The ODO Project: a Case Study in Integration of Multimedia Services

Mike Rizzo, Peter F. Linington, and Ian A. Utting
Computing Laboratory, University of Kent, Canterbury, UK

August 1994

1 Introduction

Recent years have witnessed a steady growth in the availability of wide-area multi-service networks. These support a variety of traffic types including data, control messages, audio and video. Consequently they are often thought of as integrated media carriers. For example ISDN is often spoken of as an integrated voice and data network. To date, however, use of these networks has been limited to isolated applications which exhibit very little or no integration across the services that they provide.

Rather than talk about the physical integration of networks at the communications level, this paper focusses on integration of services at the application level. Exactly what we mean by this is best illustrated by an example.

1.1 Illustration

Consider a solicitor by the name of Mary White making a call to her research company, A&B Legal Research. A computerised operator answers her call:

"Good morning. This is the A&B legal research company client helpline service. Please use your telephone keypad to enter your client identification number."

Mary keys in 89824123.

"Hello Mary. I note that Charles Smith usually handles your enquiries. If you would like to speak to him press 1, if you would like to leave a message for him press 2, if you would like make an appointment to see him press 3, otherwise press 0 for human operator assistance."

Mary presses 1. Charles is sitting at the workstation in his office. The workstation is equipped with a microphone and headset. An incoming call indicator on his screen flashes, and informs Charles that Mary White is calling. Charles clicks on a button to accept the call. The screen immediately displays the company records for Mary White.

"Hi Mary. How are you?"

"Fine Charles, thanks. Listen I was wondering whether you'd come up with anything on that Jones case?"

Charles glances at the screen and notes that a colleague had recently attached a note on the Jones case to Mary's record.

"Ah yes. We've got something for you. Would you care to hold the line for a minute."

Charles clicks on another button to put Mary on hold. During this time she is connected to a music service which allows her to select a track using her telephone keypad. Additionally, any other incoming calls for Charles are diverted, except for those originating from high priority sources determined by Charles himself.

Charles starts to read the note but is interrupted by the call indicator again. This time its his wife. Charles clicks on an 'accept' button to put the call through.

"Hi Charles, I'm in a bit of a hurry. Could you pick up Tom from school today. Something's come up and I won't be able to make it home till dusk."

"Sure hun. And leave dinner to me too. See you later."

Charles then finishes reading the note attached to the Jones case and switches back to Mary's call.

"Hi Mary. Apparently there are some complications. You'll have to speak to our criminal law expert Richard Hayes. I'll transfer you."

Richard Hayes happens to be having a break in the company coffee room at the time. The phone rings. Joe, who is having coffee with Richard, picks up the receiver.

"There is call for Richard Hayes. If he's here, please press 1, otherwise hang up."

Joe presses 1 and passes the phone over to Richard.

"There is a call from Mary White, routed through Charles Smith. To accept it press 1, otherwise press 2."

Richard presses 1 and engages in a conversation with Mary.

1.2 Some observations

In this example, Mary uses the telephone keypad to enter data and to control the behaviour of services. The voice from her telephone terminal is carried over through the computer network and, likewise, voice originating from computer-based services reach her terminal through the telephone network.

Regardless of whether she is communicating with a machine-driven interface or with another person, Mary is always connected to a service. In the course of the call she is switched from one service to another several times. Services associate data with calls, and may pass this data on to other services as the call is switched. For example, the machine-driven operator that answers Mary's call retrieves her records and passes them on to the service that allows Mary to talk to Charles, which also allows him to view her records, put her on hold, and transfer her call. Mary's records are later transferred to the service that allows her to talk to Richard, as is demonstrated by the fact that her name is announced to him before he accepts the call.

Whenever Mary is connected to another person's terminal, this terminal is determined by means of a location service that tries to establish where in the building the required person is. Thus the phone automatically rings in the common room where Richard is having his coffee-break.

Neat as this integrated system might appear to be, there are a lot of potential problems and conditions that have to be catered for. What if Richard doesn't want to be interrupted during his coffee-break? Or what if he isn't in the building at the time of Mary's call? Should Mary be told to try again later? Or perhaps a call between Richard and Mary could automatically be set up when Richard becomes available?

Charles' wife is put through to him immediately, but is this desirable for *any* incoming call, given that Charles is already dealing with a client at the time? If a second client calls, it would probably make more sense to put the call in a waiting queue. Or perhaps the system could offer to call the client back automatically when Charles becomes available? Whatever the course of action taken, the underlying point here is that Charles must have a way of indicating that some people are to be given special treatment. Apart from the identity of the caller, this treatment might also depend on such factors as the time of day, what Charles is doing at the time, and where Charles is located.

1.3 The ODO project

One can identify numerous other potential gains and pitfalls in this example. The ideal framework for the kind of integration we are describing will provide mechanisms that maximise the flexibility and functionality available to the user, and at the same time cope effectively with the various problematic conditions that may arise.

Our hypothesis was that an attempt to construct such a framework should be based on experience in designing and developing a real system. This paper reports on the ODO[1] project, a case study in integration of office speech and data services. The objectives of this study were to investigate modelling, interfacing and programming techniques which would maximise the integration of voice and data in the system. Care was taken to adopt a general approach which could be utilised in other application areas, a few examples of which are described towards the end of this paper. The study was conducted in the context of the ISO/ITU-T Open Distributed Processing Reference Model (RM-ODP) [Lin92] and prototype components were implemented using ANSAware [Arc92] as an engineering platform.

2 The Open Distributed Office

The ODO project focussed on a number of services which were deemed interesting with respect to speech and data integration. Figure 1 lists these services, illustrates the kinds of devices used to access them, and gives examples of the sort of people that would be expected to use them.

Conversational services enable a caller to engage in a conversation with a callee over some appropriate communications system. Messaging services[2] allow users to leave messages for others as well as retrieve their own. Diary services are used to maintain and browse users' calendars of appointments. Directory services enable recording, maintenance and retrieval of knowledge about various entities. Location services[3] are used to locate individuals using any means available eg an Active Badge system [WHFG92]. Finally, alarm services allow service requests to be made at some specified time in the future.

Some examples of integration in the context of the above services are:

- the location service uses an individual's diary as one source of location data,
- the location service consults a directory to obtain information about the target individual which may be used in the location process,

[1] Odo was Bishop of Bayeux and (at least) half brother of William the Conqueror. Named Viceroy and Earl of Kent by the Conqueror, he was given especial responsibility for communications with William's headquarters in France. Odo is also an acronymn for 'Open Distributed Office'.

[2] See [RLU94c] for more information on the voice messaging sevices.

[3] See [RLU94b] for a more detailed description of the location services.

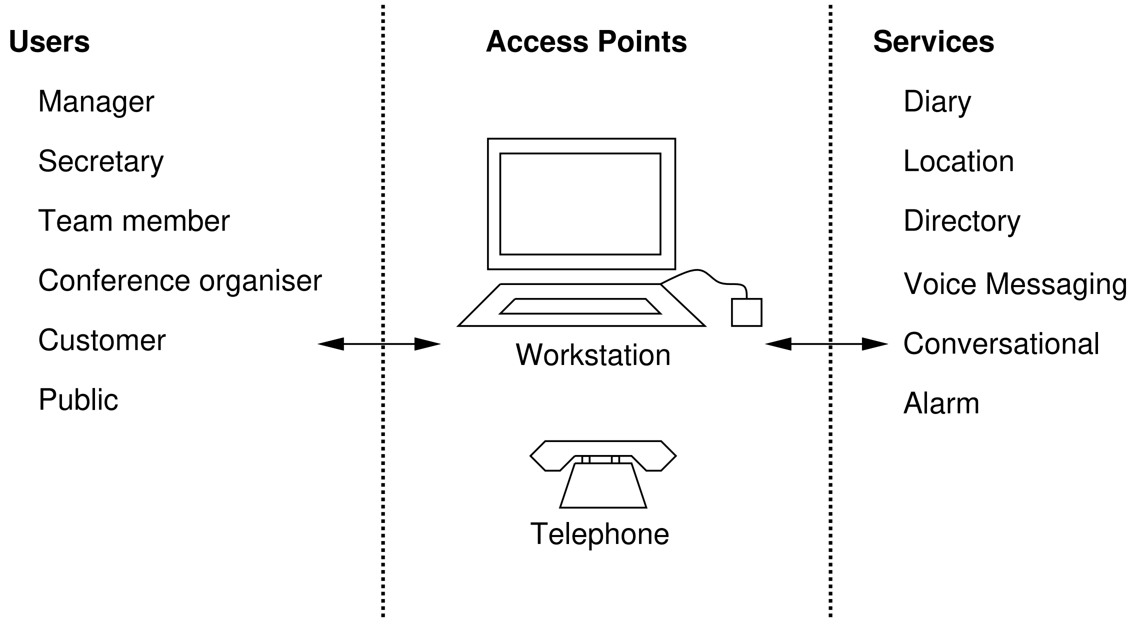

Figure 1: The Open Distributed Office

- the conversational service uses the location service to determine the nearest access point to the callee,
- the diary service is used to schedule a call (appointment over the phone) which is eventually set up via the alarm service.

Later in the project, additional services related to call management were identified. These included: an operator service to handle incoming calls, user agent services to co-ordinate service switching in the course of a call, and a selection service that enables a caller to select an alternative course of action should a service invocation fail.

ODO services were intended for use by all categories of office staff, possibly customers and colleagues from other organisations, and even ‘ordinary people’ such as friends or family. Outside the office, access would normally be via the phone, this being a ubiquitous device. Inside the office, access would more likely be via a graphical workstation equipped with audio hardware[4]. Consideration was also given to federation of ODO systems, allowing for seamless integration across administrative domain boundaries.

3 Approach to integration

Two separate levels of integration were recognised, termed the binding level (or connection level) and the service level.

At the binding level, the telephone and computer networks are integrated in two ways:

- information exchange: telephones and computers can exchange voice, and possibly keypad events (through touch-tones) when available;

[4]Although a phone could be paired with a workstation where such audio hardware was not available.

- control interface: the telephone network is controlled via a computer-based interface which hides the distinction between telephones and workstation-based voice terminal devices.

The result is a virtual communications network which consists of both telephones and workstations. We stress that this has nothing to do with physical integration of the networks in the sense that, for example, the same switch is used for both voice and data. Such integration is essentially resource sharing to cut costs, and although it could be used together with the kind of integration we have described, it is not a pre-requisite.

This virtual network is then utilised by a variety of services at the service level. Such services essentially associate data with a call, and make decisions based on this data and possibly input conveyed through the call.

Opportunities for integration also exist amongst these services. For example, one service might use another in order to carry out some sub-task, or acquire some information. A service might even offer another service in its place when it cannot satisfy a request. Integration of this kind has implications in data transfer, security, and policy enforcement, amongst other issues.

The next section concentrates on the binding level. The sections following it discuss the service level and deal with some of the related integration issues in greater depth.

4 The Binding Level

The binding level is based on the ODP streams model. In this model, a continuous stream flows amongst two or more *stream interfaces* under the control of a *binding object*. In ODO, we refer to stream interfaces as *end-points* and a binding is limited to an association of exactly two such end-points[5].

Each end-point is capable of producing and consuming an audio flow, and producing and consuming a sequence of keypad events. In other words, an end-point is an abstraction of any device which:

- is capable of transmitting and/or receiving an audio stream, and
- is optionally capable of generating and/or handling keypad events.

When two end-points are paired in a binding, information exchange commences so that audio and keypad events produced by one end-point are consumed by the other and vice-versa. This information exchange continues throughout the lifetime of the binding, which lasts until the binding is dissolved or transferred via the binding's *control interface*. As implied, this interface contains two operations: a *dissolve* operation can be used to terminate the binding, whilst a *transfer* operation can be used to replace one end-point by another to produce a new binding (see figure 2).

Our prototype implementation supports three kinds of end-point, namely: ordinary

[5] We decided not to support multi-party calls for the sake of simplicity. However, current work following on from the ODO project is taking such calls into consideration and generalising ODO concepts where necessary.

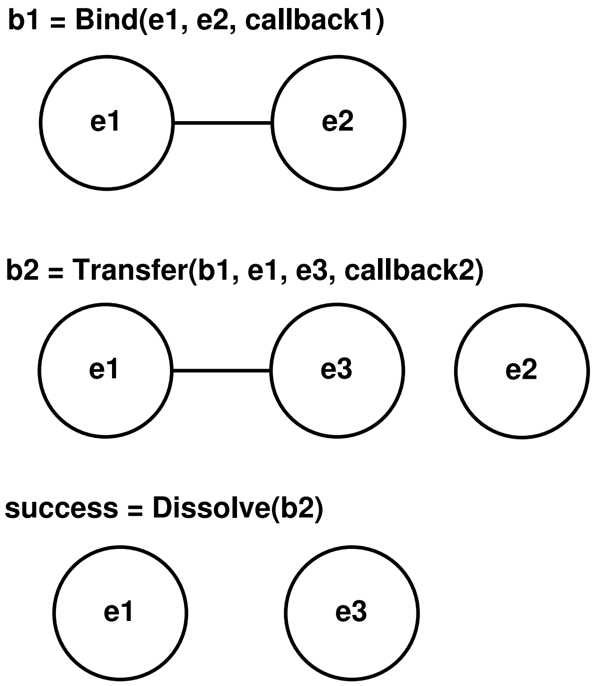

Figure 2: Binding operations

POTS phones[6], workstation audio hardware[7] , and machine-driven interactive services (such as the automated operator described in our opening illustration) [8]. Phone and workstation end-points are collectively referred to as user end-points because they allow human users to communicate with the system. Service end-points, in contrast, are driven by service user interfaces. All bindings must involve at least one user end-point.

5 The Service Level

The binding level provides an integrated voice and data network, which allows information exchange and switching control. Above this level, we package different applications into services. At one extreme, a service might be purely data-oriented. At the other it might exchange voice and exercise control over the switch. In between, some services may simply be capable of receiving, storing, and forwarding voice segments.

[6]The telephone and computer networks are bridged by means of an interface that sits between a PC-bus and the telephone local loop. This interface enables information exchange as well as some limited control over the telephone switch. Direct control over the switch would have been preferable but was not possible. Consequently a number of compromises had to be made with respect to phone-to-phone bindings; once such a binding is set up, control over the connection has to be relinquished. It is therefore not possible to dissolve or transfer a phone-to-phone binding via its control interface, nor is it possible to detect whether such a phone has hung up or initiated a transfer.

[7]Voice on the computer network is encoded using the well-known PCM encoding scheme. Workstations and X-terminals on our network are equipped with hardware that recognises this format directly.

[8]These services were developed using VitKit (Voice Interaction Toolkit)[RLU94c], an object-oriented toolkit for building telephone-based user interfaces.

A variety of issues were studied in the context of service integration. These include knowledge representation and dissemination, security, federation, scaling, policy enforcement, user interfacing and call management.

In the remainder of this section, we highlight the kinds of problems that have to be dealt with. Subsequent sections discuss some of these issues in greater detail.

It must be emphasised that ODO was not intended to be limited solely to localised installations, but was designed to work as a large, wide-area distributed system. A running system would typically span several organisations, each of which might require some degree of flexibility with respect to the above list of issues. Thus an ODO system could be partitioned into a number of *administrative domains*, each responsible for all aspects of the management of its services. Consequently, one challenge was to give such domains some flexibility in defining and enforcing their policies while at the same time allowing seamless, inter-domain integration of services across domain boundaries.

Integrated services often need to exchange data pertaining to some entity e.g. a person or a location. There is a potential problem here, in that different services may use different and incompatible abstractions for the same entity, particularly across administrative domain boundaries. This also means that there is often more than one way of an identifying an entity. For example, five ways of representing an individual's office include: a room number, an extension number, an electronic mail address, a person's name, and a network host name. A mechanism for reconciliation of views is therefore required to enable knowledge sharing and to deal with the identification problem.

Another problem is related to dissemination and uncontrolled replication of knowledge. An individual's telephone number might be extracted from one directory, communicated by word of mouth and entered into a second directory. A change in the first directory will result in an inconsistency and services using both these directories might eventually clash if used together.

Knowledge dissemination is not limited solely to transfer across directories; any service may communicate knowledge gained about some entity to another service. For example, if in the course of interacting with a service it is established that a caller speaks French, this might be communicated to conversational services so that the caller is automatically routed to a French-speaking operator, should he require human-assistance.

Security is another tricky issue, particularly with respect to access control. Flexible mechanisms are required which do not make assumptions about the nature of clients (a particular problem with the 'access control list' model). More specifically, it should be possible to have control over access rights for users from remote administrative domains just as for local users.

Autonomy is the property associated with the ability of an entity to establish its own policies with respect to some service. In ODO, care was taken to provide for autonomy at the user level as well as the organisational level. Typically, an organisation level policy usually consists of a set of contraints for user level policies and one such user policy to be used as a default in the absence of a user-defined one. For example, a user's location policy might describe the best approach for a location service to take when attempting to locate that user, replacing the default location policy set by the organisation, but still working within any constraints imposed by the organisation's policy.

Multiple instances of a service might be made available for performance-related or autonomy-related reasons. It is often desirable to federate such instances so that they appear as one unified service to clients. Federation can significantly reduce the programming effort required to access such a distributed service.

In ODO it was desirable to allow access to services via two kinds of devices, namely telephones and workstations with audio capabilities. The user interfacing requirements for these devices are clearly very different and it was therefore important to separate user interface components from service components. In particular, care was taken to ensure that services did not make unneccessary assumptions about the nature of the interfaces used to access them.

Finally, call management is responsible for the sequencing of connections and disconnections to and from services, potentially under user control. One key requirement for any call management model is the ability to transfer relevant knowledge across services. For example, if the location service returns successfully with the target callee on the line, then a subsequent invocation of the conversational service should connect straight to this same line, without the callee having to put the phone down and pick it up again.

The next three sections expand on some of the issues highlighted here. Section 6 focusses on knowledge representation and dissemination, section 7 describes the approach taken to security, and section 8 discusses service access and call management, touching on some of the other above-mentioned issues in the process.

6 Knowledge representation

6.1 Attribute lists

In ODO, knowledge is represented in the form of *attribute lists*. An attribute list is simply a list of name-value pairs and is similar in concept to the X.500 directory entry. Attribute lists are not necessarily stored in directories however. They might exist for short periods of time e.g. throughout the duration of a call, or be used to pass knowledge from one service to another.

Attribute lists are used to describe different classes of entities such as locations, users, and organisations. For a particular class, a service might recognise some attribute names and use their corresponding values. Other attributes might not mean anything to a service, but this does not prevent them from being passed along some service chain until they may eventually become useful.

Of course it is important that cooperating services agree on the meaning of specific attribute names. To this end, a number of attribute naming conventions were drawn up for each entity class.

The attribute list scheme gets round the knowledge exchange problem by using a flexible, universal representation for abstractions instead of a rigid one per service. Additionally, it is possible to introduce new services which recognise new attributes without having to modify any other services or update any stored representations.

6.2 Directories

A directory is a repository for knowledge that persists beyond the duration of a call or interactive session. Essentially it consists of a list of entries, each containing an attribute list. Editing and lookup operations are provided.

ODO does not use the concept of a single global directory view as in X.500. Instead, several directories are allowed to store potentially different information about the same entities. This allows users to maintain personal directories which might contain information that is not available in a public one. An entry in a personal directory is normally only accessible by the directory's owner and typically contains information which the owner has specifically obtained about the represented entity for his or her own personal use. The notion of a personal directory is extremely useful for access control as will be explained in section 7.

6.3 Dissemination

Having seen that the approach to persistent representation of knowledge exhibits a high degree of distribution, mechanisms are needed in order to disseminate knowledge amongst components.

In ODO a rather liberal philosophy was adopted, namely that any available mechanism be allowed, regardless of the extent to which the system could exercise any control. This means, for example, that it is perfectly acceptable for a directory to be updated manually with information that was received by word of mouth. At the other extreme, a directory might also be updated by a service with information obtained more reliably, say from a remote public directory. A half-way approach involves transfer of knowledge by embedding attributes in electronic mail messages and having them extracted automatically into a receiver's directory at the other end (at the receiver's discretion). These three mechanisms are all supported in ODO.

Of course this unconstrained approach can lead to problems of inconsistency. Consistency cannot be maintained automatically unless the system is in total control of knowledge transfer, which quite evidently is not the case here. It is worth noting, however, that this also applies to the 'real world'. For example, a telephone subscriber's number might change resulting in an appropriate update to the network operator's records. Telephone directories, however, might not be updated for several months, and pen-and-paper personal directories might take years to reflect the change, if ever. Upon failing to communicate with the intended target, the caller might resort to another information service, such as telephone enquiries or asking a friend, to get the new telephone number. Using more than one service may very well result in the caller receiving conflicting values. For example, the friend might return the target's mobile phone number, mistakenly thinking it to be his home number.

These kinds of problems can also arise in the ODO context and there is no foolproof mechanism that can get round them. However, a number of things can be done to alleviate the inconsistency problem and to make the resulting anomalies easier to work with.

First, attribute values can be time-stamped so that in case of conflict the most recent value can be used before resorting to values with older time-stamps. It is up to services to time-stamp values correctly and there is clearly an element of trust involved. Additionally, when communicating values through uncontrolled channels, it is up to the supplier of the information to pass on the timestamp along with each attribute value.

Secondly, from a user interaction perspective, user involvement in resolving inconsistency problems should be delayed as much as possible, unless the user's policy dictates otherwise. A user's lookup policy might indicate alternative sources of information when values in the user's own directory are missing or prove to be incorrect. These alternative sources may be used without the user ever knowing it.

6.4 Reconciliation of views

Because different services use different abstractions, it may not always be immediately obvious that two attribute lists are actually abstractions of the same entity. For example, the attribute list

$$< (room, S25), (ext, 7697) >$$

might represent the same location as

$$< (host, xtmr3), (occupant, mr3) >$$

but there are no attributes in common via which a connection might be made. There are cases where, even if there are attributes in common, it is not clear whether the attribute lists refer to the same entity.

Directories are useful tools for reconciliation of views. They can be used as inference tables to establish equivalence relationships amongst attribute lists. Any organisation running ODO would be expected to maintain directories of all its locations and users. Using the above example attribute lists as lookup constraints, it may be possible to infer that they are indeed the same location i.e. if both searches yield the same unique attribute list then they represent the same location. This approach is used in the ODO location service to combine knowledge obtained from a variety of location mechanisms.

6.5 Identification

Different services might identify entities in different ways. In ODO, there is no difference between representation of an entity and a reference to an entity. Both are held as attribute lists which effectively represent those characteristics known about the entity.

7 Security

In the ANSA security model [Arc93], the two fundamental concerns are *authentication* (the ability of an object to verify the identity of others with which it is interacting) and *authorisation* (the ability of an object to restrict access to legitimate clients by requiring them to satisfy certain conditions before service can be supplied). Other aspects of security such as data integrity and confidentiality are dependent on these two.

Authentication is primarily concerned with providing security against the would-be hacker. Essentially it serves to prevent maliciously invented objects from masquerading as genuine ones. There are well known techniques for authentication and it was assumed that these could easily be incorporated into the ODO system [9].

Authorisation is more directly relevant to actual users of the system as it determines what clients can and cannot do on their behalf. Special consideration was given to authorisation in the design of ODO with respect to the storage and retrieval of *access certificates* or *capabilities* as will be described later in this section.

[9] Note that the problem of authenticating a user is a different one and is really equivalent to authorisation of a client to obtain that user's data from an administration server.

7.1 Server-centric approach

A server-centric approach to security was adopted in ODO; every service provider in the system is responsible for its own security and there is no central authority for regulating access to services.

In a server-centric model, a service provider retains control over its own security policy. It is responsible for both generation and validation of control data which it may require for authorisation purposes i.e. security requirements are declared as part of the computational object providing the service.

The decision whether or not to provide service to a client is taken at the time of service invocation. This means that from the time a client is given the authority to use a service to the time it actually tries to make use of that authority, the policy may change so that the service request may be refused. This is called *revocation* of authority.

This model is suitable for large-scale distributed systems spanning multiple security domains. Each domain can determine and enforce its own policies and can change them at will with immediate effect.

Authorisation is achieved by requiring a valid *capability* to be presented to the server in order to gain access. The capability contains encrypted control data upon which the server bases its decision to grant or deny service. A capability is always generated by the server with which it is intended to be used and may be passed from object to object, eventually returning back to its creator when service is called for.

The syntax and semantics of the control data contained in a capability are determined by its creator, thus enabling each server to implement any security policy it desires. This will never cause problems with other co-operating objects as these simply retain capabilities for later presentation to the same originating server which will be capable of validating and interpreting it.

7.2 Access control policies

There are several options available to implementors of a security policy for a server. Typically a capability's potential may be restricted to a subset of the operations supported by that server. Other variations include limiting the number of times a capability may be used (n-shot capabilities), or restricting use of a capability to particular times in the day.

It is important that any data pertaining to access control policy, such as usage counters and time restrictions, should not be embedded in the capabilities themsleves, but should be held and maintained in a table by the server. The capabilities themselves contain references to entries in the table. This facilitates revocation of authority and also makes it easier to change restrictions imposed on capabilities.

7.3 Storage and retrieval of capabilities

In ODO, capabilities are used to control access to a variety of objects through services. A user typically possesses all-powerful capabilities for access to his own resources (diary, personal directory, etc) and possibly grants less powerful capabilities to other users. A user might also choose to make the same capability available to a wider audience.

A capability may be held as an attribute in an attribute list. In this way, directories can be used for the storage and retrieval of capabilities. Additionally, it is possible to pass capabilities from one service to another, along with other information, using attribute lists.

To illustrate these concepts, consider a user Jones making a request to call user Smith. Jones makes his whereabouts known to the conversational service and also passes on everything he knows about Smith i.e. the attribute list in his personal directory that corresponds to Smith. The conversational service checks its own directory to see if it can add any information about Smith, then generates a request to the location service, again passing all information accumulated thus far.

The location service might look Smith up in its own directory to see if it can obtain any more information. In particular it may find Smith's location policy, which might be something on the lines of "check my diary first, and failing that try my office". Asking Smith's diary for a location, however, requires a capability. Luckily Jones is in possession of such a capability, and this was passed along in the attribute list with the original request.

This 'cumulative' approach to gathering information can be rather useful. Smith might place a diary capability in the conversational service directory so that anyone wishing to call him would always be able to check his diary for a location. This capability would not, however, be available for uses other than looking up Smith to hold a conversation. Placing a capability in the location service directory would mean that Smith's diary could always be consulted for location purposes, whatever the subsequent course of action may be.

7.4 User Authentication

Given that capabilities are stored in directories, and that directories themselves require presentation of capabilities in order to be accessed, a mechanism is needed to give a user a 'starting set' of capabilities. In ODO, a user's capabilities that pertain to his own resources are stored in an administration directory. A user authentication service (which is always in possession of an appropriate capability for the administration directory) can make these capabilities available to a user upon successful authentication.

8 Service access and Call Management

ODO services may be accessed directly or in the course of a call. For example, a user might use a workstation-based client directly to access his voice messages. On the other hand, such messages are likely to be recorded during the course of a call. We refer to these modes of access as *direct* and *call-oriented* respectively.

In keeping with RM-ODP, services are realised by means of objects and made available via object interfaces. From a computational viewpoint, the relationship between services and objects is *M:N*. A service might be provided by a set of co-operating objects, and the same object might provide several services. Similarly, the relationship between services and object interfaces is also *M:N*.

In direct access mode, a user invokes a client which provides a user interface and generates operation invocations on the service interfaces. For example, a user might use a diary viewer to browse through his or her appointments. The user interacts with the service for a while, and eventually terminates the session by shutting down the client. Not all services are accessible via direct-acces mode; for some services eg conversational, call-oriented mode is obligatory.

In call-oriented access mode, a client (known as the call initiator, or caller) passes a request to a special service known as the *call manager*. The call manager analyses the

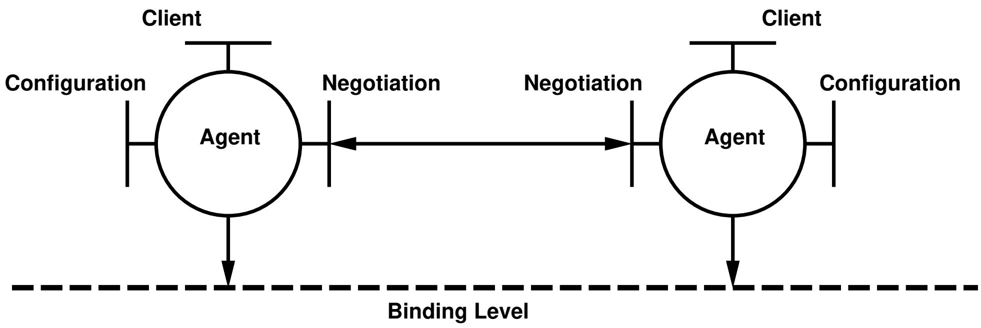

Figure 3: Agent Model for Call Management

request and connects the caller to an appropriate service. This service may eventually signal completion back to the manager, or may return a new request which determines the next stage in the call.

The progress of a call is influenced by a call request at one end, and a callee policy at the other. In ODO a lot of flexibility is allowed for both of these and the resulting conflicts and interactions may be quite complex.

8.1 Call management model

Call-oriented access is based on a negotiating agents model (see figure 3), whereby users are represented by agents which negotiate a course of action that is acceptable to all parties concerned. A user passes a request to their agent, which in turn contacts the agent of the target party in order to set up the call. The target party may be unable to participate for some reason or other, in which it case it may offer one or more fallback behaviours. The calling agent might agree to proceed with a fallback or could alternatively try to pursue an entirely different course of action.

A user's agent may take decisions based on: interaction with the user, a user-defined call management policy, a history of the user's behaviour, or possibly a combination of these. Several different kinds of agents may be employed to satisfy different user requirements; decision-making is therefore agent implementation-specific, as are the methods used to control the decision-making process. The only constraint is that agents must communicate amongst themsleves using a common negotiation protocol.

The agent model is described in greater detail in [RLU94a].

8.2 User interfacing

Users may interact with their agents and other services via telephones or workstations. Service interfaces were designed to allow different kinds of user interfaces to be attached to them. In our prototype, we used InterViews [LCI+92] over X11 [SG86] to build graphical user interfaces, and developed an object-oriented voice interaction toolkit (VitKit) [RLU94c] to build telephone user interfaces. The latter consists of a C++ class library representing a variety of interaction techniques and composition mechanisms. It supports dynamic construction and re-configuration of interfaces and adopts a very flexible approach to interfacing with underlying applications.

9 Discussion

This report has outlined the key concepts and models that emerged from the ODO project. Many of the ideas *per se* are not entirely new, but our contribution lies in the way we have adapted them for use in an interactive, flexible, secure, wide-area, open distributed system to support integration of voice and data services.

Although a number of demonstrations were built, temporal and budgetary constraints did not permit us to build a stable enough prototype for users to try out. Such a prototype will enable us to better guage the kind of functionality required of services and agents, and to design more appropriate user interfaces and policy-specification languages. Nevertheless, the architectural concepts remain valid, regardless of the fine-tuning required to maximise usability and ease-of-use.

We believe that many of the concepts and models that came out of ODO are relevant to application domains other than office support systems. In particular we have applied the agent-model to the area of advanced service provision in telecommunications [RU95] as a means of getting around the feature interaction problem [C+93] and giving subscribers more flexibility. The agent-model is also relevant to creation and manipulation of connections in multimedia systems.

We are currently focussing on the negotiating agents model, seeking ways to conduct multi-party negotiations and also trying to incorporate quality-of-service (QoS) issues in the model.

References

[Arc92] Architecture Projects Management Limited, Poseidon House, Castle Park, Cambridge CB3 0RD, United Kingdom (apm@ansa.co.uk). *RM.099.02: An Overview of ANSAware 4.1*, May 1992.

[Arc93] Architecture Projects Management Limited, Poseidon House, Castle Park, Cambridge CB3 0RD, United Kingdom (apm@ansa.co.uk). *AR.008.00: A Framework for Federating Secure Systems*, May 1993.

[C+93] E. Jane Cameron et al. A feature-interaction benchmark for IN and beyond. *IEEE Communications Magazine*, 11(3):64–69, March 1993.

[LCI+92] Mark A. Linton, Paul R. Calder, John Interrante, Steven Tang, and John M. Vlissides. *InterViews Reference Manual, Version 3.1*, 1992.

[Lin92] Peter F. Linington. Introduction to the basic reference model of open distributed processing. *IFIP Transactions C, Special Issue on Open Distributed Processing*, C-1:3–13, 1992.

[RLU94a] Mike Rizzo, Peter F. Linington, and Ian A. Utting. Call management in the Open Distributed Office. Technical Report 15-94, Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent CT2 7NF, United Kingdom, August 1994.

[RLU94b] Mike Rizzo, Peter F. Linington, and Ian A. Utting. Integration of location services in the Open Distributed Office. Technical Report 14-94, Computing

Laboratory, University of Kent at Canterbury, Canterbury, Kent CT2 7NF, United Kingdom, August 1994.

[RLU94c] Mike Rizzo, Peter F. Linington, and Ian A. Utting. VitKit: a voice interaction toolkit. Technical Report 13-94, Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent CT2 7NF, United Kingdom, August 1994.

[RU95] Mike Rizzo and Ian A. Utting. An agent-based model for the provision of advanced telecommunications services. In *Proceedings of TINA '95*, pages 205–218. IEEE Communications Society, 1995.

[SG86] Robert W. Scheifler and Jim Gettys. The X window system. *ACM Transactions on Graphics*, 5(2):79–109, April 1986.

[WHFG92] Roy Want, Andy Hopper, Veronica Falcao, and Johnathon Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.