1

A Generic Component Model for the design of future telecommunication services

Xavier Logean, Jean-Pierre Hubaux*, Simon Znaty◇*

**Telecommunication Services Group, TCOM Laboratory*
Swiss Federal Institute of Technology, Lausanne
email: {logean, hubaux}@tcom.epfl.ch
URL: http://tcomwww.epfl.ch

◇Network and Service Department, ENST Bretagne, Rennes
email: Simon.Znaty@enst-bretagne.fr

Abstract

This paper proposes a generic component model which is a common software architecture applicable to the design of every component of a distributed telecommunications service. The concepts, principles and rules behind this model are introduced, and its architecture is detailed. It is then applied to the design of a broadband virtual private network service.

Keywords

TINA, Multimedia Service Specification and Management

1 INTRODUCTION

Nowadays the telecommunications market is characterized by a fast evolving technology, a multiplicity of providers and the need for value-added services such as Multimedia Services. Such trends promote rapid introduction of new and enhanced services and their management. Telecommunications services have requirements in terms of *reusability* for their quick provision, *interoperability* to alleviate service interactions management, *distribution* -a service consists of a set of components that interact with each other-, and *manageability* since control and management of services should be integrated within this service. To meet these requirements, a high abstraction level of services and systems is necessary.

The work* presented in this paper aims at defining a Generic Component

*This work is part of a project called Ernes*TINA* partially funded by Swiss Telecom PTT.

Management of Multimedia Networks and Services R. Boutaba & A. Hafid (Eds.)

Model (GCM) used to build distributed telecommunication services . Genericity enables the model to be reusable for the design of any component, and therefore speeds up its introduction. Moreover, every component provides the same skeleton or template, therefore providing the basis for an easy treatment of the service interaction management problem.

The IN (ITU-T Q12xx 1993) Service Independent Building Block (ITU-T Q1213, 1994) is a type of generic component; it exists only in one plane of the IN conceptual model and implements a very simple functionality. The INA building block (Rubin and Natarajan, 1994), ODP cluster and capsule (ISO X901, 1993) belong to this category of generic software components, but are too low level to be directly used for designing services. In ROSA (Race R1093, 1992), CASSIOPEIA (Race R2049, 1995) and TINA (Berndt *et al.*, 1995) (Chapman and Montesi, 1995) generic service component architectures have been proposed but are yet to be enhanced for applicability. TINA proposes a generic component model called Universal Service Component Model (USCM) (Brown 1993). Using this model, all services or service components are described as consisting of a core surrounded by an access layer. The access layer of a service is divided according to three access aspects namely usage, management and substance. These divisions are referred to as sectors.

1. The *Core* comprises the logic and data that define the intrinsic operation of the service.
2. Components in the *Usage* sector provide interfaces for the external components or services that motivate and directly control the operation of the service (e.g., clients or users of the service).
3. *Substance* sector components are the internal representation of the external components that the service uses.
4. Components in the *Management* sector provide the logic and data management to control the initialization, configuration, accounting, and other management functions needed to establish, configure, and characterize the service.

The TINA consortium has already finalized the work on the USCM, but no emphasis has been put on the issue of applying the USCM to service components definition and specification, nor on how to relate these specifications to the computational modelling concepts (Abarcan *et al.*, 196).

In this paper we present a Generic Component Model which specifies and enhances the TINA USCM by:

- the definition of generic primitives for the access sectors of the component,
- the unification and specification of the internal structure of the component,
- the unification and specification of the organization of the internal data of the component.

Although the GCM is based on the TINA USCM, it is not restricted to the TINA world, and can be used for the design of any distributed application.

Section 2 introduces the GCM and its underlying concepts. Section 3 illustrates the use of the GCM for the several types of TINA service components. Finally, the GCM is applied to the design of a broadband-virtual private network service in a TINA-like platform called OAMS (Open Service Architecture for Multimedia Services over ATM) (Znaty, 1996) which is our TINA testbed.

2 GENERIC COMPONENT MODEL

Management of network resources has become a crucial stage in the creation and use of open, object-oriented, generic and transparent services. Network Management allows to keep the separation between a service, its management and the network resources. Systems management involves managing and managed systems (Hebrawi, 1995). A Managed system is itself divided into an *Agent* process and *Managed Objects* (MOs) which are stored in a Management Information Base (MIB) (Black, 1992). The Agent process is in charge of offering to the Manager a set of operations to be performed on the MOs and therefore hides the implementation of the MOs. The MOs provide a data representation of the resources for the purpose of management (i.e., network resources, software). The Agent translates the Manager requests which are CMIS-based or SNMP-based into methods offered by MOs.

The GCM presented in this paper follows the same philosophy, i.e., the clear separation between the way to act on a system and its implementation. Any component in a distributed system can be divided into two parts; on the one hand, the service offered to the "outside world" and on the other hand the data, operations and functions specific to the component (Brown 1993). Similarly, each aspect (usage, substance, and management) of the GCM has the following structure: an *Agent* which is the interface to the outside world and an *Information Base* (IB) which contains all the data and internal functions of a given aspect of the component. The GCM is therefore compliant with the TINA USCM which introduces the notions of core and access layer. Each of the sectors encapsulates an agent while the core of the component provides one IB per sector. All the data and internal functions of a component are contained in the corresponding IB: Usage IB (UIB), Management IB (MIB) or Substance IB (SIB). Figure 1 presents an OMT (Rumbaugh *et al.*, 1991) meta- information model of the GCM. Without going into all details, we can see from this figure that the GCM is made up of three parts (Usage, Management, Substance). A set of GCMs can interact with each other to provide a service (cooperative approach and association relationship), as well as a set of GCMs can be grouped into a global GCM (integrated approach and containment relationship) to provide a service functionality. With such properties, a GCM may represent a large scale, complex service component or service,

as well as a small simple one. Whatever size or complexity of the service, its structural organization is consistent with the GCM division, and therefore compliant with other services or service components. Figure 1 illustrates the

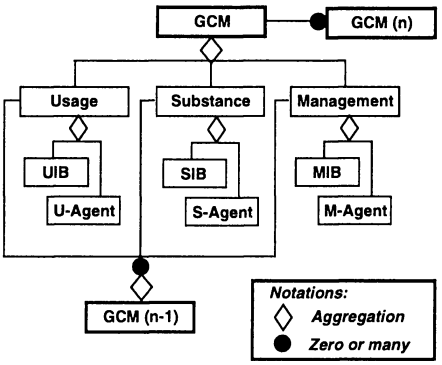

Figure 1 OMT diagram of a Generic Component Model

GCM structure. A Generic Component at level(n) (where letter "n" stands for the level of the component) can be composed of multiple components at level($n-1$). Similarly, a GCM can be associated with other GCMs at level(n). Each part of the GCM consists of an *Agent* and an *IB*. The next two sections detail the generic aspects of each sub-component of the GCM. Section 2.1 presents the generic Agent of a GCM, while Section 2.2 describes its generic IB.

2.1 Generic Agent

A GCM Agent is characterized by its own *Architecture* (internal view-core) and the *Service* it provides (outside view-access sector) (Figure 2). The Service Part is subdivided into *Basic Services* and *User Defined Services.*

The *Agent Services* defines the minimal functionality that a component should offer. The associated primitives are compliant with CMIS (Common Management Information Service) requests. These are Set, Get, Create, Delete, Action and EventReport (Black, 1992) . Any operation can be achieved using one or a set of these primitives. The Set, Get, Create and Delete primitives allow simple operations on objects. The Action primitive encapsulates all the other operations, except events generated by the component which are carried through the EventReport message. The *User Defined Services* are services that other components can add and use. User Defined Services may be seen as supplementary services that are used to customize a given service. For example, specific security features may be implemented as User Defined Services. Service management tasks may also be delegated to a given service component

through its management sector as a User DefinedService, where the User is in fact a manager. The *Architecture* consists of two entities, namely *Connection*

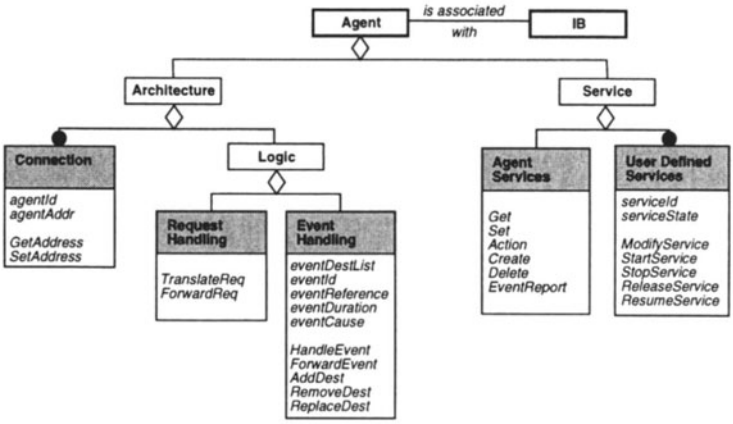

Figure 2 The Generic Agent of the GCM

and *Logic*. The role of the entity *Connection* is to maintain the Address of each internal Agent (according to the "composition" property, an agent may be itself a GCM). If a request cannot be handled by the component, the Usage Agent forwards it to the Substance Agent whose responsibility it is to find a service component that is able to perform the requested functionality. The entity *Logic* handles the Internal (Event) and External Requests and forwards them to the appropriate destination which may be an external component or a sub-component of the GCM. Some parts of the GCM are optional. For example, the Substance sector is different from the other parts; it is a one-way request access. Through the Substance access sector, requests can only be issued. Therefore, there are no User-Defined Services and Connection entities for a Substance Agent. Each Agent has access to its own IB only, unlike the Management Agent which is able to get information from any IB, in order to maximize the information available to the manager.

2.2 Information Bases

Similar to the generic agent, the structure of the generic IB is subdivided into the entities *Architecture* and *Service*. Figure 3 shows a meta information model of an IB.

The *Architecture* includes the internal information of the component and the entity *Service* contains the information associated with the services provided by the component. The entity *Architecture* is further decomposed into the classes *Management* and *Entity*. All the dynamic data can be found in

the *Entity* class. The static data are contained in the *Management* class. For example, the data specified at the initialization, such as version, name, configuration parameters, etc., are considered as static data. Data that are changed on a real-time basis are identified as dynamic data. The entity *Service*

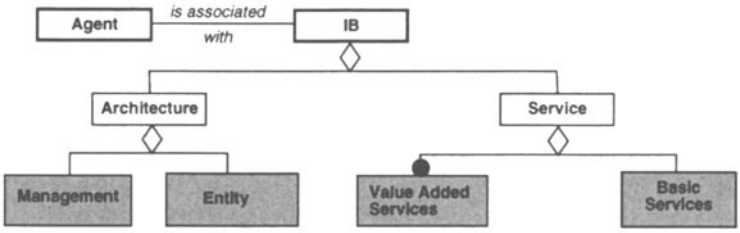

Figure 3 The Generic IB of GCM

is subdivided into the class *Basic Services*, which contains the information about either requests/responses or streams, and the class *Value-Added Services* which is specific to the service offered by the component -its specific functionality, e.g., authentication component-.

Since the IBs (data) are specific for each part of the component, only the 'structure' is generic. The next three subsections detail the UIB, MIB and SIB.

2.3 Usage Information Base (UIB)

The *Usage Information Base* (UIB) (Figure 4) contains the data associated to the usage services offered by the component (i.e., the services supplied to the outside world).

As mentioned in Section 2.2, the *Management* part of the UIB collects the static data, configured at the initialization of the component. The *Entity* part deals with the dynamic data: time since the initialization and several status of the component (operational, usage and administrative states). The *Entity* class is associated with the *Internal Flow* class which holds the information relative to the internal request of a component (e.g., requests forwarded from the Usage Agent to the Substance Agent). The *Value Added Services* part is concerned with the information specific to the functionality of the component. One component could be designed for authentication, and will therefore manage cryptographic algorithms, password lists, etc. Each Value Added Services class is associated with a *Session* (Abarcan *et al.*, 196) class with data such as associated service, session identifier, session participants, etc.

The entity *Basic Services* is subdivided into classes *Flow Service* and *Operational Services*. In a distributed environment a component offers interfaces through which interactions will occur with other components. Depending

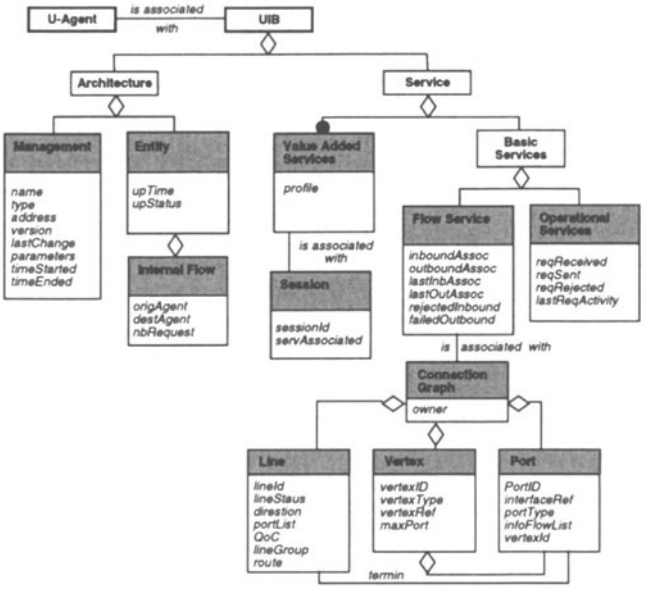

Figure 4 The UIB of GCM

on the nature of interactions that will occur, the interface is classified as being either an *operational interface* or a *stream interface*. The interactions that occur at an operational interface are structured in terms of invocations of one or more operations and responses to these invocations. When objects interact via stream interfaces, the information exchange occurs in the form of stream flows between the components, where each stream flow is unidirectional and is a bit sequence with a certain frame structure and quality of service parameters. The separation between operational and stream interfaces is clearly made at the computational level in TINA (Natarajan *et al.*, 1995). The information concerning the requests, number of requests received/sent, rejected, etc., are found in the *Operational Services* class. The *Flow Service* contains the data specific to the stream flows exchanged between two components. Similar data have been identified in the Internet world in the Request for Comments on Network Services Monitoring MIB (Kille *et al.*, 1994).

In a distributed environment, each association between two components can be modeled as a *Logical Connection Graph* (LCG), where vertices represent the component, ports represent the stream interfaces and lines represent the streams (Liccardi, 1994). Therefore, each *Flow Service* is associated with an LCG specifying all the relative data: ports used, Qos of the line, etc. The information encapsulated in the LCG could be considered as management information.

2.4 Substance(SIB) and Management Information Base (MIB)

The role of the substance sector of a GCM is to provide the component with an outside view. The component can access the outside world through this sector. Therefore, in the Management part of the SIB, data specific to the other components can be found such as, name, list of operations available, etc.

Only the data specific to management holds in the MIB. The information encapsulated in the different classes of the MIB are those relative to management requests and dynamic data (Entity class).

3 USE OF THE GCM

In a distributed environment, a service is based on the interaction and cooperation of many components. The GCM presented in this paper gives the skeleton to design any component in a distributed environment. In this section, we first present the use of the GCM for designing TINA-like components. To help the designer of the service, TINA offers a set of components (Berndt *et al.*, 1995) and specifies their functionality. In Section 3.1 we present the use of the GCM to design those components. Section 3.2 depicts the use of the GCM to design components of a VPN in a distributed platform based on CORBA.

3.1 Using the GCM in a TINA environment

(a) TINA computational viewpoint

The TINA Computational modelling concepts (Natarajan *et al.*, 1995) define the rules of how computational objects interact with one another. Computational objects are the units of programming and encapsulation. Objects interact with each other by sending and receiving information to and from interfaces. An object may provide many interfaces, which may be of different types: operational and stream. The goal of the designer is to focus on the objects required by specifying the interfaces they offer and what interfaces of other objects are required. The notation chosen for computational specification is called *TINA ODL* (Object Definition Language) (Kitson *et al.*, 1995).

The TINA Service Architecture gives a set of service components that is divided as follows:

1. Independent Service Objects
2. Dependent Service Objects

3. Specific Service Objects

The first kind of components are identical and mandatory for any service; they are also specified by TINA in ODL. Applying the GCM to such components is therefore not of great interest, except for the validation of the GCM concepts in the TINA world. The second kind of components are needed within each service but are dependent of the service. They contain generic operations but also operations and data specific to the service. The genericity of the GCM is helpful for the design of those components, especially concerning reusability. The third kind of components are specific to the service (e.g., an authentication service, videoconferencing service). Applying the GCM to design such components warrants their compliance to TINA and their interoperability with the other TINA components.
Figure 5 shows how the GCM may be applied to each category of service components. The GCM will be used to design the *TINA Independent Service Objects*. Then these components will be directly instantiated at service run time. Therefore, they will not be reused but supplied within a service creation environment for the design and implementation of a TINA service. The *TINA Dependent Service Objects* will also be designed using the GCM. However in this case, the derived objects will be reused by the service designer. The *TINA Specific Service Objects* will be produced by the service designer directly from the GCM. In this case the GCM is the component to be reused to derive all the specific objects of the telecommunication service.

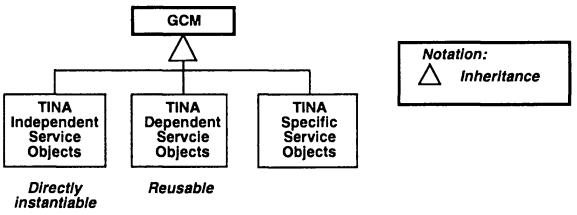

Figure 5 Inheritance tree of components

(b) mapping of the GCM to ODL

The mapping of the GCM to TINA ODL (as shown in Figure 6) is facilitated by the intrinsic features of the GCM: the clear separation between the internal operations and the services offered to the other components . This separation allows a simple identification of the interfaces. Indeed, the *Service* part (*Agent Services* and *User Defined Service*) of the Agents of the GCM can be directly mapped to operational interfaces. The Architecture part of the Agents as well as the IBs identify the different attributes of the GCM. ODL has been

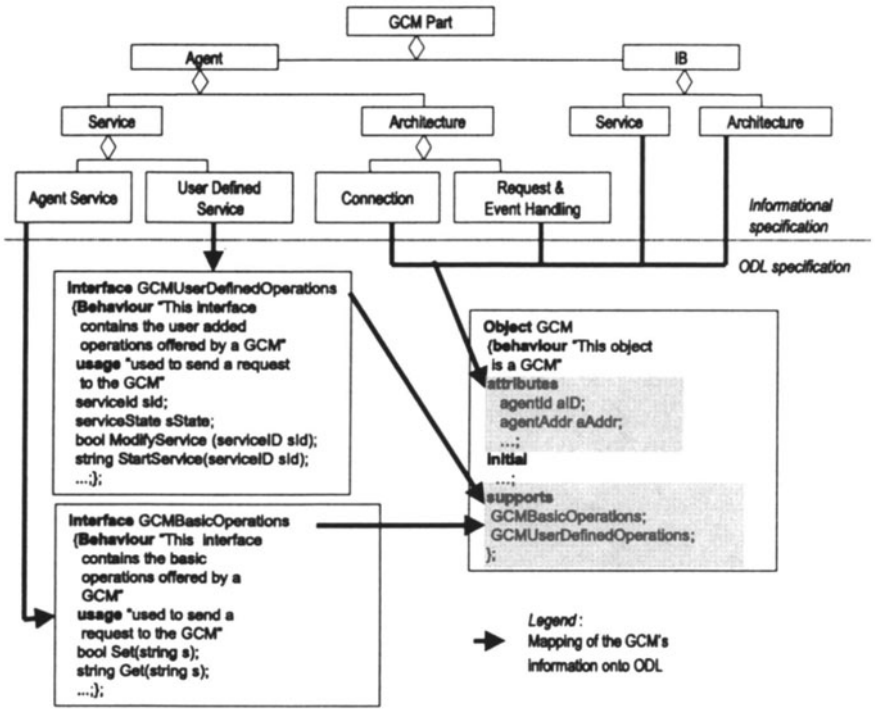

Figure 6 Mapping of GCM informational models onto ODL

designed to capture the distributed concepts, therefore only the interfaces and attributes interesting from a point of view of distribution are specified.

The GCM also gives information on the internal part of the component and proposes a set of internal operations which are not present in ODL. Using these internal information might ease the implementation of the computational objects. This information could complete the skeleton built by the translation of TINA ODL to any implementation language (Hooke, 1996).

3.2 Using the GCM in a TINA-like environment for the design of a Broadband Virtual Private Network Service

In this section, we show how to apply the generic component model for the design of a Virtual Private Network (VPN) . We consider an underlying architecture called OAMS (Open Service Architecture for Multimedia Services over ATM) (Znaty, 1996) which is a TINA-like management architecture where telecommunications services are deployed. As depicted in Figure 7, OAMS consists of four levels, telecommunication service architecture, management architecture, distributed computing environment and ATM communication platform. OAMS embraces TMN (ITU-T M3010, 1992) within a framework

based on Open Distributed Processing (ODP) (ITU-T X.901-X.904, 1995) principles and object orientation. Currently, the management architecture

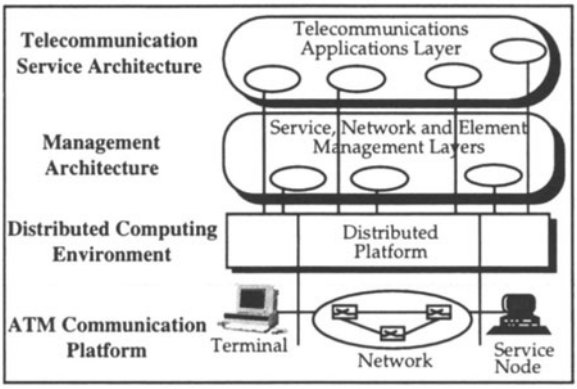

Figure 7 OAMS architecture

consists of a connection management architecture which enables the establishment, control and release of connections with long holding times, i.e., of semi-permanent or permanent nature. The VPN provides corporate networking between geographically dispersed customer premises using the public switched networking infrastructure, in our case, an ATM network. The VPN service can be seen as a distributed application running on the multiple nodes of a telecommunication network. The VPN service and its management services will consist of a set of service components all deriving from the GCM and present with the OAMS telecommunication service architecture. For example

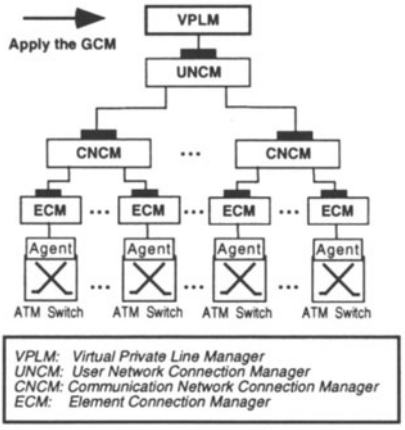

Figure 8 The VPLM within the OAMS connection management architecture

the VPN configuration management service will enable the VPN customer to dynamically modify the configuration of his VPN, i.e., add or remove logical links between the different sites of his organization. The VPN service consists of a set of computational objects (CO). The CO which is the interface to the VPN customer is called Virtual Private Line Manager(VPLM). It allows the customer to establish, control and release virtual lines. All the other VPN COs are supplied by the OAMS connection management architecture. The CO that the VPLM makes use of is called User Network Connection Manager (UNCM) and provides the end-to-end connections that are the physical representation of the virtual lines. These end-to-end connections are then mapped into subnetwork connections established by communication network connection managers (CNCMs); subnetwork connections are finally translated into sets of cross-connections setup within switching elements (Figure 8). In the current OAMS architecture, UNCM, CNCMs and ECMs are not implemented using the GCM approach. Figure 9 shows the VPLM as a GCM. The VPLM

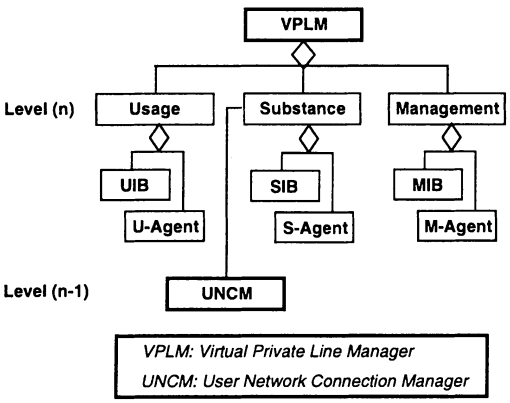

Figure 9 The Virtual Private Line Manager

U-Agent enables the establishment of virtual lines, their release and modification by the customer thanks to CMIS M-CREATE, M-DELETE and M-SET operations. Data about these virtual lines may be read through an M-GET request; the customer may be informed about problems via the M-EVENT-REPORT primitive. Therefore the U-Agent of the GCM already provides all necessary functionalities. The UIB provides the concept of connection graph with lines and ports. These concepts are useful and sufficient for the modeling of a graph which consists of a set of virtual links and VPN service access points which represent the usage data. The VPLM S-Agent has access to the services provided by the UNCM for the establishment, control and release of end-to-end connections thanks to the appropriate CMIS requests. Therefore it is a subset of the generic S-Agent. The S-MIB will contain all data concerning

the possible operations on the UNCM. It complements the GCM S-MIB with information specific to the end-to-end connections established by the VPLM and which serve the virtual lines. The VPLM M-Agent will be able to monitor and control the operation of the VPLM in terms of a number of statistics that will be processed by an external VPN manager. The VPLM MIB will contain all data such as counters, ratios which characterize those statistics. Therefore the VPLM MIB derives from the GCM MIB and enriches it with specific VPN management information.

4 CONCLUSION

In this paper, we introduced the concept of a generic component model (GCM) for the design of distributed telecommunication services with a focus on TINA-like services. The concept goes beyond TINA; it supports reusability, extendibility, maintainability and interoperability criteria expressing current telecommunication software requirements. For example, it may be used for the provision of CORBA-based telecommunication or management services (Saydam, Logean and Znaty, 1997). The concepts behind the GCM are mainly those of OSI network management with the notions of agent and MIB. These enable the structuring of the GCM. To move from concepts to realization, we have considered a VPN configuration management service and shown how to design such service using the GCM.

5 ACKNOWLEDGEMENTS

The authors would like to thank S. Koppenhöfer for constructive remarks which significantly improved the quality of this paper. We also thank F. Dietrich for many useful discussions.

REFERENCES

ITU-T Recommendation Q120X, Q121X, Q122X: The Intelligent Network (1993).

ITU-T Recommendation Q1213 Global Functional Plane for Intelligent Network CS-1 (1994).

H. Rubin, N. Natarajan (1994)A Distributed Software Architecture for Telecommunications Networks, IEEE Network.

ISO/IEC JTC1/SC 21/WG7, Draft Recommandation X901 (June 1993) Basic Reference Model of Open Distributed Processing - Part1: Overview and Guide to Use.

Race project R1093, 10th Deliverable (Dec 1992) The ROSA Architecture Release Two.

Race project R2049, 4th Deliverable (March 1995) CASSIOPEIA Open Services Architectural Framework.

H. Berndt, L.A. de la Fuente, P. Graubmann (Feb 1995) Service and Management Architecture in TINA-C, TINA'95, Australia, 81–86.

Chapman, M., Montesi, S. (Feb 1995) Overall Concepts and Principles of TINA, TINA-C, Version 1.0.

D. Brown (July 1993) The Universal Service Component Model,TINA Consortium, Document No. EN_B1.DKB.003_1.4_93.

C. Abarcan, et. al. (Oct 1996) Service Architecure, TINA Consortium, Document No. TB_RM.001_4.0_96.

J. Siegel (1996) CORBA Fundamentals and Programming, John Wiley.

Znaty, S. (July 1996) Service and Network Management in the OAMS Open Service Architecture, ACM Computer Communication Review.

B. Hebrawi (1995) GDMO Object Modelling & Definition for NETWORK Management, Technology Appraisals, UK.

U. Black (1992) Network Management Standards The OSI, SNMP and CMOL Protocols, MCGra-Hill, Inc.

J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen (1991) Object-Oriented Modeling and Design, Prentice-Hall.

N. Natarajan, F. Dupuy, N. Singer, H. Christensen (Feb 1995) Computational Modelling Concepts, TINA Consortium, Document No. TB_A2.HC.012_1.2_94.

S. Kille, N. Freed (1994) Network Services Monitoring MIB, IETF, Network Working Group.

C. A. J. Liccardi (1994) Eurescom Project EU-P103, Network Resource Model, Eurescom.

H. Berndt, et. al (March 1995) Service Component Specifications, TINA Consortium, Document No. TB_HK.002_1.0_94.

Kitson, B., Leydekkers, P., Mercouroff, N., Ruano, F. (June 1995) TINA Object Definition Language (TINA-ODL) Manual, TINA Consortium, Version 1.3.

N. Hooke. (Sept 1996) A CORBA-based ODL C++ language mapping, In proceedings of TINA'96, Heildelberg.

Principles for a Telecommunications Management Network (1992) ITU Rec. M.3010.

ITU Rec. X.901-X.904 Open Distributed (1995) Processing-Reference Model.

T. Saydam, X. Logean, S. Znaty (April 1997) A Service Management Architecture, International Conference on Telecommnicatons, ICT'97, Melbourne, Australia.