23

An Overview of the Telecommunications Information Networking Architecture

TINA Consortium
c/o Bellcore
331 Newman Springs Rd.
Red Bank, NJ 07701
USA
Tel: +1 908 758 2467
Fax: + 1 908 758 2865

1. INTRODUCTION

The Telecommunications Information Networking Architecture Consortium (TINA-C) is a worldwide consortium formed by network operators, and telecommunication and computer equipment suppliers. The consortium is aiming at defining and validating an "open" architecture for telecommunications services in the emerging broadband, multi-media and "information super-highway"/"information society" era.

The architecture is based on distributed computing, object orientation, and other concepts and standards from the telecommunications and computing industries, e.g., Open Distributed Processing (ODP), Intelligent Networks (IN), Telecommunication Management Networks (TMN), Asynchronous Transfer Mode (ATM), and Common Object Request Broker Architecture (CORBA).

The intention of this paper is to give an introduction to, and an overview of the TINA architecture. First, it presents an overall view of the TINA architecture, followed by a brief introduction to the main parts of the architecture, namely the computing, network, service, and management architectures of TINA. This is followed by a brief overview of interworking and migration scenarios for legacy systems. Finally, an overview of a tool-set to aid service specification and design is presented.

2. THE TINA-C VISION

The vision of the TINA efforts is, in short, to define and validate a consistent architecture for "open" telecommunications. TINA will address the needs of traditional voice-based services, future interactive multi-media services, information services, and operations and management type services, and will provide the flexibility to operate services over a wide variety of techno-

D. Gaïti (ed.), *Intelligent Networks and Intelligence in Networks*

logies [6][7]. This vision implies a **software architecture** that offers reusable software components, supports network-wide software interoperability, eases service construction, testing, deployment and operation, and hides from the service designer the heterogeneity of the underlying technologies and the complexity introduced by distribution.

The intention is to make use of recent advances in distributed computing (e.g., Open Distributed Processing (ODP) and Distributed Communication Environment (DCE)), and in object-oriented analysis and design, to drastically improve interoperability, re-use of software and specifications, and flexible placement of software on computing platforms/nodes. In addition, the consistent application of software principles to both services and management software is a primary goal. The TINA Architecture is furthermore defining how IN and TMN are "integrated" (related) within a common distributed framework. This will allow for a multi-supplier environment for telecommunications services and management systems.

Today, after three years of work with defining the TINA Architecture, the TINA-C Core Team is approaching a software architecture that seems to serve its initial purpose; a more suitable technique to handle distribution is taken into account, based on the standard to be RM-ODP [11]; end-user services and management services interwork smoothly and quite seamlessly; most are independent of the underlying technologies (computers, networks); while the most valuable concepts from the existing software architectures have been kept.

From this architecture, the steps to follow are to validate this architecture, to work out interworking scenarios with networks to be deployed and related interfaces (e.g., Intelligent Networks, Telecommunication Management Networks and ATM networks), to further develop related detailed specifications, and to further relate and to present the architecture to bodies and organizations working on similar issues (e.g., Network Management Forum, ATM Forum, Eurescom, RACE/ACTS and Object Management Group).

3. THE TINA OVERALL ARCHITECTURE

The technical goal of the TINA Architecture is to provide a set of concepts and principles to be applied in the design, processing, and operation of telecommunications software. The prevalent principle of TINA is that telecommunications services and management systems are considered software-based applications that operate on a distributed computing platform. The platform hides from applications the details of underlying technologies and distribution concerns, thus supporting the construction of portable and interoperable code.

3.1 Architecture introduction

The following outline three of the key foundations to the TINA architecture.

A TINA Service Component Model

A TINA service, as with any other telecommunications or computer applications, is provided as a set of interacting software components. These ***service components*** are structured according to a model called **USCM** (Universal Service Component Model) (Figure 1). The USCM states that each component should be composed of a **core** describing the nature of the object, regardless of how it is used or managed, a **usage** part describing its appearance to users and user access con-

siderations, a **management** part providing operations, maintenance, and administration for the component, and a **substance** part representing the component's dependence upon other components [14].

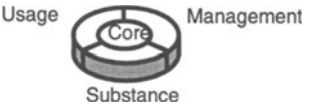

Figure 1. A service component model

Each service component can be divided into one or a number of software units, called ***computational objects***, as further described in Section 4 below.

A TINA Distributed Processing Environment

The communication (interaction) between the service components, or actually their computational objects, is supported in TINA by distributed computing mechanisms, e.g., Remote Procedure Calls (RPC). These mechanisms are provided by the core of a ***Distributed Processing Environment (DPE)***, as shown in Figure 2. The DPE provides a software sub-layer that operates above a native computing and communications environment (NCCE) and thus shields application programmers from technology dependant features.

In order to further enhance the components' capabilities to easily interact, generic support functions called *DPE Services* are defined. They are provided as additional services to a basic DPE. Examples of such generic servers are Traders and Name Servers, in order to help identify and locate relevant software units, Notification Servers, to help forward messages to the relevant software units, and Security Servers, to ensure only authorized interaction between service components takes place.

A TINA Component Categorization

In order to achieve good structure, modularity, and software re-usability, the service components are divided into three ***component categories - service components***, ***resource components*** and ***elements***, as shown in Figure 2. Components in the service category address the core functionality of services (service logic), and also covers access and service management considerations. Services can make use of common resources by interacting with components in the resource category. The resource components described (high-level) abstractions of the available resources, in a technology independent way, and provide facilities for the management of resources. Elements are software representations of individual resources such as transmission equipment, switch fabrics and computers.

3.2 Architecture subdivision

The TINA-C Overall Architecture is subdivided into four technical areas: the Computing Architecture, the Service Architecture, the Management Architecture, and the Network Architecture as shown in Figure 3.

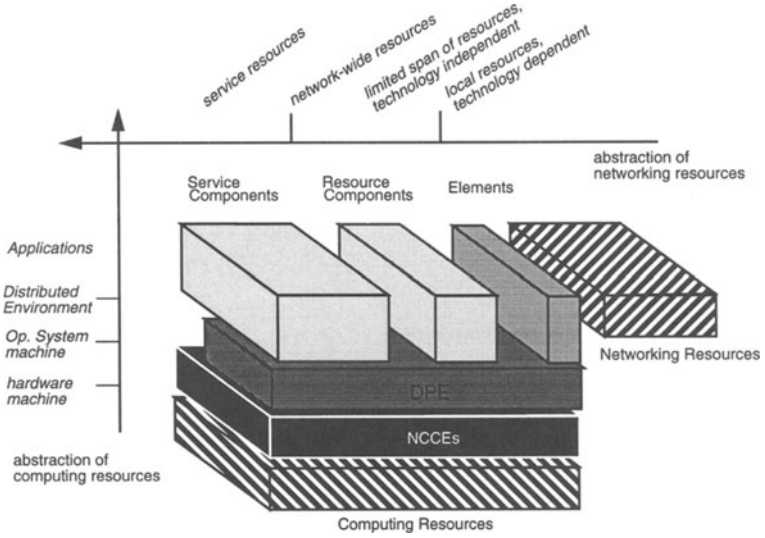

Figure 2. Overall Architecture

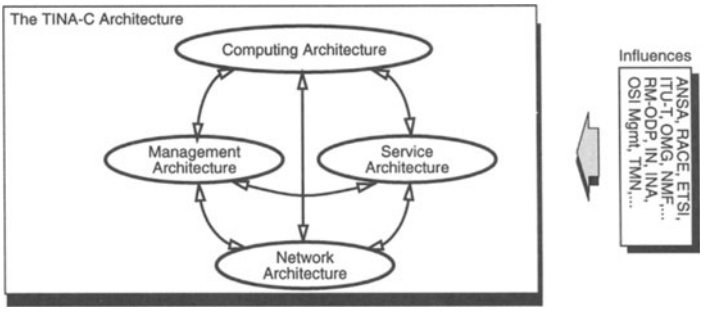

Figure 3. Partitioning of the TINA-C Software Architecture

- The *Computing Architecture* aims to provide a set of concepts and rules for how interacting software components (computational objects) are specified and how they communicate. The Distributed Processing Environment (DPE) defines the computer platform support for distributed execution of such components.
- The *Service Architecture* aims to provide a set of concepts and principles for specifying, analyzing, reusing, designing, and operating service-related telecommunications software components.

- The *Network Architecture* aims to provide a set of generic components for network resources. These components provide high-level abstractions of resources (e.g., connections) used by several service applications, and provides for the management of these resources.
- The *Management Architecture* aims to provide a set of generic management principles and concepts that are applied to the management of Service, Network, and Computing Architectures.

Components in the service, network and management architectures are structured according to the principles defined in the computing architecture, thus consistently applying the same software techniques.

Each architecture area is further discussed in the following sections.

4. THE TINA COMPUTING ARCHITECTURE

The computing architecture define the modelling concepts that should be used to specify object-oriented software in TINA systems. It also defines a distributed processing environment (DPE) that provides a support system that allows objects to locate and interact with each other. These concepts are based on the Reference Model for Open Distributed Processing (RM-ODP) [11][12].

The TINA computational modelling concepts defines 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, either of the same or a different type. There are two forms of interface that an object may offer or use: **operational interface** and **stream interface**. An operational interface is one that has defined operations, that allow for functions of the offering (server) object to be invoked by other (client) objects. An operation may have arguments and may return results. A stream interface is one without operations (i.e., there is no notion of input/output parameters, requests, results, or notifications). The establishment of a **stream** between stream interfaces allows for the passing of other structured information, such as video or voice bit streams. Streams are established by interacting with service and network components, see below. Figure 4 depicts these concepts.

The interfaces are specified independently of any programming language by means of a specification notation. As none of the currently existing and relevant notations cover all the features of the TINA computational model, extensions to the Object Management Group's Interface Definition Language (OMG IDL) have been made. This notation is called TINA Object Definition Language (ODL)

Although the computational concepts are appropriate for describing how an application is structured into separate, logically distributed components, it is not appropriate for describing what this application is actually doing. This is the purpose of the **information modelling concepts**, from which an application designer can identify the information-bearing entities in the problem domain under study, show their relationships, and state the constraints directing their behavior. In TINA, information specifications are made using a quasi GDMO-GRM notation [13]. Stan-

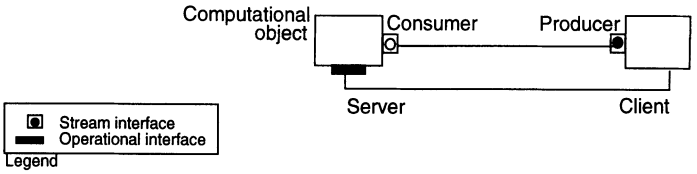

Figure 4. Computational modelling concepts

dard GDMO-GRM allows for many concerns to be expressed in a specification, including information and computational issues. TINA quasi GDMO-GRM uses only aspects of the language suitable for TINA information modelling.

A computational object can interact with another computational object without knowing on what computer the other object resides. There is an assumption that all computational objects reside on a platform, called a distributed processing environment (DPE). The DPE provides location and interaction mechanisms allowing computational objects to locate one another at execution time. **Engineering modelling concepts** define this DPE infrastructure.

From an application designer's standpoint, the DPE can be considered as one homogeneous infrastructure hiding the complexity and heterogeneity of the underlying network and computing resources. However, in reality a DPE consists of "kernels" implemented on top of different heterogeneous computing environments. The DPE kernels are interconnected by a specific logical network, called the **Kernel Transport Network** (KTN) [8]. The KTN facilitates the exchange of messages between computational objects. The DPE kernels are enhanced by additional generic software components, called DPE services, dealing with distribution, security, quality of service, operability, etc. An example is the trader which is a service that allows a computational object to locate interfaces of other computational objects.

A more detailed description of the DPE can be found in [2].

5. THE TINA NETWORK ARCHITECTURE

The purpose of the network architecture is to provide a set of generic concepts that describe transport networks in a technology independent way. The network architecture defines a set of abstractions that the resources layer can work with. At one end it provides a high level view of network connections to services. At the other end it provides a generic descriptions of elements, which can be specialised to particular technologies and products.

The network architecture has been defined by taking into account the layering principles of ITU-T Recommendation G.803 [9]. These principles separate the functions of a transport network into layers according to the characteristic information handled by the function. The **Network Resource Information Model** (NRIM) is an information specification of transmission and switch technologies, in which the technology dependent aspects have been extracted (e.g. hide differences between ATM and SDH switches). The model concerns how individual elements are

related, topologically interconnected, and configured to provide and maintain end-to-end connectivity. The model therefore defines technology independent concepts that can be used to derive technology independent control and management functions. When designing and implementing a real network the technology dependant aspects must be taken into account as specializations of the generic model. The concepts found in this model include Layer Network, Sub-Network Connection (SNC), connection, topological link, and network termination points.). Figure 4 shows an example network.

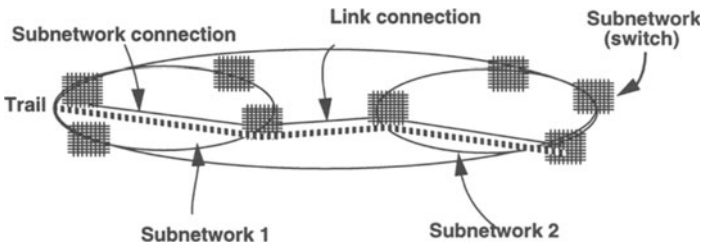

Figure 5. Example of abstract resource components

The network resource information model can be used to describe in detail a real network. However, it contains details which a user may not want to be, or should not be, aware of. Any use of a network, aside from management activity, must be as a result of using a service. It is important therefore to provide a service oriented view of connectivity. The concept of **connection graph** is used for this purpose. A connection graph depicts vertices with ports, where ports are connected by lines to represent connectivity.

There are two basic types of connection graph. In a **physical connection graph** the vertices represent physical nodes, the ports network access points and the lines connections. In a **logical connection graph** the vertices represent objects, the ports stream interfaces and the lines streams. The purpose of these two graphs is to provide two different views of a set of connections. Service software will request the establishment of streams between computational objects, and will build a logical connection graph to represent this. To build a stream will require the establishment of network connections and internal node bindings between the network access points and the stream interfaces. Network connections are expressed as physical connection graphs. Note that in each type of graph only the end-points of connections are represented. Intermediate nodes are abstracted away from since this information is not required by services.

A service can express connectivity requirements by building a graph of end-points (vertices and ports) and lines, and request a **communication session manager** to build an appropriate physical connection graph and intra-nodal bindings. The communication session manager accepts logical connection graphs as input from a service and produces a physical connection graph as output. It is responsible for negotiating with **connection management** to establish the physical connection graph (i.e. the connections), and to interact with nodes to perform internal network

access point/stream interface bindings. The communication session manager is also responsible for monitoring and communicating any changes in each graph that could affect the other. The connection management components are derived from the NRIM definitions.

6. THE TINA SERVICE ARCHITECTURE

The service architecture provides means to build services and a service support environment [14]. The term "service" in TINA is used in a wide sense. It includes telecommunication services, management services and end-user services. Thus the service architecture is applicable to a wide range of service types, including management services, information services, transport services, and access services.

From the information viewpoint, the TINA service architecture provides a framework for describing what services do (their prime function), how they are managed, and how terminal and user mobility is achieved. Two main separation principles are defined in the service architecture; the separation of call and connection control, and the separation of user and terminal related issues. The separation of calls from connections, allows a more flexible approach to the handling of services, whereby the negotiation and involvement of users and their activities can be separated from the negotiation and involvement of communication resources. The separating of terminal and user issues allows for a flexible mobility model to be defined.

From the computational viewpoint, the TINA service architecture describes how a distributed service should be structured in order to be provide its function to a user. The service architecture identifies the run-time software components that should serve as a foundation for all services and therefore constitute a library of reusable components. The most important of these components are **user agents**, **terminal agents**, **service session**, **subscription manager**, and the **communication session manager**.

A user agent represents the user in the network. It handles, on a users behalf, incoming and outgoing service establishment, and maintains profiles of the user and any customized details associated with services. A terminal agent represents a terminal attached to a network. It provides a technology independent view of the terminal to the network, and manages terminal specific information. The power of the user and the terminal agent concept is that users and terminals can be located by finding the related agent. This avoids having to build in location knowledge and thus allows users and terminals to move around in the network.

A service session is an instance of an executing service. It maintains the state of a service, and provides interfaces for the joining and leaving of users to a service session. Embodied within a service session is the service core, or service logic. In response to commands from users, or service logic, connections may need to be established, modified, or 'dropped.' Service sessions do this by constructing logical connection graphs and instructing a communication session manager to build the connections (see Section 5).

The subscription manager maintains information related to customer and user subscriptions. It is use during the establishment of a users involvement in a service session to check access permission with respect to a subscription.

Figure 6 highlights these components operating above the DPE. More detail on these components can be found in [1][3].

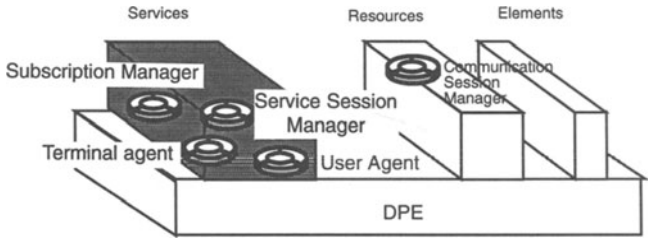

Figure 6. The Service Architecture reusable components

7. THE TINA MANAGEMENT ARCHITECTURE

The TINA management architecture [15] defines a set of principles for the management of the entities in a TINA system. There are two basic types of management in TINA systems: **computing management**, and **telecommunications management** (Figure 7).

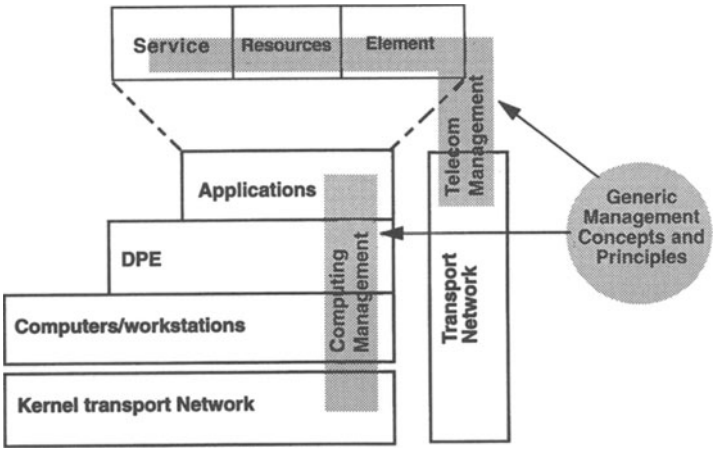

Figure 7. Two basic types of management.

Computing management involves the management of the computers, platform, and of the software (in general terms) that runs on that platform. Management here does not concern itself with what applications are doing nor on application specific management. The main concern is the deployment, installation, and load balancing of software.

In TINA the application software relates to telecommunications services, and software for the control and management of transmission and switching networks. These telecommunications specific software components require management. This is called telecommunications management.

The above two types of management are very broad, and therefore are divided into sub-types of management. Telecommunications management is broken down into service, network, and element management, in much the same way as TMN systems. Computing management can be broken down into generic software management, such as deployment, configuration, and instantiation of software, and management of the DPE and computer environments. The service, network and computing architectures therefore each contain management concepts and principles.

Even though different types of management have been identified, the TINA architecture defines a set of generic management concepts and principles.There are two basic sets of principles for generic management. The first deals with functional separations. This breaks down the problem of management into distinct areas of concern. The second deals with modelling of management systems. This provides principles for how to express management relationships and operations in terms of the TINA object models (information, computation and engineering). Five functional separations have been taken from OSI Systems Management, namely fault, configuration, accounting, performance, and security

Further details of management in TINA can be found in [1].

8. INTERWORKING SCENARIOS AND MIGRATION PATHS TOWARDS TINA

The TINA Architecture is being defined to suit the needs of a future marketplace of information services. However, the success of the architecture will largely depend on its ability to interwork with existing systems and on its applicability to migrate existing systems towards being consistent with the TINA Architecture. Thus work has begun on interworking and on migration issues.

Obviously, a number of different technologies are being deployed or are about to be deployed in today's networks - examples include IN, ISDN, ATM, TMN and OSI. It is not within the scope of the Core Team to examine all issues related to all these technologies. Instead a few characteristic examples will be chosen to be studied in **reference scenarios.**

A reference scenario shows, for a given technology and network environment, typical interworking and mapping possibilities. Interworking shows how a TINA system interacts with a legacy system in order to form a cooperation, or to reuse the existing technology. Mapping on the other hand shows how some of the TINA concepts can be used to evolve an existing system or architecture. Migration describes the development and realization of a series of reference scenarios.

It should be made clear that the Core Team is not working on migration scenarios per se, or on defining specific network solutions to be adopted and deployed by network operators. The work is instead intended as relevant "real-world" examples of how the TINA architecture might be

used in different network environments. The results will focus TINA to better understand the **requirements** on the architecture which have to be satisfied before operational use of TINA can be achieved.

9. TINA METHODOLOGY AND TOOLS

Even though the intent and scope of the core team are not to build a service creation environment, some tools are currently used and customized by the core team. One is an Object Modelling Technique (OMT) tool that is used and customized in TINA-C to model information objects. From this modelling, quasi-GDMO-GRM specifications can be derived automatically as a text representation of the information model. We envisage expanding the OMT tool (into a skeleton of SCE) to help the designer to produce the TINA-C ODL specifications of computational objects. The other tool is the ODL specification pre-processor that produces IDL specifications suitable for compiling onto Common Object Request Broker Architecture (CORBA) platforms. Figure 5 depicts this tool chain.

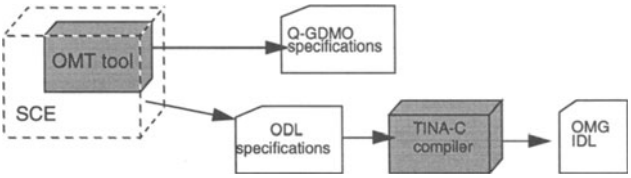

Figure 8. TINA-C Design Support Tools

10. CONCLUSION

Many telecommunications stakeholders have already shown their interest in the TINA architecture and expect it to be usable in 1996. Therefore, an important part of the TINA-C activities is devoted, prior to any standardization effort, to validating the current specifications through auxiliary projects, collaborative worldwide demonstrations, and implementations carried out by the TINA-C Core Team. It is also expected that typical migration scenarios from currently deployed intelligent networks and telecommunications management networks to TINA networks will be available soon. This will make it possible to proceed gradually and cost effectively towards a Telecommunications Information Networking Architecture.

11. REFERENCES

[1] H. Berndt, L. de la Fuente, P. Graubmann, *"Service and Management Architecture in TINA-C"*, TINA 95 Conference Proceedings.

[2] E. Kelly, N. Mercouroff, *"TINA-C DPE Architecture and Tools"*, TINA 95 Conference Proceedings

[3] H. Yagi, K. Ohtsu, M. Wakano, H. Kobayashi, R. Minerva, *"TINA-C Service Components"*, TINA 95 Conference Proceedings.

[4] J. Bloem, J. Pavon, M. Schenk, H. Oshigiri, *"TINA-C Connection Management Components"*, TINA 95 Conference Proceedings.

[5] V. Cohen, G. Chapman, N. Dathi, F. Ruano, *"TINA Validation through Prototyping"*, TINA 95 Conference Proceedings.

[6] William J. Barr, Trevor Boyd, Yuji Inoue, *"The TINA Initiative"*, IEEE Communications Magazine, March 1993.

[7] N. Natarajan, G.M. Slawsky, *"A Framework Architecture for Multimedia Information Networks"*, IEEE Communications Magazine, February 1992.

[8] G. Brégant, R. Kung, Y. Lepetit, J-B. Stefani, *"The evolution of networks: towards more integrated architectures"*, XIV ISS, October 1992.

[9] CCITT Recommendation G.803, *Architectures of Transport Networks Based on the Synchronous Digital Hierarchy (SDH),* June 1992.

[10] S. Rudkin, *"Templates, types and classes in open distributed processing"*, BT Technology Journal, July 1993.

[11] ISO 10746-2 RM-ODP Part 2, *"Descriptive model"*, Yokohama, June 1993.

[12] ISO 10746-3 RM-ODP Part 3, *"Prescriptive model"*, Yokohama, June 1993.

[13] ISO 10165-7, *"Information Technology - Open Systems Interconnection - Structure of Management Information - Part 7: General Relationship Model"*, January 1993.

[14] H. Berndt, P. Graubmann, *"Service Architecture, Service Session and Service Federation in TINA-C"*, submitted to ISS'95.

[15] L. A. de la Fuente, J. Pavon, J. C. Moreno, *"The TINA-C Approach to TMN"*, submitted to ISS'95.

[16] F. Dupuy, R. Minerva, J. Bloem, H. Hammainen, J. Moreno, *"The TINA-C Architecture as related to IN and TMN"*, 3rd International Conference on Intelligence in Networks, Bordeaux, October 1994.