Bellcore has been defining the Information Networking Architecture (INA).[16] This architecture has similar scope and objectives to those of ROSA. Indeed, the architecture has followed the same approach as ROSA in applying object orientation and open distributed processing techniques to telecommunication architecture.

Many of the concepts and approaches advocated in the RACE II work plan can be traced to ROSA concepts, and there are opportunities for further exploitation of ROSA in RACE II. The flexible evolutionary architecture that ROSA offers is tailormade to fit the RACE II Integrated Service Engineering, and therefore, the ROSA project will be fundamental in shaping the open services architecture projects in RACE II. The work on an open services architecture for RACE is being continued in the RACE II project CASSIOPEIA.

Acknowledgments

The ROSA project is partially funded by the European Community, and this paper is an edited summary of the collective work of many individuals who are participating in the project on behalf of the following companies: Architecture Projects Management Limited (United Kingdom), British Telecommunications plc (United Kingdom), CSELT (Italy), Ericsson Telecom AB (Sweden), Ericsson Telecom, Ltd. (United Kingdom), FRANCE TELECOM, FUB (Italy), GMD FOKUS (Germany), IBM France, Norwegian Telecom, PTT Research (Netherlands), Synergie Informatique et Développement (France), and TFL (Denmark).

BT Laboratories is the prime contractor for ROSA, and they maintain the archives of the project and distribute the public ROSA deliverables, Malcolm Key being the contact for information.

Cited references and note

1. A program for Research and Development in Advanced Communications in Europe (RACE) funded by the Commission of the European Communities.
2. *Principles of Intelligent Network Architecture*, CCITT Draft Recommendation I.312/Q1201 (December 1991).
3. *Specifying IBC Services Using the ROSA Architecture*, ROSA Workpackage Z, Deliverable 93/BTL/DNR/DS/A/003/b1 (May 1992).
4. *ROSA Architecture, Release Two*, ROSA Workpackage Y, Deliverable 93/BTR/DNR/DS/A/005/b1 (May 1992).
5. *Foundation of Object-Orientation in ROSA, Release Two*, ROSA Workpackage X, Deliverable 93/BTL/DNR/DS/A/010 (May 1992).
6. *ROSA Case Study*, ROSA Workpackage W, Deliverable 93/BTR/DNR/DS/A/011/b1 (May 1992).
7. *The ROSA Handbook, Release One*, ROSA Workpackage Q, Deliverable 93/BTL/DNS/DS/B/012/b1 (September 1991).
8. *Research and Development in Advanced Communications Technologies in Europe, RACE '92*, Commission of the European Communities (March 1992).
9. *Principles for a Telecommunications Management Network*, CCITT Draft Recommendation M.30, Version R1 (October 1990).
10. *Working Document Basic Reference Model of Open Distributed Processing*, Part II: Descriptive Model, ISO/IEC JTC1/SC21/WG7/N6079 (August 1991).
11. *Working Document Basic Reference Model of Open Distributed Processing*, Part III: Prescriptive Model, ISO/IEC JTC1/SC21/WG7/N432 (November 1991).
12. *Working Document Basic Reference Model of Open Distributed Processing*, Part I and Part IV, ISO/IEC JTC1/SC21/WG7/N313 (October 1990).
13. *The ANSA Reference Manual*, Release 01.00, APM Ltd., Cambridge, U.K. (March 1989).
14. *IBC Functional Reference Model Release 2*, R1044-FRM Deliverable 4 (March 1989).
15. *Service Independent Functionalities*, 4th Deliverable of CSF, 44/RIC/CSF/DS/A/004/b1 (February 1990).
16. J. J. Fleck, C. C. Liou, N. Natarajan, and W. C. Phillips, "The INA Architecture: An Architecture for Information Networks," *Proceedings TINA '92,* Narita, Japan (January 1992).

Accepted for publication June 19, 1992.

Abimbola O. Oshisanwo *BT Laboratories, Martlesham Heath, Ipswich, Suffolk, IP5 7RE, United Kingdom.* Dr. Oshisanwo obtained a Ph.D. in electrical engineering from the University of Waterloo in Canada in 1989. He joined the Switched Networks Division of BT Laboratories in 1989, where he is involved in research into software architectures for telecommunication systems. He is also involved in work on service creation environments for Intelligent Network services. He has participated in the ROSA project from its inception through his contribution to the definition of the architecture.

Martin D. Chapman *BT Laboratories, Martlesham Heath, Ipswich, Suffolk, IP5 7RE, United Kingdom.* Dr. Chapman received a B.Sc. in data processing in 1985 from Loughborough University and a Ph.D. in distributed computer systems in 1989 from Oxford Polytechnic, in collaboration with the SERC Rutherford Appleton Laboratories. During his Ph.D. research he developed a service locator and directory service for a wide-area multimedia network. He has been employed at BT Laboratories since 1989, where first he developed integrated user interfaces for communication tools. Subsequently, he has participated in the ROSA project and, more recently, in the follow-up CASSIOPEIA project.

Malcolm Key *BT Laboratories, Martlesham Heath, Ipswich, Suffolk, IP5, 7RE, United Kingdom.* Mr. Key is the project manager for ROSA. He is a chartered engineer who was involved in the design and development of call control in SPC systems since the late 1960s. During the System X design phase he was involved in specifying software for three of the major subsystems and, as a head of group, undertook System X teletraffic studies. During 1987 he was task leader in British Telecom's contribution to ESPRIT 169 (LION). For the last four years he has been researching call control architectures for multiservice networks, and was active in the Customer Service Function project for RACE during 1987, where he was a task leader.

Alvin P. Mullery *RACE European Telecom Projects, IBM Centre d'Etudes et Recherches, 06610 La Gaude, France.* Mr. Mullery received a Sc.B. in E.E. in 1958 from Brown University and an S.M. in applied mathematics in 1959 from Harvard University. He joined the IBM Thomas J. Watson Research Center in 1960 as a research staff member. There he managed projects on a computer with a high-level programming language as a machine language, on an object-oriented operating system, on an expert system for medical use, and on the design of a telecommunications controller, among other projects. He has been at the IBM CER in La Gaude, France, since 1984. There he developed a special operating system for telecommunications. Since 1986 he has been part of IBM France's group participating in RACE, and is manager of the participation in the SPECS and ROSA projects, and in the RACE II SCORE project among others.

Jacques Saint-Blancat *RACE European Telecom Projects, IBM Centre d'Etudes et Recherches, 06610 La Gaude, France.* Mr. Saint-Blancat received his degrees from the Ecole Nationale Supérieure d'Ingénieurs en Electronique, Electricité, Informatique et Hydraulique de Toulouse, first in electronics engineering in 1968, then in computer science in 1969. In 1970, he joined IBM in La Gaude, where he developed languages and software simulators for signal-processing equipment. In 1975, he left for a three-year assignment at the IBM UK International Centre to work on the support and development of the IBM DCS (Data Centre Services) network. Back in La Gaude in 1978, he worked with Al Mullery on an operating system for an 801-based communication controller and then worked for six years on the development and support of several software tools for telecommunication products. In 1986, he was appointed coordinator of the local software and telecommunication program. Since 1988, he has been part of IBM France's RACE group, participating in RACE I's ROSA and SPECS projects and now in RACE II's SCORE project, which is a follow-up of both ROSA and SPECS.

Reprint Order No. G321-5493.