# Object Oriented Modelling of Advanced IN Services with SDL-92

F. Lucidi, A. Tosti, and S. Trigila
Sett. Elaborazione dell'Informazione
Fondazione Ugo Bordoni
Via B. Castiglione 59, Rome, Italy
nando@fub.it

**Abstract**[1]

*Long term trends in a telecommunication market, characterized by a fast evolving technology, a multiplicity of providers and the presence of chains of value-added services, call for architectural frameworks which allow the introduction of new and enhanced services and their management much faster and more efficiently than possible today. The Intelligent Network (IN) standards [1, 2] have been defined to meet this demand taking as a starting point the basic telephony system architectures. This paper describes a methodology for specification, validation and implementation of IN services and service features based on SDL-92 and its object-oriented constructs [6]. This methodology has been exercised in the realm of the Universal Personal Telecommunication service.*

## 1 Introduction

The liberalisation of the telecommunications market in combination with emerging advanced services such as mobility and multi-media services, or combinations thereof, will call for new architectural frameworks to telecommunications service provisioning [3] and sophisticated Service Creation Environments to support service creation [4].

The objective of the IN architecture under definition by ITU-T [1, 2] is to facilitate the creation, the deployment and the introduction of new telecommunication services (e.g. Freephone, Virtual Private Network - VPN, Universal Personal Telecommunication - UPT) as extentions of basic telephony services. The standardization process, carried on by ITU-T, toward the target IN architecture is a phased one. For each phase x a set of capabilities referred to as IN capability set x (CSx) is intended to be provided to address requirements for issues like creation, management and interaction of services or network management/interworking [5]. The first phase ended in 1992 with the production of the CS1.

Beside this first set of IN capabilities a conceptual model, namely the IN Conceptual Model (INCM), has been provided [2] as a framework for the design and description of services from their specification, from a user point of view, up to their deployment on the physical network. The adoption of the INCM should allow:

- service independence thanks to the definition of generic service building blocks (SIBs) for the construction of many different services;
- network independence based on the definition of functional network elements with specific functionality and
- vendor independence relying on the definition of standardized interfaces and protocols between the defined IN network elements.

A presentation of the major IN architecture constituents is given in Section 2.

Service creation is a complex process involving several activities within which methods, techniques and tools to be used are of paramount importance. The adoption of formal methods and object-oriented techniques is widely recognized as the best solution to face, in a heterogeneous market environment, emerging issues related to "customization" and "value chains of services". This issues, in fact, require both a high degree of flexibility, e.g. the possibility to reuse existing solutions in an organized/structured way, and unambiguous specifications of interfaces and services exchanged between different providers.

The "Specification and Description Language" (SDL) developed and standardized by ITU-T [6] is a major candidate for formal support of a methodology for computer-aided service creation, deployment and maintenance in conformance with a service architecture like IN. Basic reasons behind the choice of SDL are the following:

- it has been successfully used for years in specification and description of telecommunication systems;

[1] Work carried out in the framework of the agreement between Fondazione Ugo Bordoni and Telecom Italia

- it has a wide industrial acceptance;

- it is object-oriented: in its current version (SDL-92) a number of object-oriented concepts have been formally introduced into the language;

- it has a good tool support: a number of toolsets integrating all phases from specification to test are commercially available [7, 8].

This paper provides a methodology for specification, validation and implementation of IN services and/or service features based on SDL-92. Basically, the focus is on the Global Functional Plane (GFP) of the INCM since this is the place where service features are logically framed in terms of the broad functions required to support them from a network-wide viewpoint, i.e. looking at the underlying IN-structured network as a single entity.

The technical approach taken here is based on two interrelated elements: a "service-oriented semantics" and a generic model for IN services in the GFP. A service-oriented semantics, i.e. a mapping between IN architectural concepts and SDL language constructs, constitutes the rationale according to which the use of SDL for IN service modelling is recommended. A proposal for such a semantics is given in Section 3.

Taking as a starting point the correspondences implied by the service-oriented semantics, an SDL model of a generic IN service is proposed which presents on one hand characteristics that are common to all services and on the other generic structures that can be specialized to specific services. This model is discussed in Section 4.

In order to assess the envisaged methodology, an application example in the realm of UPT is chosen as a representative example of new telecommunication services that IN is intended to support. A mention to this application example is summarized in Section 5. The paper is concluded by evaluation remarks and recommendations for future work (Section 6).

## 2 Intelligent Network

IN is an architecture which, taking as a starting point the basic telephony system architectures, aims, especially for the benefit of telecom operators, to facilitate and accelerate the introduction of new and advanced telecommunication services in a cost effective way [9]. A fundamental element of the IN is the IN Conceptual Model (INCM) which represents a framework used by designers both to design IN services and to describe the IN architecture itself [10].

### 2.1 The IN Conceptual Model

The INCM is a well known reference model, generally presented according to Figure 1, displaying a hierachy of four Planes,

- Service Plane;
- Global Functional Plane;
- Distributed Functional Plane;
- Physical Plane;

each presenting a canonical viewpoint of the telecommunications network as a system for "open service provision", where independence between services and network technology is the main driving principle.

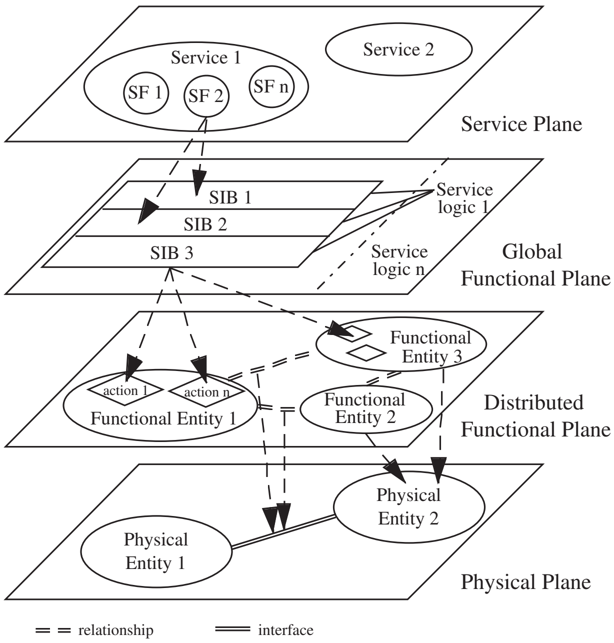


**Figure** 1: IN Conceptual Model

The *Service Plane* constitutes the topmost view, providing a common universe of discourse between service users and providers. It describes *services* and *service features* from a user perspective, independently of how the service is implemented in, and provided by, the network. A service is an organized package of telecommunications capabilities offered to users. A service feature is a specific "component" of a service that can also be used in another context, packaged with other services/service features as a part of commercial offering.

Service features may concur to the core of a service or may constitute optional part offered as an enhancement to a service. It is to be noted that this distinction has been criticised by a trend in literature, aiming to a long term vision, where there are no components 'more equal' than others [17]. Anyhow, although this is a very hot issue, this paper will deliberately take a conservative attitude, in order to try to make the best with the INCM as it stands in the Recommendations.

The *Global Functional Plane* is directly related to the Service Plane and is within the realm of service providers and service designers. This plane models network functionality from a global or network-wide point of view. As such, the IN Structured Network is said to be viewed as a single entity in the GFP. In this plane, service and service features are redefined in terms of overall network functions required to support them. These functions are referred to as *Service Independent Building Blocks* (SIBs). This paper focuses on the Global Functional Plane. Therefore, a specific section is devoted to it.

The *Distributed Functional Plane* is the common universe of discourse between service providers and network operators and gives a distributed view of network. Each SIB identified in the GFP is mapped onto a *Distributed Service Logic* (DSL). A DSL is described as one or more *Function Entity Actions*, each being performed on a *Functional Entity*, and as information that flows between Functional Entities, referred to as *Relationship*. A Functional Entity is a unique group of functions in a single location and a subset of the total set of functions required to provide a service. The functional entities and relationships are described independently of how the functionality is physically implemented or deployed in the network.

The *Physical Plane* is in the market place where network operators and equipement providers meet. It describes the physical architecture alternatives for an IN structured network in terms of potential physical system (referred to as *Physical Entities*) in a network, and *interfaces* between Physical Entities.

The main relations between the logical entities in different planes are depicted with dashed arrows in Figure 1 and are best described by an almost direct quote of [2]:

- services and service features on the Service Plane are realized by means of the combination of SIBs on the Global Functional Plane;

- SIBs on Global Functional Plane are realized on the Distributed Functional Plane by a sequence of Functional Entity Actions and resulting information flow;

- one or more Functional Entities from the Distributed Functional Plane may be realized in a Physical Entity on the Physical Plane;

- one or more relationships from the Distributed Functional Plane may map into an interface on the Physical Plane.

## 2.2 The Global Functional Plane

Services identified in the service plane are decomposed into their service features then mapped onto one or more SIBs in the Global Functional Plane (GFP).

As described in [11] the logical entities contained in GFP are:

- *SIBs*;

- *Basic Call Process* (BCP);

- *Global Service Logic* (GSL).

### 2.2.1 SIB

A SIB is a standard reusable network-wide capability used by service designers to develop new services/service features. SIBs have the following characteristics:

- are service independent;

- are monolithic building blocks whose implementation details are hidden;

- define one complete activity.

A SIB has one logical starting point and one or more logical end points. Moreover, two types of data parameters are required for each SIB, dynamic parameters called Call Instance Data (CID) and static parameters called Service Support Data (SSD). CID defines dynamic parameters whose value will change with each call instance. SSD defines data parameters required by a SIB which are specific to the service feature description. These are data parameters whose value is fixed for all call instances.

A graphical representation of a SIB is given in Figure 2.

### 2.2.2 BCP

The Basic Call Process (BCP) can be viewed as a specialized SIB which provides the basic call capabilities that allow connectivity between parties in the network like, for example, connecting call, disconnecting calls and retaining CID for further processing. Also the BCP, as any other SIB, is service independent.

IN supported service/service features are represented through the use of chain of SIBs connected to the BCP [11]. The behaviour control from the BCP to the chain of SIBs (and vice versa) is passed at specific interface points called *points of initiation*

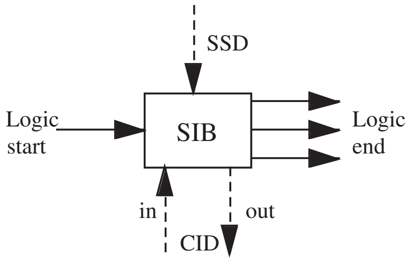


**Figure** 2: Graphical representation of a SIB

(POIs) and *points of return* (PORs). A POI is defined as the BCP functional launching point for the SIB chains. A POR, instead, identifies the functional point in the BCP where the SIB chains terminate.

#### 2.2.3 GSL

The GSL is the only entity, within the GFP, which specifically depends on the service to be realized. It defines how SIBs are to be chained together in order to provide service features. In particular, the GSL specifies the execution sequence of SIBs in a chain based on the knowledge of the chain topology and the possible termination results of each SIB.

The GSL also describes interaction between the BCP and the SIB chains in terms of interaction points (POI and POR) and exchanged data parameters.

A graphical illustration of the SIBs/BCP/GSL relationship is shown in Figure 3.

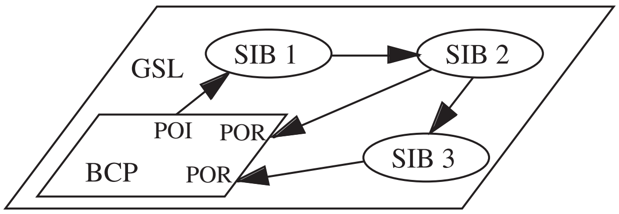


**Figure** 3: Global Functional Plane Scenario

## 3 Service-Oriented Semantics

The starting point is to model the generic IN service as an SDL system. Since IN services are logically framed in terms of service features, which are the smallest part of the service that can be perceived by the service user, and a basic call, which can be considered as a special kind of service feature common to all services, the SDL system is refined into two blocks: *ServiceFeatures* and *BasicCall* (see Figure 4).

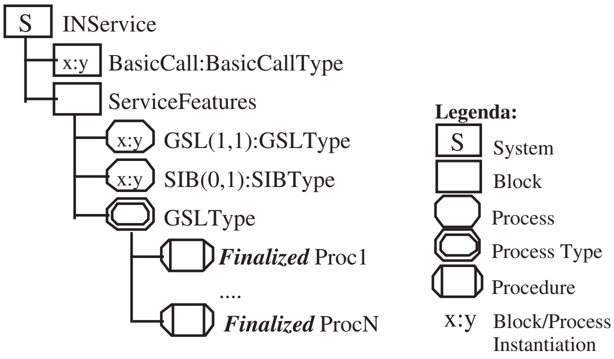


**Figure** 4: IN Service Specification Diagram Structure

The block *BasicCall* is an instantiation of the block type *BasicCallType*. In such a way the specification of the IN basic call can be taken outside the actual specification of the specific IN service in an external library modelled as an SDL package (see Figure 5) which is common to all IN service specifications.

The BCP completely identifies the behaviour of the IN basic call. Since the core structure to express behaviour in SDL is the process, it follows quite naturally to refine the block type *BasicCallType* into just one process modelling the BCP (see Figure 5).

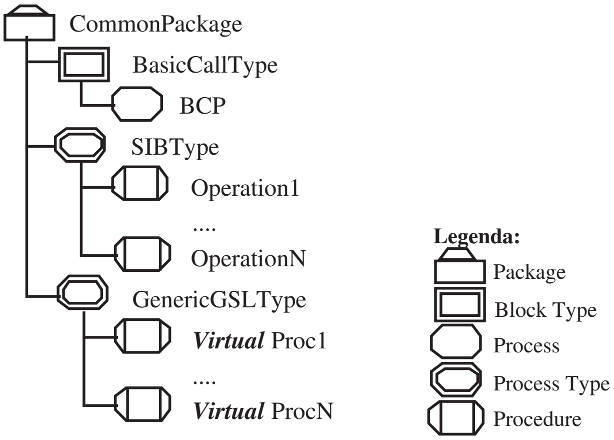


**Figure** 5: Common Package Diagram Structure

Service features are accomplished by chaining SIBs together according to a given logic which resides in the GSL. Two possibilities for the representation of the GSL and the SIBs have been proposed in [12]: a centralized variant, where the GSL is explicitly modelled as a process, and a decentralized variant where the GSL is distributed among the SIBs. The approach followed here is to have a centralized GSL. With this approach SIBs do not need to be adapted to different uses in different services as recommended in [11] and the service logic is specified in one place only. Therefore, the block *ServiceFeatures* is conveniently refined into one process *GSL* and a process *SIB* (see Figure 4).

The process *GSL* is an instantiation of the process type *GSLType* which on its turn is a specialization of the process type *GenericGSLType*. This last process type, whose actual specification is kept in an external package (see Figure 5), constitutes a generic model of the GSL. It provides on one hand the complete specification of the behaviour common to all GSLs, e.g. the interaction modality with the BCP or the way of triggering SIBs, and on the other hand specification "place-holders" for those behaviour parts directly related to service features, e.g. the knowledge of the chain "link" pattern which describes the service feature to be performed (see [11]). Such specification place-holders are represented by applying the *virtual procedure* construct.

The process type *GSLType*, instead, constitutes the specific model of the GSL for a given service. It inherits the process type *GenericGSLType* finalizing, for that service, all the virtual procedures.

Similarly to the block *BasicCall*, the process *SIB* is an instantiation of the process type *SIBType* whose actual specification is provided within the common package (see Figure 5). *SIBType* constitutes the general model of a SIB which can perform a given set of "SIB operations", e.g. all those described in the CS1 (see [13]). Each single SIB operation is represented as a procedure (*Operation1*, *Operation2*, ..., *OperationN*).

Since BCP, GSL and SIB are modelled as processes, communication among them are naturally represented by input/output signals. These signals carry as parameters all the needed informations to be exchanged, i.e. POI, POR, and "logical end". Hence POI, POR, and "logical end" are modelled by means of appropriate data structures. Also CID, SSD are modelled as data structures defined at SDL system level. In this way they can be accessed and manipulated by all processes in the system specification, i.e. *BCP*, *GSL* and *SIB*.

For the reader's convenience, Table 1 summarizes the mapping between IN service concepts and SDL language constructs envisaged so far.

**Table** 1: A Service-Oriented Semantics for SDL-92

| **IN Service Semantics** | **SDL Construct** |
|---|---|
| service | system |
| basic call | block type |
| service features | block |
| BCP | process |
| GSL | process (type) |
| SIB | process (type) |
| SIB operation | procedure |
| BCP/GSL/SIB interaction | signals (in/out) |
| POI, POR | data type (literals) |
| logical end | data type (struct) |
| CID, SSD | data type (struct) |

## 4 Modelling of IN Services

Starting from the associations established by the service-oriented semantics presented in the previous section an SDL model for IN services within the GFP is derived. This model consists of the package *BasicGFP* (see Figure 6), which collects characteristics that are common to all IN services and provides a number of generic structures that can be specialized for the purposes of specific services, and the system *INService* (see Figure 7), which actually provides the intended IN service specification.

*BasicGFP* contains the specifications of the IN basic call, the SIB, the generic GSL, and the definitions of a number of "common" data types and signals definitions. Among these data types two *literals*, namely *POIType* and *PORType*, are introduced to model the nine POIs and the six PORs identified for the CS1 (see [13]). In order to model the "logical end", CID and SSD, instead, the structured sort concept *struct* is used.

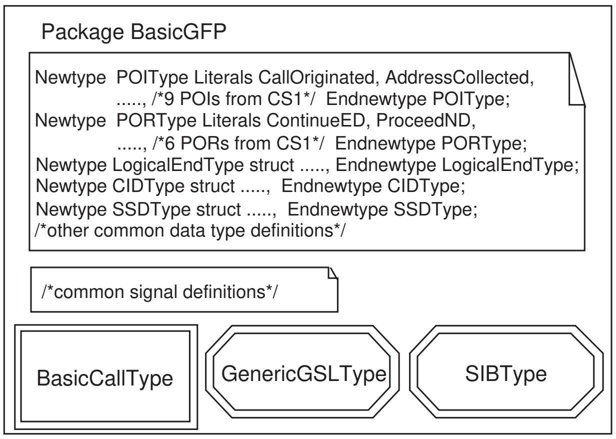


**Figure** 6: Common Package BasicGFP

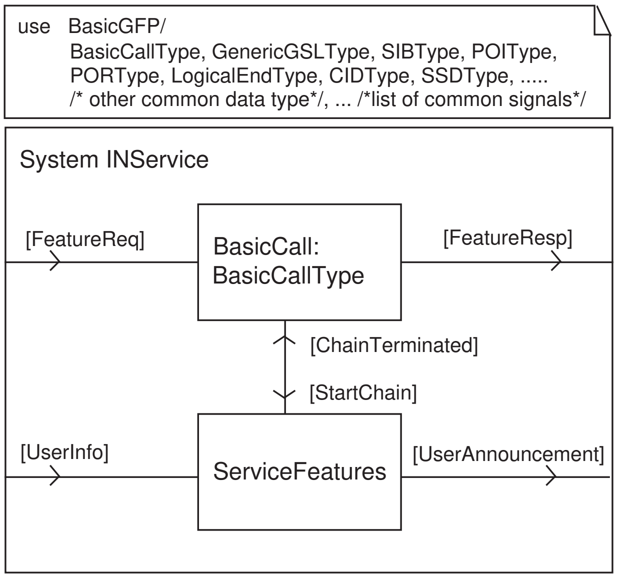


**Figure** 7: System INService

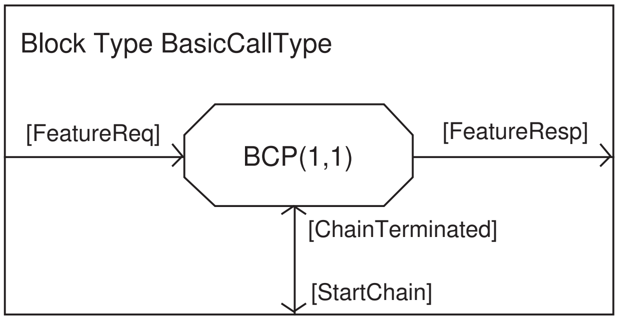


**Figure** 8: Block Type BasicCallType

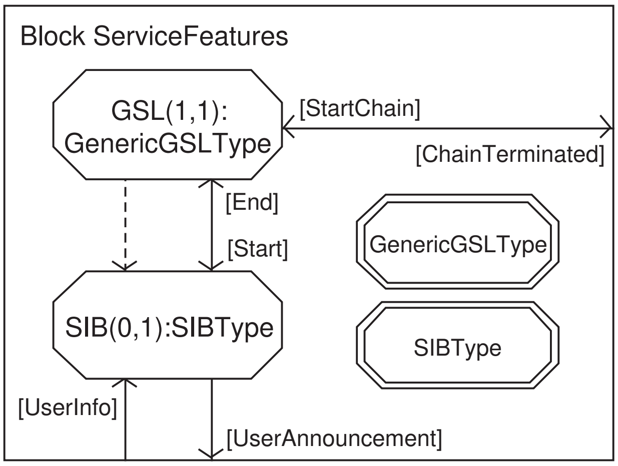


**Figure** 9: Block ServiceFeatures

All specifications and definitions contained in *BasicGFP* are made available for *INService* by means of the *use* clause.

Within *INService* the block *BasicCall* receives requests to perform a feature from the environment (signal *FeatureReq*), thought as the service user, processes it, and informs the environment for the actual execution of the feature or for the fact that the feature is not currently supported (signal *FeatureResp*). Moreover, it realizes all the needed basic call capabilities and demands the execution of the requested feature to the block *ServiceFeatures* (signal *StartChain*), which builds the related chain, manages its running, and informs back for its termination (signal *ChainTerminated*). *ServiceFeatures* can directly interact with the service user by means of signals *UsersInfo* and *UserAnnouncement* which convey the service feature related informations.

The block type *BasicCallType* and the block *ServiceFeatures* are refined as shown in Figure 8 and Figure 9, respectively.

Within the block *ServiceFeatures* the process *GSL*, which has one instance only created at start-up time, acts as a "main module". It coordinates and delegates operations to the dinamically created process *SIB*. In particular, for each single operation, that is required to accomplish a given feature, a process *SIB* is created. On the receipt of signal *Start* the process *SIB* selects the required operation and begins its execution. When the operation is completed it informs the process *GSL* by means of the signal *End* and stops. It should be noted that signals *UsersInfo* and *UserAnnouncement* will be used just in case the required operation is a "User Interaction" (see [13]).

## 4.1 Model of BCP

The BCP does not depend on any particular IN service. According to this assumption the process *BCP* is fully identified and its specification is made ready to be reused in the modelling of all IN services. The core specification of its behaviour is shown in Figure 10.

When the request to perform a service feature is received (signal *FeatureReq*), it is checked (procedure *CheckFeature*). If the service feature is inactive, the request will be discarded and the signal *FeatureResp* with the appropriate result will be issued. On the contrary, the request will be processed (procedure *CallOperation*), that is:

- the necessary basic capabilities are activated;
- CID and SSD are built;
- the need of a chain is evaluated and
- the relevant POI is envisaged.

If a chain is needed, it will be invoked providing the due POI (signal *StartChain(POI)*). Otherwise, the requested feature is considered performed.

The receiving of the signal *ChainTerminated(POR)* will cause again the call of the procedure *CallOperation* according to the carried value of *POR*.

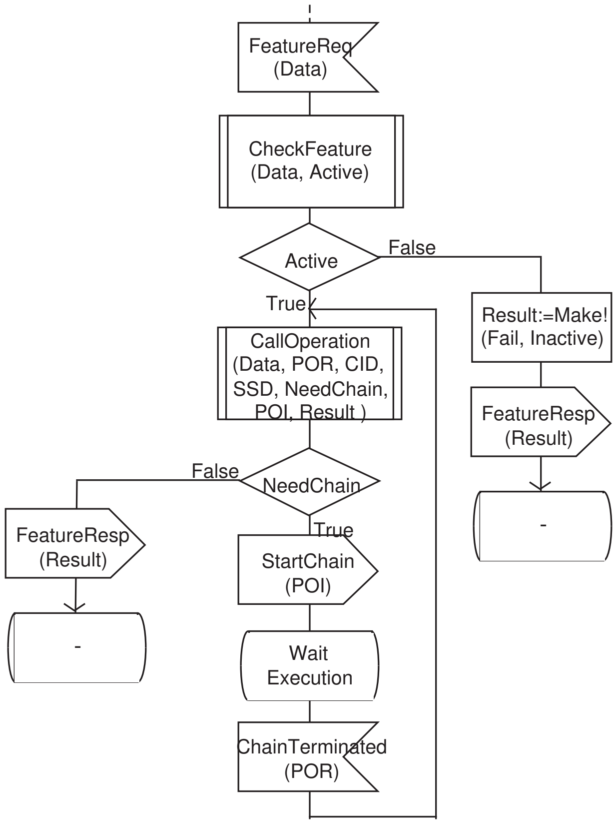


**Figure** 10: Core behaviour of process BCP

## 4.2 Model of GSL

The GSL is the sole entity within the GFP which expressly depends on the IN services to be modelled. This dependency is described by introducing two virtual procedures, namely *ProcessPOI* and *ProcessEnd* (see below), which can be suitably finalized in order to satisfy the needs of specific services. These procedures constitute the only part of the entire IN service model that is required to be specified from scratch.

A main fragment of the behaviour of the process *GSL* is given in Figure 11. When the process *GSL* receives a request to start a chain with the associated POI (signal *StartChain(POI)*), it processes such request, taking the SSD and the CID as input parameters, in order to determine the SIB operation to execute (procedure *ProcessPOI*). Then the process *SIB* is created with the specific task to accomplish the envisaged operation. The signal *Start* triggers the actual operation execution off. On the receiving of the signal *End(LogicalEnd)* the procedure *ProcessEnd* is called, taking the *CID* and the *LogicalEnd* as input parameters, in order to determine the POR and the next operation to execute. In case no other operation are needed to complete the chain, the signal *ChainTerminated(POR)* is sent.

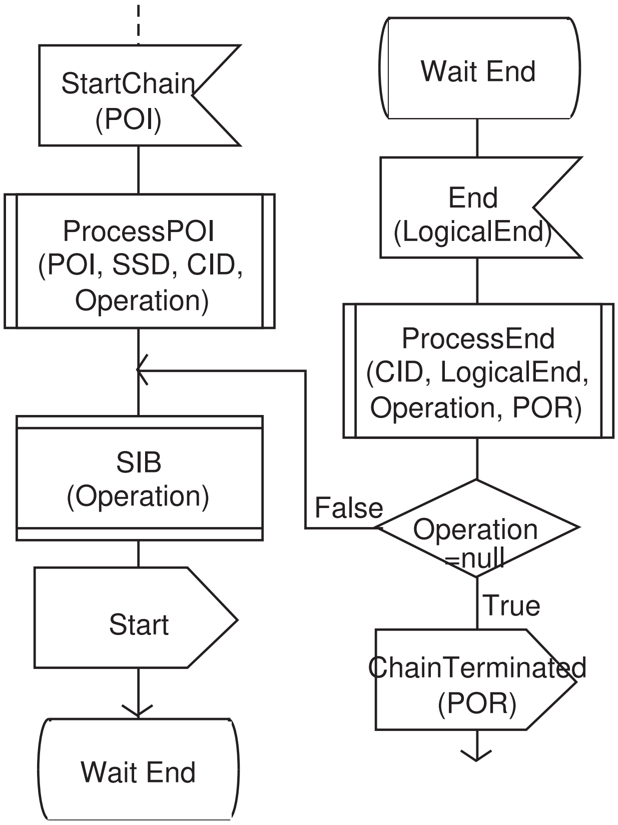


**Figure** 11: Core behaviour of process GSL

## 4.3 Model of SIB

Similarly to the BCP also the SIB does not depend on any particular IN service. Therefore, the process *SIB* is fully identified and its specification is made ready to be reused in the modelling of all IN services. The core specification of its behaviour is shown in Figure 12. In practice, it consists of a number of procedures (one for each possible SIB operation) and a simple mechanism for selecting the particular procedure to be executed.

For the CS1 thirteen SIB operations have been identified(see [13]). They are: Algorithm, Charge, Compare, Distribution, Limit, Log Call Information, Queue, Screen, Service Data Management, Status Notification, Translate, Verify and User Interaction.

Each procedure takes the SSD as input parameter and CID and LogicalEnd as input/output parameters.

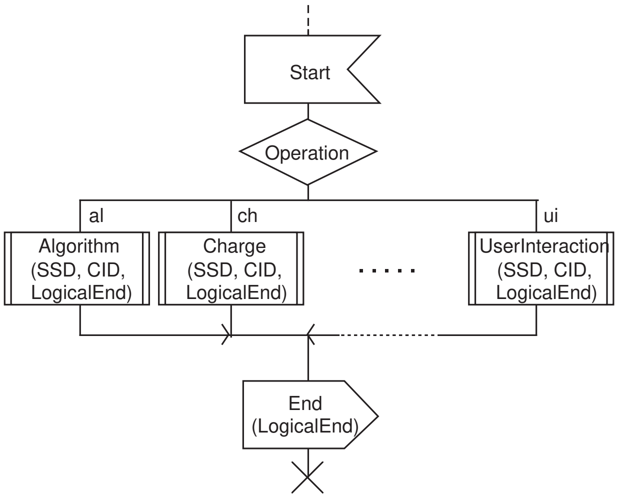


**Figure** 12: Core behaviour of process SIB

## 5 An Application Example

The modelling approach presented here has been exercised in the specification of part of the Universal Personal Telecommunication (UPT) basic operations [14]. The complete work can be found in [15].

UPT has been selected for its being a major representative of a new generation of services which exploit in depth the Intelligent Network concepts. It was therefore a good candidate to illustrate advanced IN service modelling.

### 5.1 The UPT Service

UPT is based on the innovative concept of *personal mobility*, which is meant to overcome the paradigm of fixed correspondence between terminal and user identification typical of current generation of mobile networks.

In the UPT environment, since the association terminal and user identification is removed, users can place and receive calls, or in more general terms access telecommunication services, using *any terminal at any location*: all they need is a user identification number (*UPT number*). In other words, identification of users is performed separately from addressing of terminals. UPT numbers can be dialled on a global basis and routed from any fixed or mobile terminal, across multiple network irrespective of geographical location. Nothing hinders from assigning a UPT user more than one UPT number, very much like a person can have multiple credit card accounts.

UPT also allows the UPT user to participate in a user-defined set of subscribed services, from amongst which the user defines personal requirements, to form a UPT service profile. The actual use on a call basis of services and capabilities registered in a UPT *service profile* is only limited by terminal and network capabilities and restrictions imposed by the network operator.

UPT standardization is being carried out in a phased approach. In the first phase a restricted short-term scenario is considered where only the telephony service is supported: the so-called UPT *Service Set 1* [14] includes a limited set of features which are categorized into *essential*, i.e. mandatory in any UPT implementation, and *optional*, i.e. just meant to offer enhancements to the basic UPT operations.

### 5.2 Specification of Essential UPT Features

In the work referenced in this paper, only three essential UPT features have been described, namely[2]:

- *User identity authentication*, i.e. a procedure by which the service provider can verify the claimed identity of a user;
- *InCall registration* allowing the user to indicate the terminal address where incoming calls are to be presented;
- *Outgoing UPT call* which enables the user to initiate, from any terminal, an outgoing call.

Since the primary purpose of the work was not to provide a complete and faithful specification of the CS1, but to explore the feasibility of the proposed methodology, some working assumptions during the modelling activity in SDL have allowed on one hand to abstract away from details, that would only add complexity without specific substance, and on the other hand to better highlight some aspects of the methodology.

#### 5.2.1 Modelling the BCP as an SDL process

When it comes to the behaviour specification of the BCP, the recommendation lends itself to a number of different interpretations as regards some aspects and leaves open the choice of some non-neglectable details. In other words, the SDL designer is faced with the need to "complete" from a substantial viewpoint the informal specification provided in the recommendation, before he can proceed to the formal model.

[2]In this section the words 'service provider', 'subscriber' and 'user' should be meant as prefixed by the UPT keyword.

More precisely, to avoid "invention of procedural details" by the SDL designer, plain call processing (i.e. connecting and disconnecting calls) has only been referenced but not specified. On the contrary, the way the BCP establishes relations among the different elements within the Global Functional Plane has been fully described.

#### 5.2.2 Modelling of SIB operations as SDL procedures

In addition to the SIB operations identified for the CS1 [18], two more operations, namely *UPT registration* and *UPT authentication*, have been devised and specified. The procedures describing these operations are mainly based on the definition of suitable data structures for the subscriber list, the terminal list, the association between user and terminal, etc., and primitives like "insert", "remove" and "search" acting on such data.

#### 5.2.3 Modelling of CID and SSD as data types

To reduce the complexity of the specification, only an essential subset of all possible CID and SSD identified for the CS1 [18] has been tackled. Nevertheless, all UPT relevant data like, for example, calling/called user access code, user PIN code, terminal identifier, deregistration time, etc., have been examined and formally described.

### 5.3 Tool Support

The specification has been carried out with the assistance of the SDT tool [7]. This tool has also allowed:

- to check the specification against correctness of the syntax and the static semantics, on a per SDL diagram basis;
- to automatically build an executable SDL specification in C;
- to exercise the executable specification, by feeding suitable input signal suites and checking the actual output against the intended one;
- to produce a number of MSCs (Message Sequence Charts) [16] by means of a special editor included in the tool.

Since the SDT version 3.0 does not fully support SDL-92, some object-oriented pieces of specification had to be processed manually into equivalent non object-oriented pieces, before resorting to the tool. However — as soon as a new version of the tool becomes available — the above "good pieces" of specification can be directly used.

## 6 Conclusions

The paper has been advocating the advantages of a formal and object-oriented methodology for the creation of services based on the IN paradigm and, possibly, also on a long-term architectural approach [17]. The methodology is strictly based on the assets of SDL-92 as a powerful specification and development tool.

The focus of the paper has been on the Global Functional Plane of the INCM. For this plane the various problems associated with the relationships among logical entities contained in it , i.e. SIB, BCP, GSL, have been studied and a service oriented semantics for SDL-92 has been proposed. In particular, the paper has provided a comprehensive set of guidelines prescribing how to best use the constructs of the language in the specification of: IN services, IN service features, BCP, GSL, SIB, SIB operations, POI, POR, logical end, CID and SSD. In other words, the proposed guidelines propose a "recommended" way to use SDL-92 for IN service modelling. The value of the guidelines has been assessed through a specification exercise, which gives encouraging indications for a final tuning of the methodology .

As a tangible result of the work, major elements that are common to all IN services have been modelled and gathered in an "SDL package", that could be seen as a set of templates from which the service designers may want to start the specification of any new service. Most templates are based on the use of specification place-holders, basically represented by means of SDL virtual procedures. In principle all a service designer needs to do, is to "finalize" (in the meaning that this term has in SDL) a few virtual procedures and perhaps, specify some additional data types.

Future work would be of great benefit in these areas:

- formal modelling in the SDL world of the Service Plane and the Distributed Functional Plane of the INCM;
- formal modelling of the interactions between the different planes of the INCM;
- extension of the methodology proposed in this paper, towards long-term service architectures.

## References

[1] ITU-T, *Recommendation Q.1200* , "Q-Series Intelligent Network Recommendation Structure", March 1993.

[2] ITU-T, *Recommendation Q.1201* , "Principles of Intelligent Network Architecture", March 1993.

[3] G.Bruno, J. Insulander, U. Larsson, F. Lucidi, "A Service Driven Vision of Integrated Broadband Communication: the OSA Approach", *LNCS: "Towards a Pan-European Telecommunication Service Infrastructure - IS&N '94"*, n. 851, Springer-Verlag, Berlin, pp. 529-538, 1994.

[4] J. Hallstrand, D. Martin, "Industrial Requirements on a Service Creation Environment", *LNCS: "Towards a Pan-European Telecommunication Service Infrastructure - IS&N '94"*, n. 851, Springer-Verlag, Berlin, pp. 17-25, 1994.

[5] J.J. Garrahan, P.A. Russo, K. Kitami, R. Kung, "Intelligent Network Overview", *IEEE Communication Magazine*, March 1993, Vol. 31. N. 3, pp. 30-36.

[6] ITU-T, *Recommendation Z.100* , "CCITT Specification and Description Language (SDL)", March 1993.

[7] Telelogic AB, *SDT* , Box 4128, S-203 12, Malmo, Sweden.

[8] Verilog SA, *GEODE* , avenue Aristide Briand, 92220 Bagneux, France.

[9] E. Cancer, R. Mc Cam, M. Abondharam, "IN Roll-Out in Europe", *IEEE Communication Magazine*, March 1993, Vol. 31. N. 3, pp. 38-47.

[10] M. Appeldrom, R. Kung, R. Saracco, "TMN + IN = TINA", *IEEE Communication Magazine*, March 1993, Vol. 31. N. 3, pp. 78-85.

[11] ITU-T, *Recommendation Q.1203* , "Intelligent Network - Global Functional Plane Architecture", March 1993.

[12] A. Nyeng, B. Moller-Pedersen, "Approaches to the specification of Intelligent Network Services in SDL-92", *Proceedings of the Sixth SDL Forum*, Darmstadt, Germany, October 1993, North Holland, pp. 427-440.

[13] ITU-T, *Recommendation Q.1213* , "Global Functional Plane for Intelligent Network CS-1", March 1993.

[14] ITU-T, *Recommendation F.851* , "Universal Personal Telecomunication (UPT) - Service Description (Service Set 1)", February 1995.

[15] A. Tosti, "Analisi e progettazione del software di controllo e di gestione di servizi multimediali", *Tesi di laurea*, Universita degli Studi di Roma 'La Sapienza', Italy, December 1995.

[16] ITU-T, *Recommendation Z.120* , "Message Sequence Chart (MSC)", September 1994.

[17] G.Nilsson, F. Dupuy, M. Chapman, "An Overview of the Telecommunication Information Networking Architecture", *Proceedings of TINA 95*, Melbourne, Australia, February 1995, pp. 1-12.

[18] ITU-T, *Recommendation Q.1219* , "Intelligent Network User's Guide for Capability Set 1", April 1994.