Modeling and Formal Specification of the Personal Communication Service

D. Desbiens, G.v. Bochmann, A. Das and J. Dargham
Département d'IRO, Université de Montréal
CP 6128, Succursale A
Montréal, Québec, Canada, H3C 3J7

Abstract

Personal Communication Service is an evolving network service concept being developed in order to provide greater personal mobility, increased service flexibility and customer control by eliminating the tight relationship between terminal identity and subscriber identity. In this paper, we present a model and a formal specification of the Personal Communication Service obtained by the application of an object-oriented system design methodology and described using the executable object-oriented specification language Mondel. The goal of developing a specification for PCS is primarily to introduce some structure and formalism in its description, which has so far been done informally, and also to provide a better understanding of its constituent elements and their interrelationships. As Mondel is a executable specification language, simulation was used to verify the basic functionality defined in this specification of PCS. Simulation using various scenarios also provided a means of presenting the different concepts of PCS.

1: Introduction

Historically, telecommunication services have always used terminal identification as a means of customer identification. Eliminating the close relationship between terminal identity and subscriber identity by the introduction of a unique, network-independent identification for each subscriber would permit greater personal mobility, increased service flexibility and customer control. The Personal Communication Service (PCS) has been defined as a network service concept which will enable subscribers to establish and receive calls for different service types based on a single Subscriber Number (SN), across multiple networks, from any arbitrary network entry point with service access limited only by the access/destination network and terminal capabilities [CCITT 89a-c] [Rgn 90].

This basic definition clearly shows that PCS is a network concept which implies inter-networking. More precisely, PCS can be seen as an inter-network service where a subscriber can access its different services by any arbitrary entry point of any sub-network integrated into the PCS inter-network, limited only by access/destination sub-network and terminal capabilities.

PCS is an evolving concept which is being discussed in standardization committees and for which experimental pilot systems are being developed. In this stage of its development, it is important to well understand the implication of the various design decisions and to validate the resulting system designs. We think that formal models, like the one described in this paper, can be useful for improving the understanding of the problems and the validation of the design decisions.

One of our goals for the development of a formal model of PCS has been the introduction of structure and formalism into the description of PCS, which has previously only been described in the english language. Such structure and formalism may provide a better understanding of the constituent elements of PCS and their interrelationships. The specification can also be used as a form of structured documentation. As the specification has been written in an executable specification language, simulations can be performed to verify the basic functionality defined. The simulation of various scenarios also provides a better understanding of PCS and thus helps in further refining its concepts.

The development of our PCS specification has been an experiment in the application of an object-oriented design methodology [Boch 92] and the use of a new object-oriented specification language, called MONDEL [Boch 90l].

The PCS model presented in this paper attempts to achieve a full representation of the service, with complete abstraction of architecture, physical distribution, detailed interfaces and implementation considerations. All aspects regarding Intelligent Networks [AIN 91] [Homa 92] and communication architecture are excluded from this model. Thus, the resulting model provides the service view as seen, in an abstract sense, by the PCS subscriber. It may be considered to be the input to the network architecture and protocol designer. Since we are primarily interested in obtaining a global view of PCS, details pertaining to interworking among several PCS networks are omitted in this paper.

The paper is organised as follows. In Section 2, we give an overview of PCS. In Section 3, we first give a brief description of the design methodology. We then describe the application of this methodology to the PCS example. In Section 4, we give an overview of Mondel and different specification styles. We also present in this section a specification in Mondel of PCS and discuss the simulation studies performed within the Mondel development environment. We end with a few concluding remarks in Section 5.

2: Overview of PCS

This section describes the environment in which PCS [CCITT 89a-c] is developed and the services provided. It also takes into account the architecture and implementation issues that are required to support the basic elements of the services, in order to show their feasibility. The definition of PCS used as framework in the present work relies mainly on the UPT Service Description [CCITT 89a-c] enhanced from discussion with the PCS team of Bell Northern Research (BNR) at Montréal.

2.1: The support environment of PCS

It is not our intention to describe the intelligent network and how it provides a framework for the implementation of PCS. This section takes in consideration the architecture only to support the feasibility of the service elements of PCS and to situate those aspects which we wish to study. PCS relies heavily on the widespread penetration of digital switching technology and on the availability of an Intelligent Network architecture (IN) [AIN 91]. Intelligent Network is a telecommunication network service control architecture, which allows database management for distributed applications to realize some functions such as tracking and service provisioning for mobile subscribers.
The main technical features that the IN architecture introduces are:

- Network connection control intelligence at centralized nodes. The process that provides the network connection control intelligence is known as the Service Control Point (SCP).
- Network process which provides switching services and functions to provide enhanced services to the subscriber. This process is known as the Service Switching Point (SSP).

The IN architecture implies the use of standard communication architecture, such as Open System Interconnections (OSI) and Common Channel Signalling no.7 (CCS7), to realize the signalling between the different elements composing the distributed systems.

2.2: Basic PCS concepts

The Personal Communication Service (PCS) can be defined as a network service concept which will enable subscribers to establish and receive calls for different service types based on a single Subscriber Number (SN), across multiple networks, from any arbitrary network entry point, and with service access limited only by access/ destination network and terminal capabilities.

Based on this basic definition of PCS, it is clear that it is a network concept which implies inter-networking. More precisely, PCS can be seen as an inter-network service where a subscriber can access its different services by any arbitrary entry point of any sub-network integrated into the PCS inter-network, limited only by access/destination sub-network and terminal capabilities.

Note: In what follows, the term network will be employed to refer to the PCS inter-network. The term sub-network will be employed to refer to a particular network which is an integrated part of the PCS inter-network.

The major attributes of PCS are as follow :

- The service capabilities of a subscriber are associated with the subscriber identity (the Subscriber Number), rather than the physical address of a terminal. This removes the dependency between the terminal and the subscriber.
- A subscriber subscribes to the services offered by a specific sub-network. He has to specify which type of services he wants to subscribe to among the particular services offered by the network, and may specify particular parameters for each service type he/she subscribes to. These specifications form the Subscriber Feature Profile.
- The subscriber has a direct control on his feature profile.
- For each service type, the subscriber can specify location characteristics for default destinations, and also for temporary destinations.
- The subscriber can access any of the services subscribed to, from any sub-network in the PCS inter-network, limited only by the capabilities of the sub-network/ terminal access/destination.
- All terminal devices like telephones and handsets are owned by PCS subscribers or by the network operator. These owners are responsible for what is done on their terminal devices, in terms of that they are billed for the services accessed by the use of their terminal when nobody explicitly identifies themselves. Then, the model will allow some kind of control for the owner for his terminal devices.

The characteristics of PCS involve a certain management of the subscriber communication environment. This environment has to be defined in terms of the subscriber service profile, the service type accessed, physical address localization, access terminal identity/ type/capabilities and sub-network identity/type/ capabilities.

These elements of the subscriber environment can be divided into two major groups:

- The service characteristics, relating to the service type accessed and the subscriber service profile.
- The location characteristics, relating to network port, terminal identity/type/capabilities, sub-network identity/ type/capabilities, network operator identity/capabilities and service provider identity/capabilities.

The communication environment information will be located in the network. This implies that the PCS network is an "intelligent network", it has the capability to access different databases which store the necessary information. The network databases would provide all the information regarding the communication environment required to make the call, such as :

- the destination for the call in terms of the sub-network identity, the terminal identity and the network port,
- access/destination terminal capabilities: type, access medium, bit rate, terminal, protocol,
- access/destination sub-network capabilities:type, network protocol, service capabilities, service provider,
- the service type concerning the call, and
- the service profile of the called/calling party.

A certain controller in the network will use this information to address the call and to adjust the service profile of the call to the capabilities/restrictions on the access/destination network and terminal, and to the called party service profile.

2.3: The PCS functionalities

When a subscriber wants to invoke PCS, he/she has first to identify himself. This will be done by (1) identifying the service accessed, by first invoking PCS, and then identifying the service type to be used, (2) providing his Subscriber Number , and (3) providing a Personal Identification Number (PIN some kind of password to ensure access security of his service).

After the identification phase, the subscriber can invoke the PCS functions which permit the subscriber

- to make calls based on either Subscriber Number or Telephone Number,
- to receive calls addressed to its Subscriber Number,
- to use an arbitrary terminal to access his personal service for a short period or a longer period of time,
- to indicate where he wants to receive his incoming calls for different services,
- to have control on his Subscriber Feature Profile,
- if he owns a terminal device, to control the feature profile of this device.

3: Modeling PCS

3.1: Overview of design methodology

Even though the success of a design still relies on human expertise in the application domain, a design methodology provides some simple, efficient means to structure the process leading from an informal and incomplete definition to a formal specification. Our methodology is based on the object paradigm [Booc 86] [Cham 92]. The idea sustained by the methodology is to describe the application as a composition of objects with relationships between them, interacting together to realize some functions. This methodology could be applied to any object-oriented project modelisation, using any object-oriented language. It provides a means to organize and structure the specification development process and to check the consistency of the model. It promotes an efficient and easy re-use of specifications and software items common to different specifications by providing data abstraction.

Our design methodology [Boch 92] consists of a preliminary step and three modeling steps.

During the preliminary step we define the problem. The problem is defined in this step by analyzing and determining the requirements. This can be simply done through discussion with people that know the problem and will use the software under construction. The result of this activity must be expressed in a language or notation that can be understood both by the analyst and the users such that they come to an agreement with what has to be done.

Step 1 results in a conceptual model of the domain structure. This model shows the classes of objects, the inheritance relationships, the structure of the objects, their attributes and the relations among objects classes. The conceptual model is represented graphically using entity-relationships diagrams [Chen 76], which are extended to show the inheritance relationships.

At Step 2, operations are identified and allocated to the classes of objects. Sources of operations are functions required and identified during the preliminary step. In accordance with the object orientation, operations are allocated such that every object encapsulates all the operations required to modify and access its internal structure. At this stage, parameters and results of operations may also be determined. Some functions may also uncover some new objects which must be integrated into the application domain.

It is in Step 3 that the behaviors of the objects are defined; these behaviors determine the semantics of the operations and the conditions related to their execution. This step leads, in general, to a better understanding of the problem as well as to new classes of objects, new relations and new operations. These new elements have to be integrated in the model obtained during the previous activities. This methodology is therefore iterative and the design activities of the different steps are not isolated from each other.

Note that this modeling process starts by describing aspects which are rather static, e.g. the classes of objects. Then, we gradually integrate in the model the dynamic aspects. First by describing the interfaces of the objects (i.e. the operations) then by specifying their internal aspects (i.e. the meaning of the operations). Objects are relatively independent of each other in terms of the actual algorithms selected for implementing the operations. This implies that the algorithms are easier to modify and maintain.

In the following subsections, these four steps of the methodology are applied to obtain a formal model for PCS. A more complete description can be found in [Desb 92].

Concerning the preliminary step, most of the information has already been described in section 2 of this paper. We delimit the scope of the model by considering only a global view in which a full representation of the service is taken into account, with complete abstraction of architecture, interface and implementation considerations. All aspects regarding Intelligent Networks and communication architecture are excluded from this model. Those aspects pertaining to interworking of PCS networks are also omitted.

3.2: Design step 1: definition of the domain

This step can be decomposed into two substeps: (1) identification of the entities of interest and their attributes, and (2) identification of the various relationships that exist among these entities. For the global PCS specification we have identified the the entities and relationships [Desb 92]. A subset of these entities and relationships is described in the following:

3.2.1: Entities

Subscriber: This class represents the physical subscribers. This class is inherited from the generic User class and is refined by the fact that it is the entity to which we recognize the access to PCS by giving it, and associating with it, a Subscriber Number.

Subscriber Number (SN): This class describes the network identifiers of the subscriber. An identifier uniquely identifies the subscriber in the PCS network. This is the primary key in the PCS system used to manage the subscriber services, therefore many relations depend on it. (Note: Just as the telephone number was doing earlier this entity represents the public address of the subscriber. The value of the Subscriber Number is unique).

Personal Identification Number (PIN): The subscriber number being part of the public domain, this entity class has a subscriber service access security purpose. It can be seen as a password to provide access security to the service.

Service Type (ST): The purpose of this entity class represents the different types of service that can be used independently in the telecommunication context and have to be addressed independently to receive or make calls. In the PCS context "Service" is intended to identify the application context of the call. Examples of application context are: telephony, facsimile, data, ISDN, etc. (Note : This class plays a part in addressing. A Subscriber Number may be located for receiving calls in different locations, for different types of service. A Subscriber Number cannot be located in more than one location for the same Service Type for receiving calls.)

Subscriber Feature Profile (SFP): This entity class represents the Feature Profile of a subscriber. It is intended to specify the general subscription information of a PCS subscriber as its billing address, and the different feature parameters of its subscription, such as call screening parameters.

Default Feature Profile (DFP): This entity class has the intent of providing some access and utilization control of the network port to the owner of the network port. For example, the owner may restrict the access to the network port he owns to certain persons only, on certain specific hours only, or he may prohibit long distance calls.

Network Port (NP) : This entity class represents a physical network access.The network port is identified by a physical address. It identifies uniquely a physical network access point. The physical network access point may be a line card in the case of a wired sub-network. As the PCS subscribers will have to address the physical location where they want to relocate their services, they will have to use a public identification of the Network Port; thus in the context of PCS, the Network Port will be identified as the actual public identification of the physical network end point : the "telephone number". (Note : This is an important entity in the PCS system; many relations depend on it. For the network, this is the entity for physical location addressing.)

3.2.2: Relationships

A relationship is a perceived association between entities in the domain. This information will allow some domain semantics. A relationship may be understood as a set of n-tuple grouped under a common meaning, where each n-tuple links n entities by this meaning. An instance of a relationship class is an object, thus a relationship has

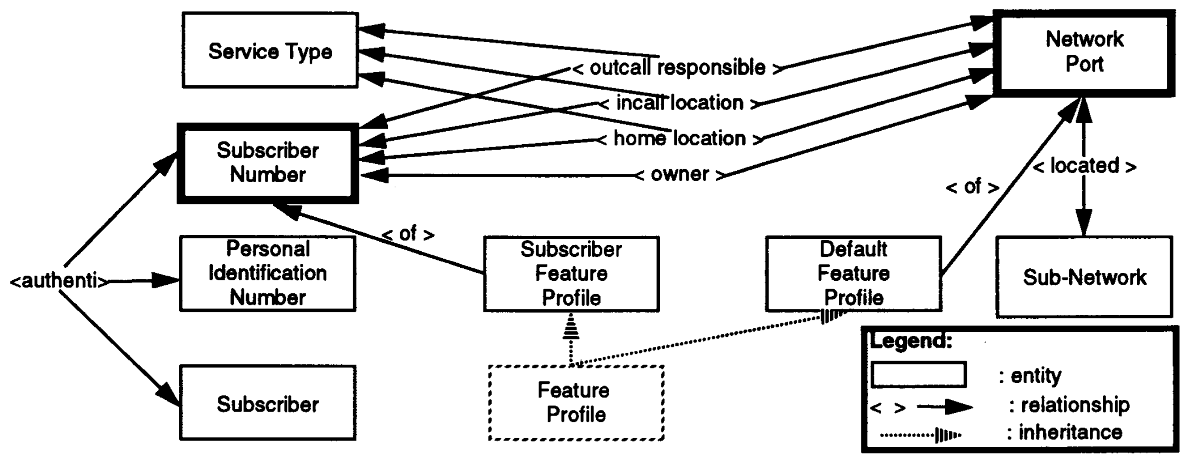

Fig. 1 : Partial entity-relationship diagram of PCS

attributes and a behavior. The attributes are mainly the different entities which are implied in the relationship, however some attributes may represent specific characteristics of the relation, (e.g. the time duration of an incall location of a subscriber onto a network port). For the PCS application, the following key relationships have been identified. They are shown on the diagram of Figure 1.

Authentication : The Authentication relation links all components involved in the authentication of the subscriber. Thus, given a SN, the network can check the PIN in order to authenticate the subscriber who will in fact be billed for all services used after authentication (see Figure 1). Note that a subscriber can have several SNs and to each SN corresponds exactly one PIN.

Owner : This relationship represents the identification of the subscriber who possesses a particular Network Port. By default, he is responsible for the Network Port he owns. This means that, he is the one who will be billed for all services used on this Network Pork without prior authentication. The owner exercises a certain control on the utilisation of the Network Port via the entity Default Feature Profile which defines the service available to users without prior authentication. He can also restrict the range of services that are available with authentication; for instance, the relocalization of incoming calls can be restricted or prohibitted.

Home Location : This relationship represents the default location for routing incoming calls of a subscriber for a particular Service Type (e.g. a user may be located for reception of a voice call on Network Port A, whereas reception of a fax call may be located at Network Port B, both calls could occur simultaneously).

Incall Location : This relationship represents the current location where incoming calls must be routed. There always exists an incall location for a subscriber and is dynamic. During an incall session the incall location of the subscriber is the one specified for the incall session otherwise it is the one specified by the home location. (Note : An arbitrary number of subscribers can be located at a given Network Port and a given Service Type. Also, a subscriber should be located at exactly one Network Port for a particular Service Type at any given moment.)

Outcall Responsible : This relationship represents the subscriber responsible for the utilisation of the services provided at a given Network Port, if these services are used without prior identification(e.g. subscriber A could be responsible for all voice services, and subscriber B for all calls related to fax services). There always exists an outcall responsible for a network port and can be dynamically reassigned. During an outcall session the outcall responsible of the network port is the one specified for the outcall session otherwise it is the one specified by the owner relation. (Note: At all times, responsibility for the use of services of a particular Service Type at a given Network Port should lie with exactly one subscriber. However, the same subscriber can be responsible for a particular Service Type for any number of Network Ports.)

The diagram of Figure 1 shows the PCS entities and relationships. Entity classes are represented by rectangles. An inheritance relation between two entity classes is represented by a dashed arrow from the subclass to the superclass. For instance, the "Feature Profile" class is specialized, using inheritance, into two subclasses, namely "Subscriber Feature Profile" and "Default Feature Profile". In the diagram, we represent classes of relationships (i.e. relations) between objects by angled brackets. For instance, there are the classes of objects "Subscriber Number" , "Service Type" and "Network Port" which are linked together by a relation named "owner".

The aspects related to the ownership and access control of a Network Port (modeled by the Default Feature Profile entity and the <owner> relationship) represent new insight of interest for PCS. They have not yet been addressed by the standard bodies. It presents a security aspect which could be interesting even outside the scope of PCS.

3.3: Design step 2: identification of operations

The aim of the specification is to stress the basic characteristics of PCS. Consequently, the functionality gives emphasis to personal mobility. The definition of the PCS functionality relies mainly on the UPT Service Description [CCITT 89a-c] enhanced from discussion with the PCS team of Bell Northern Research (BNR) at Montreal.

At this stage we allocate operations to objects. To achieve the identification of the set of operations supported by each object of the domain, the functionality of the application has to be first defined. Thus this step deals primarily with the functional definition of the application. The different tasks involved in this step are: (1) definition of the functions of the application, and (2) definition of the role of each object (i.e. definition of the set of actions that have to be done by each object in order to provide each function).

Personal mobility is provided by allowing the PCS subscribers to receive/make their calls at any arbitrary network port. The main PCS functions allow the subscribers to specify where they currently want to receive/make their calls for a fixed period of time. The subscribers are able to relocate themselves for incoming calls and for outgoing calls.

For example, the process of defining the functions related to the outcall relocation of a subscriber is detailed in the following subsections (a discussion of the other functions can be found in [Desb 92]).

3.3.1: Example of a function definition

The outcall relocation allows a subscriber to use a network port as his own. The subscriber is billed for all telecommunication services used at the network port for the time of the outcall relocation. Three function groups have been identified in the context of outcall relocation:

The outcall session allows the subscriber to take possession of a network port for a certain service type. An outcall session is started by invoking a specific PCS function (start_outcall_session) and is stopped by invoking a specific PCS function (terminate_outcall_session), or can stop automatically at the expiration of a time-out defined at the invocation of the session. When the outcall session is terminated, the owner of the port takes possession of the network port. The owner of a network port is responsible for it by default. A subscriber may engage in any number of outcall session simultaneously, however only one outcall session may be active on a network port at any time.

The outcall series allows the subscriber to take possession of a network port for a certain service type for a duration limited to the activation of the network port on which it takes place (e.g. the time a phone is off the hook in the case of a conventional phone). An outcall series is initiate by invoking a specific PCS function (do_series). The outcall series overrides the outcall session during its activation. The outcall series is similar to the existing calling card service.

The abort outcall session allows the owner of a network port to abort any outcall session taking place on the network port he owns. An abort outcall session is done by invoking a specific PCS function (terminate_other_outcall_session).

3.3.2: Definition of the role of each object

As the specification describes a service, the most important objects of the model are the subscriber, who uses the service, and the network port, which provides it.

The role of the subscriber is to model the correct behavior of a real PCS subscriber. The subscriber provides a high level interface in order to perform the specific PCS functions identified in the previous section. The subscriber's interface is the point of interaction for simulation. The subscriber can be asked to start an outcall session, to terminate an outcall session, to do a series of calls, or to terminate another outcall session.

These operations of the Subscriber (related to the outcall relocation), and their parameters, can be described as follows :

```
Start_Outcall_Session
( Np : string; {the Network Port of interaction}
  Sn : string; {the identification of the callinguser for the
authentication}
  Pin : string; {Personal Identification Number}
  Action_Np : string;{the Network Port involved for this
operation}
  Service_Type : string;{the service type concerned }
  Duration : integer;{The duration of the session})
Terminate_Outcall_Session
( Np : string; {the Network Port of interaction}
  Sn : string; {the identification of the callinguser for the
authentication}
  Pin : string; { The Pin for authentication }
  Action_Np : string;{ The NP implied in operation }
  Service_Type : string;{the service type concerned });
Do_Series
( Np : string; {the Network Port of interaction}
  Sn : string; { identification of the calling user }
  Pin : string; {Personal Identification Number}
  Service_Type : string;{the service type concerned });
```

```
Terminate_Other_Outcall_Session
 ( Np : string; {the Network Port of interaction}
   Sn : string; {the identification of the  calling user}
   Pin : string; {Personal Identification Number}
   { The Sn authenticated has to be the owner of the
     Action_Port }
   Service_Type : string {the service type concerned })
```

The role of the network port is to represent the interaction between the subscriber and the PCS application. The behavior of the PCS application is represented by the network port because the specification is aimed at modeling the PCS service. The network port allows the subscriber to activate it or deactivate it (by the specific operation off_hook and on_hook) and to signal the different codes specific to PCS, such as Signal_PCS_Code to signal that he wants to invoke the PCS application, or signal_start_outcall_session to signal that he wants to start an outcall session. The signaling is represented in an abstract manner by operations because the interactions with the subscriber in an implementation of PCS may be carried out in different ways (by pressing a specific series of digits on a pulse tone phone set, by voice recognition, or any other sophisticated user interaction technology)

These operations of the Network Port (related to outcall relocation), and their parameters, can be described as follows :

```
Off_Hook(user : User );
On_Hook;
Signal_Pcs_Code;
Signal_Authentication(SnValue, PinValue : string );
Signal_Start_Session_Outcall
    ( NpValue : string; StValue : string; Duration : integer );
Signal_Terminate_Session_Outcall
    ( NpValue : string; StValue : string );
Signal_Terminate_Other_Session_Outcall
    ( StValue : string );
Signal_Start_Series( StValue : string );
Dial_Series( SnValue : string );
Signal_Continu;
```

3.5 :Design step 3: definition of behavior

This last step consists of describing precisely the meaning of each operation. An operation may have a different meaning (thus lead to a different behavior) in regards to the contexts of the global behavior of the object when it is invoked. The behavior of an object is seen as the set of actions related to each operation in regards to each context in which the object can be. One of the possible conceptual tools for modeling this behavior are state-transition diagrams where each context of the object is a state and each action is a transition.

An object may assume different functions simultaneously, for instance a user may be participating in an incall session, an outcall session and also initiating a call, simultaneously. The behavior of an object may be partitioned into several parts where each part represents the behavior associated to a given function. The behavior related to a particular function can be modeled by an appropriate finite state machine. The behavior of an object can be viewed as the composite behavior of the automata representing the various functions running concurrently.

For instance in the PCS specification, the behavior of a subscriber is expressed as the composition of seven parallel automata, each one related to a PCS function assumed by a subscriber.

```
parallel
              Outcall_Session_Idle;
      and     Incall_Session_Idle;
      and     Series_Idle;
      and     Outcall_Session_Abort;
      and     Incall_Session_Abort;
      and     Call_Idle;
      and     Response_Idle;
end;
```

Figure 2 presents the finite state machine which describes the behavior of the subscriber related to the Outcall_Session function. In the state Outcall_Session_Idle, the subscriber may start an outcall session. Starting an outcall session involves several operations invoked on other objects of the PCS system. The subscriber will activate the network port (Off-Hook signal), validate its personal security code and signal the new outcall session parameters. This implies the creation of a new instance of the "outcall responsible" relationship shown in Figure 1. After the execution of this operation, the subscriber is considered to be involved in an outcall session, represented by the Outcall_Session_Active state. In this state, the only operation available is the termination of the session, which again involves similar operations on the other objects of the system.

Transition diagrams, such as the one above, are useful for defining the temporal order of interaction, but do not express all aspects of object behavior. In particular, the values of the operation parameters are not considered and the effect of operations on object instances in the system are not addressed (e.g. the creation of new relationships as discussed above). For the PCS specification considered in this paper, we have used the Mondel specification language for this purpose. The use of this language for the description of the PCS example is described in Section 4.2.

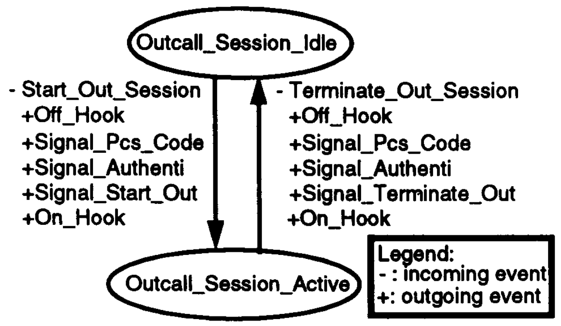

Figure 2 : Finite State Machine description of the subscriber's outcall session

4: A PCS specification written in MONDEL

4.1: Overview of MONDEL

Mondel [Boch 901] is an Object-Oriented specification language with certain particular features, such as multiple inheritance, type checking, rendez-vous communication between objects and the possibility of concurrent activities performed by a single object. It is an executable object-oriented specification language with a formally defined semantics. The purpose of its development was the modelisation and specification of applications from the distributed systems domain.

A Mondel specification is basically composed of a set of objects interacting together. Each object composing the specification can be seen as an independent parallel process communicating with other objects by a rendez-vous mechanism. The communication rendez-vous is an operation call on an object which offers it. The operation will be executed when the caller object accepts to execute it. The caller object is blocked at the rendez-vous point until the called object releases it.

4.2: Different specification styles

A specification may be written at a more or less abstract level depending on the goal of the specification. A specification may be intended for the expression of a design and for its validation, for the verification of certain properties of a system, for the generation of an implementation, for the derivation of test cases for the validation of implementations, or for all of these purposes [Boch 90g]. Two basic approaches to behavior definition are assertional and algorithmic. Both of them are supported in Mondel; the statements of the language support the algorithmic style, while the assertional styte, which is useful for verification purposes, is supported through special features such as the ASSERT statement and invariants.

Our PCS specification has been written with the aim of expressing the design of PCS as a service and its validation by executing the specification. We used an algorithmic approach with a state-oriented style. Each state is represented by a Mondel procedure which defines the behavior of the object in the given state. The overall structure of the PCS specification may be considered resource-oriented in as far as each object of the system represents a "resource".

4.3: FSM-oriented specification style and PCS example

This section gives a description of the subscriber's behavior for the function Outcall_Session in Mondel. The behavior specification is included in the class definition of the subscriber type, which is given below. This type definition includes first the declaration of the object attributes, in this case the name of the subscriber, and of the operations which take the same form as given in Section 3.3.2.

In this example, we show only two functions in parallel: Outcall_Session and Incall_Session. In the behavior definition, they are represented by their respective initial state, e.g. Outcall_Session_Idle. The behavior in the Outcall_Session_Idle state is defined by the procedure of the same name, which specifies a single possible transition which starts with the acceptance of the Start_Outcall_Session operation. The execution of this operation implies a number of operations to be executed on the associated network port (np). The execution of the Signal_Start_Session_Outcall operation implies the update of the "outcall responsible" relation shown in Figure 1 as defined in the definition of this operation in the behavior of the Network Port object type (not shown here).

The keyword "return" indicates the end of the operation. It is followed by the invocation of the procedure Outcall_Session_Active, which represents the new state of the Outcall_Session function. This new state is invoked in parallel with another instance of the Outcall_Session function, which is still in its initial "Idle" state for the case that the subscriber wants to start another Outcall_Session function, possibly for another service type.

```
type Subscriber = persistent with
  name : string;
operation
*** see Section 3.3.2 ***
behavior
{ two different "state machines" are executed in parallel }
parallel
    Outcall_Session_Idle;
```

```
and  Incall_Session_Idle
end;

where
procedure Outcall_Session_Idle =
{being in the Idle state the possible transaction is
Start_Outcall_Session}
  accept
    Start_Outcall_Session
    {The input parameters are Np, Sn, Pin, Action_Np,
     Service_Type, Duration}
   do
      define np = valid_np(Np) in
       np!Off_Hook(self);
       np!Signal_Pcs_Code;
       np!signal_Authentication(Sn,Pin);
       np!Signal_Start_Session_Outcall
               (Action_Np,Service_Type, Duration);
       np!On_Hook;
      return;
  end;
     {after this interaction the next state is
      Outcall_Session_Active}
      parallel
        Outcall_Session_Idle;
      and  Outcall_Session_Active
               ( Action_Np, Service_Type, Sn);
     end; {parallel}
endproc Outcall_Session_Idle


procedure Outcall_Session_Active(concern_np,
                    concern_st, concern_sn : string) =
{being in the active state the possible transaction is
Terminate_Outcall_Session}
  accept
    Terminate_Outcall_Session
    {The input parameters are Np, Sn, Pin, Action_Np,
     Service_Type}
       {triggered only  if it is the same parameters as for the
      start_outcall_session}
    provided concern_np = Action_Np and
            concern_sn = Sn and
            concern_st = Service_Type
    do
    define np = valid_np(Np) in
      np!Off_Hook(self);
      np!Signal_Pcs_Code;
      np!signal_Authentication(Sn,Pin);
      np!Signal_Terminate_Session_Outcall
                     (Action_Np,Service_Type);
      np!On_Hook;
      return;
  end;{accept}
endproc _Outcall_Session_Active

{procedure to validate the existence of the  network port }
procedure valid_np(Np_Value:string): NP =
  ifexist np : NP suchthat np.value = NpValue then
    return np;
  else
    writeln "inexistant _np";
    abort;
  end;
endproc valid_np
endtype Subscriber
```

4.4: Simulation of the model

In order to exercise the PCS specification, we have used a specification development environment [Boch 901] which includes the following tools: A syntax analyzer verifies the context-free syntax of a specification. A compiler verifies the static semantics of a specification and generates an intermediate Prolog representation. An interpreter uses this representation and permits the simulation of the specification. A verifier generates a reduced reachability graph of the specification, if the specification satisfies certain restrictions which ensure a finite reachability graph. The states of this graph correspond to all possible states which the system may reach, and is the basis for exhaustive validation for the absense of deadlocks and undesirable loops and the satisfaction of assertions and invariants [Barb 91].

The PCS specification has been simulated with two objectives: for the validation of the specification and for demonstration. For validation, a simulator object has been defined which allows the user to interactively guide the execution of the various PCS operations of the subscriber in an arbitrary order with arbitrary parameters. The system specification includes an initial behavior which initializes a small PCS system creating a certain number of PCS objects (of the types shown in Figure 1), and which then initializes the simulator object which waits for simulation requests from the user. This kind of simulation environment was used to validate the PCS specification through executing a large number of different scenarios.

For the purpose of a demonstration, a predefined scenario was defined as the behavior of a demo-simulator. The system including this simulator automatically executes the scenarios when the interpreter starts the automatic execution of the specification. Appropriate messages on the screen show the advancement of the scenario.

5: Conclusions

The PCS model has been developed in collaboration with, and for the needs of the PCS team of Bell Northern Research (BNR) at Montreal.The modeling of the PCS service using the methodology presented in the previous sections and the formal description technique Mondel has been useful for the BNR team in increasing the understanding of PCS elements and their interworking. The PCS modeling experience shows that the use of the methodology and simulation provides means to improve the understanding of complex, still yet vaguely defined applications.

The methodology has helped structure the process of specifying PCS by allowing to adapt the level of specification to the specifier's needs. The documentation of the methodology and the Mondel specification have facilitated the transfer of knowledge. Mondel has been perceived as an easily readable language.

The ability to simulate service scenarios has provided the detection of some service discrepencies in the first definition of PCS functionality. The simulation has also been useful in communicating the PCS service framework in an easily understood form to the different groups involved in various aspects of PCS in BNR.

The modelisation provided new insight of interest for PCS regarding aspects related to the ownership and access control of a Network Port. These aspects have not yet been addressed by the standard bodies.

Presently we are working on an extension of the PCS specification which models the interworking aspects of PCS. While in the overall service specification presented in this paper the physical location of objects has been completely ignored, except for the association of users with Network Ports, in this extended version of PCS, we must consider the association of the PCS objects shown in Figure 1 with the different PCS subnetworks and the physical databases contained in these networks. Such a distributed model of PCS will allow the modelling of various interworking approaches and protocols to be explored.

Acknowledgements: We would like to thank Louis Gagnon from BNR for many discussions and input which lead to the PCS specification described here. The development of the PCS specification described in this paper was part of a joint research project by the Computer Research Institute of Montreal (CRIM) and Bell-Northern-Research. We thank the project team, in particular Michel Barbeau, Aiwa Liu, and Louis Lecomte, for their support.

References:

[AIN 91] Bellcore, Advanced Intelligent Network (AIN) Release 1 Switching Systems Generic Requirements, TA-NWT-001123, Issue 1, May 1991.

[Barb 91] M. Barbeau and G.v. Bochmann, Formal Specification and Formal Verification of Object-Oriented Specifications Based on the Colored Petri Net Model, Publication #784, Département d'IRO, Université de Montréal, August 1991.

[Boch 90g] G. v. Bochmann, Protocol Specification for OSI, Computer Networks and ISDN Systems, April 1990.

[Boch 90l] G. v. Bochmann, M. Barbeau, M. Erradi, L. Lecomte, P. Mondain-Monval and N. Williams, Mondel : An Object-Oriented Specification Language, submitted for publication.

[Boch 90z] G. v. Bochmann and e. al., System specification with MONDEL and relation with other formalisms, Progress Report No. 13 for CRIM/ BNR project, June 1990.

[Boch 92p] G. v. Bochmann, S. Poirier and P. Mondain-Monval, Object-oriented design for distributed systems and OSI standards , Proc. of IFIP Int. Conf. on Upper Layer Protocols, Architectures and Applications, Vancouver, May 1992.

[Booch 86] G. Booc, Object-Oriented Development, IEEE TSE, February 1986

[CCITT 89a] CCITT, Universal Personal Telecommunications Definition and Attributes, COM XVIII D.130, June 1989.

[CCITT 89b] CCITT, Network Elements to support Universal Personal Telecommunications , COM XVIII D.131, June 1989.

[CCITT 89c] CCITT, Universal Personal Telecommunications (UPT) : A Framework For Describing UPT Data (The Network Service Database) , COM XVIII D.132, June 1989.

[Cham 92] Dennis de Champeaux and Penelope Faure, A Comparative Study of Object-Oriented Analysis Methods, Journal of Object-Oriented Programming, March 1992, pp 21-33

[Chen 76] P.P. Chen, The Entity-Relationship model - Toward a unified view of data, ACM Trans. on Database Systems, Vol. 1,No. 1, March 1976, pp.9-36.

[Desb 92] D. Desbiens, Modelisation and Specification of the Personal Telecommunications Services, Msc Thesis, Université de Montréal, 1992.

[Homa 92] J. Homa and S. Harris, Intelligent Network Requirements for Personal Communications Services, IEEE Communication Magazine, February 1992, pp. 70-76

[ISO 88b] ISO, Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique Based on The Temporal Ordering of Observational Behavior, IS 8807, 1988

[ISO 89] ISO, DIS 9595 Common Management Information Service Definition,1989.

[[Régn 90] J. Régnier , W.H. Cameron, Personal Communications Services-The New Pots, Globecom 90, San Diego, December 1990.