UMA PLATAFORMA DE COMPUTAÇÃO PARA ADMINISTRAÇÃO DE IBCNs

A. Mauro B. OLIVEIRA
mbo@masi.ibp.fr

Jose Neuman DE SOUSA
jn@masi.ibp.fr

Manoel Camilo PENNA
map@masi.ibp.fr

Joaquim CELESTINO Junior
jc@masi.ibp.fr

Laboratoire MASI
Universite P. et M. Curie-CNRS
39, avenue des Etats-Unis
78000 - Versailles (FRANCE)

Sumário

Este trabalho tem como objetivo descrever a arquitetura de uma plataforma de computação para a Administração de IBCNs (*Integrated Broadband Comunications Networks*). Um protótipo para essa arquitetura é apresentado, com ênfase nos Modelos de Computação e de Engenharia do RM ODP (*Reference Model for Open Distribution Processing*). Tal arquitetura está sendo desenvolvida por vários grupos de pesquisa da Comunidade Européia dentro do Projeto ADVANCE, o qual integra o Programa Europeu RACE.

Abstract

The aim of this paper is to describe a computer platform architecture for IBCN (Integrated Broadband Comunications Network) administration and a prototype implemented for this architecture. The Computation and Engineering ODP (Open Distribution Processing) aspects of this prototype are focused. This architecture has been developed by several partners of the Europeen Community on the ADVANCE Project of the RACE Program.

1. INTRODUÇÃO

Integrated Broadband Comunications Networks (IBCNs) seguem a tendência iniciada pelas ISDNs (*Integrated Service Digital Networks*) [1]. São sistemas que fornecerão aos usuários das empresas de telecomunicações na Europa, em um futuro próximo, novas facilidades e serviços integrados de dados, voz e vídeo. Esses serviços são hoje disponíveis de uma forma não satisfatoriamente integrada, devido a vários fatores, dentre os quais, as diferentes tecnologias necessárias para torná-los acessíveis aos usuários.

Os autores são financiados pelo CNPq, CAPES, ETFCe e Banco do Nordeste do Brasil

IBCNs se apresentam, então, como uma solução à difícil tarefa de integração de modernos serviços de telecomunicações. Esse ambiente complexo necessita ser administrado de maneira eficiente, para que novas aplicações e serviços possam ser disponíveis e a um custo razoável. A complexidade dessa administração torna-se ainda maior quando consideramos que as IBCNs serão constituídas de uma enorme quantidade de componentes de diferentes tecnologias, provenientes de diferentes fabricantes.

O projeto ADVANCE é um entre quatro projetos do programa Europeu RACE [2]. Seu objetivo maior é a investigação de avançadas tecnologias para TMN (*). Nesse contexto, ADVANCE se preocupa com problemas de administração relacionados aos usuários e à infraestrutura de telecomunicações em IBCNs, denominada NCAS (*Network and Customer Administration System*) [3]. Uma arquitetura lógica para NCAS e um protótipo foram desenvolvidos pelos participantes do Projeto ADVANCE, sendo um desses participantes a Equipe *Reseaux et Performance* do Laboratório MASI (Paris 6).

Neste trabalho descrevemos nossa participação no Projeto ADVANCE. Essa participação se resume no desenvolvimento de uma Plataforma de Computação para dar suporte à arquitetura do NCAS. Iniciamos este documento descrevendo alguns conceitos básicos pertinentes; em seguida apresentamos a arquitetura lógica (blocos funcionais) do NCAS e a Plataforma de Computação desenvolvida, onde focalisamos o problema de localização de objetos. Finalizamos o documento apresentando o cenário de um protótipo, no contexto do RM ODP (*Reference Model for Open Distribution Processing*) [4] e a descrição funcional das interfaces que permitem a conexão de uma aplicação de administração à plataforma de computação.

2.ELEMENTOS BÁSICOS DA PLATAFORMA DE COMPUTAÇÃO

Nesta seção nós apresentamos conceitos e ferramentas fortemente relacionados à Plataforma de Computação do NCAS, implementada no Laboratório MASI. São eles o RM ODP (*Reference Model for Open Distribution Processing*) [4], o *Trader* [5] e a linguagem GMS [6]. Optamos por uma apresentação comparativa desses elementos com padrões existentes, pelo fato de eles serem ainda incipientes (RM ODP), ou não bastante difundidos (ANSA Trader) ou particulares (GMS) a um ambiente específico.

2.1 OSI & ODP

Os sistemas distribuídos devem oferecer às aplicações, serviços que estão disponíveis em sistemas centralizados. Um desses serviços, por exemplo, é a possibilidade de uma aplicação comunicar-se (*openness*) com outros sistemas.

A idéia de sistemas abertos (*open systems*) não é nova e tem evoluído na última década. Os protocolos OSI da ISO têm sido intencionalmente desenvolvidos para interconexão de sistemas abertos. Embora seja cogitado o uso de Protocolos OSI em projetos de sistemas distribuídos, os trabalhos no contexto OSI não fornecem, a priori, padrões para processamento distribuído.

Um esforço de normalização para produzir padrões para ODP (*Open Distributed Processing*) é agora efetivo no comitê ISO/IEC JTC1 SC21 WG7. Nesse sentido, tem sido observado que são vários os pontos de vista através dos quais a distribuição pode ser considerada. Esses pontos de vista ou projeções, são organizados em 5 modelos:

(*) TMN (Telecommunication Management Network) diz respeito a sistemas baseados em computador que dão suporte à todas as operações de gerenciamento de redes em ambientes de telecomunicações. TMN é definido na Recomendação M30 (blue book) do CCITT.

- **Empresa**
- **Informação**
- **Computação**
- **Engenharia**
- **Tecnologia**

Assim, um sistema é representado por um modelo, segundo uma determinada projeção. Isso permite a descrição de um sistema distribuído com o resultado revelando suas diferentes facetas. Cada descrição é completa e auto-suficiente para especificar o sistema, não importando quanto do sistema foi descrito mas que aspectos do sistema foram enfatizados. Uma descrição segundo um modelo ou projeção representa um conjunto diferente de abstrações do sistema original. A seguir apresentamos uma descrição de cada uma das projeções do RM ODP:

- **Empresa**: descreve os objetivos gerais do sistema em termos de funções e atividades que existem dentro da organização utilizando o sistema; as interações entre o sistema e o ambiente no qual está inserido; a estrutura organizacional da empresa; quais informações seram acessíveis aos diferentes usuários, etc.

- **Informação:** especifica as estruturas dos elementos de informação de um sistema, as regras estabelecendo as relações entre esses elementos e as restrições impostas a ambos, regras e elementos. Esse modelo deve também mostrar como a informação é distribuída através do sistema e não se preocupa em especificar quais partes do sistema serão executadas automática ou manualmente;

- **Computação:** fornece as estruturas de programação e as ferramentas para o desenvolvimento de programas que estarão disponíveis aos programadores de aplicações distribuídas.

  Alguns dos aspectos enfatizados no Modelo de Computação do ODP são:

  - modularidade da aplicação distribuída;
  - transparência de acesso na invocação de interfaces;
  - mecanismo de passagem de parâmetros;
  - configuração e transparência de localização de interfaces;
  - restrições de concorrência e sincronização nas interfaces;
  - extensão de linguagens existentes para dar suporte programação distribuída .

- **Engenharia**: fornece ao programador de sistemas os mecanismos de tradução (compilação, ligação e edição, interpretação) e um núcleo de funções de suporte de base, necessários à realização da computação em ambientes distribuídos e heterogêneos.

  Alguns dos aspectos tratados no Modelo de Engenharia do ODP são:

  - gerenciamento de processos (*task* e *thread*);
  - gerenciamento de espaço de endereçamento;
  - protocolos de aplicação distribuída;
  - protocolos de redes;
  - localização de interface ;
  - interface *traders* (mecanismo de identificação de interfaces);
  - gerenciadores de configuração e de operações atômicas

- **Tecnologia**: especifica as ferramentas operacionais nas quais o sistema distribuído é construído. A descrição pode incluir padrões OSI ou tecnologias proprietárias. Esse modelo mostra como hardware e software são mapeados nos mecanismos identificados no modelo de engenharia, incluindo sistemas operacionais locais, dispositivos de entrada/saída, memória e pontos de acesso para comunicação.

A figura abaixo ilustra as 5 projeções do RM ODP:

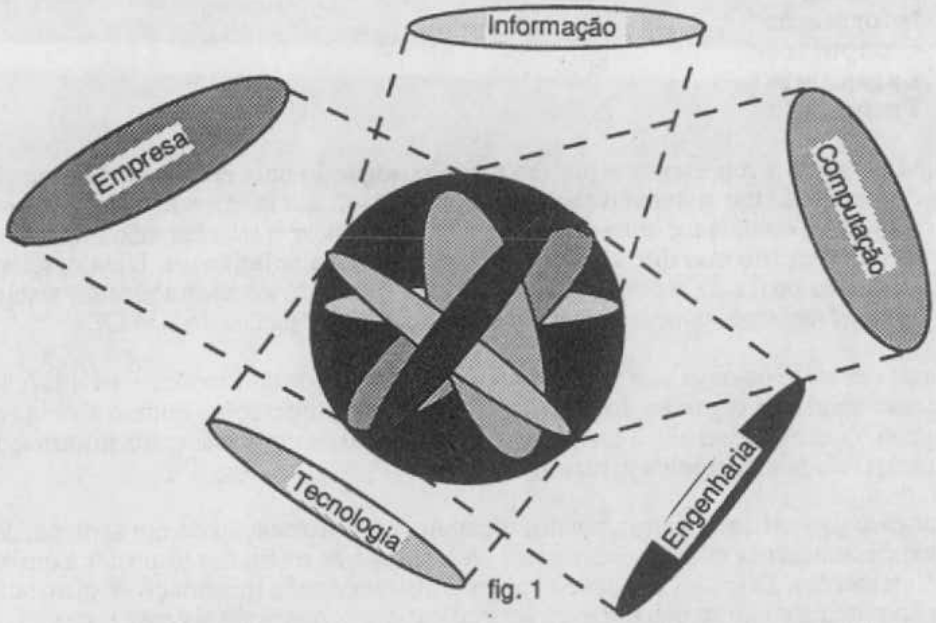


fig. 1

2.2 Diretório X.500 & Trader

Para que qualquer sistema de comunicação funcione satisfatoriamente, é necessário um esquema de nomes e endereços (*naming & addressing*), ou algum tipo de interface para um sistema de diretório. A idéia é permitir ao usuário o acesso a um objeto de maneira mais natural possível, sem que o mesmo necessite conhecer o "exato" endereço desse objeto.

Como definido em [7], o Diretório X.500 é uma coleção de sistemas abertos que cooperam entre si para manter uma base de dados de informação DIB (*Directory Information Base*) sobre um conjunto de objetos do mundo real. Exemplos de tais objetos inclui pessoas (nome, endereço, telefone,...); organizações (..., telex, correio eletronico,...); facilidades às aplicações OSI, aos processos de administração OSI, às outras entidades das camadas OSI e aos serviços de telecomunicações.

O modelo de referência ODP emergente, prevê uma função denominada *Trading* que permite o casamento (*matching*) entre Serviços oferecidos e Clientes potenciais desses serviços. Essa função é realizada por um componente de infraestrutura denominado *Trader*.

O *Trader* pode ser visto como um diretório acrescido de uma facilidade de seleção de serviços, segundo critérios estabelecidos pelo Cliente. Os Serviços são organizados de acordo com uma estrutura de identificação e de um esquema de tipos e propriedades. Um Serviço pode ser então pesquisado por contexto de identificação, por propriedades, ou uma combinação de ambos. Essas propriedades podem incluir: o nome da entidade; a funcionalidade oferecida por essa entidade; a informação ou tipo de informação contida dentro da entidade; um outro critério usado pelos programadores de aplicação e de uso frequente na clasificação da natureza da entidade.

Um Servidor deve "exportar" uma referência à interface de seu serviço, que é então registrada em um contexto de identificação do *Trader*, tornando-se assim acessível para Clientes potenciais. Um Cliente deve, então, "importar" um Serviço junto ao *Trader* . A operação de "importação" retorna uma Referência de Interface não ambígua para o Cliente, permitindo ao mesmo aceder ao serviço "exportado".

2.3 CMIS & GMS

GMS (Generic Message System) é uma linguagem interativa, baseada em Objeto, a qual foi projetada para facilitar comunicações em sistemas de informação distribuídos e heterogêneos [6]. Ela é usada no protótipo (descrito no final) pelas aplicações para acessar e manipular informações compartilhadas no NCAS, bem como invocar funções remotas de gerenciamento.

GMS possui similaridades com o CMIS da OSI, fornecendo primitivas como GET (equivalente ao M-GET do CMIS), CREATE (M-CREATE), ACTION (M-ACTION), DELETE (M-DELETE) e MODIFY (M-SET) [8]. No entanto, GMS é uma linguagem conceitualmente concisa, o que reduz a complexidade na especificação de consultas (seu paradigma é a linguagem SQL) fornecendo mecanismos de abstração construídos dentro da própria linguagem.

A estrutura básica da primitiva GMS é:

{<operation> <identification> <qualification>}

Nesta estrutura, **operation** especifica o tipo de solicitação (GET, DELETE, etc.), enquanto **identification** e **qualification** especificam, respectivamente, a classe de objetos a ser ativada e eventuais limitações a serem aplicadas à classe, permitindo uma solicitação mais acurada na seleção de um objeto específico.

3. ARQUITETURA LÓGICA DO NCAS

A arquitetura do NCAS consiste de componentes que interagem no tratamento de diferentes aspectos de solicitações funcionais e não-funcionais do sistema [3]. Esses componentes do NCAS constituem o Modelo Computational do protótipo. São eles:

- Aplicações de Gerenciamento (Management Application: MA)
- Modelo de Informação Comum (CIM)
- Serviços Comuns para dar suporte ao Modelo de Informação
- Serviços Genéricos para dar suporte às MAs
- Interfaces e Plataforma de (Distribuição) Computação

3.1 Aplicações de Gerenciamento (MA)

As Aplicações de Gerenciamento representam os elementos da arquitetura lógica diretamente acessíveis aos usuários finais e a outras MAs, na realização da atividade de gerenciamento de rede.

3.2 Modelo de Informação Comum (CIM)

O CIM é um importante componente da arquitetura do NCAS. Ele é responsável pelo suporte às MAs, na medida em que ele representa uma abstração dos objetos gerenciados e gerenciadores da arquitetura, fornecendo às MAs uma representação conceitual dos recursos do sistema, através de um modelo orientado a objeto.

Portanto, o CIM contém um modelo conceitual (descrição dos conceitos de gerenciamento compartilhado de um TMN) e mecanismos para acessar e/ou alterar esse modelo.

O modelo conceitual dentro do CIM é composto de três sub-modelos [9]:

- Modelo de Objeto Gerenciado (**MOM**) - é a representação do sistema gerenciado (redes, serviços, clientes etc) e da informação de gerenciamento (performance da rede, configuração, planos de serviço, etc.) sobre o sistema, em forma de objetos;

- Modelo do Sistema de Gerenciamento (**MSM**) - é a representação da funcionalidade de gerenciamento do NCAS e de outros sistemas de gerenciamento no TMN. Esse modelo fornece a uma aplicação as diferentes funcionalidades do TMN que podem ser invocadas;

- Modelo de Outros TMNs (**MOT**) - é a representação de outros TMNs (do sistema gerenciado, de informações de gerenciamento e de funcionalidades).

3.3 Serviços Comuns que dão suporte ao CIM

Os serviços comuns oferecem suporte às solicitações do CIM para acesso transparente aos recursos compartilhados do sistema, tais como dados, base de conhecimento, recursos de interface homem-máquina e de rede.

3.4 Serviços Genéricos que dão suporte às MAs

Os serviços genéricos são definidos como os componentes da arquitetura lógica NCAS que dão suporte às solicitações das MAs para acesso transparente aos dados, base de conhecimento etc, para uso privado.

3.5 Interfaces

Foram identificados na arquitetura NCAS três pontos onde há necessidade de interfaces bem definidas. São eles:

- Entre as MAs e o CIM: Esta interface fornece às MAs os meios para acessar e/ou alterar informações e funcionalidades dentro dos sub-modelos do CIM;

- Entre o CIM e os Serviços Comuns: Esta interface fornece ao CIM acesso consistente aos recursos compartilhados;

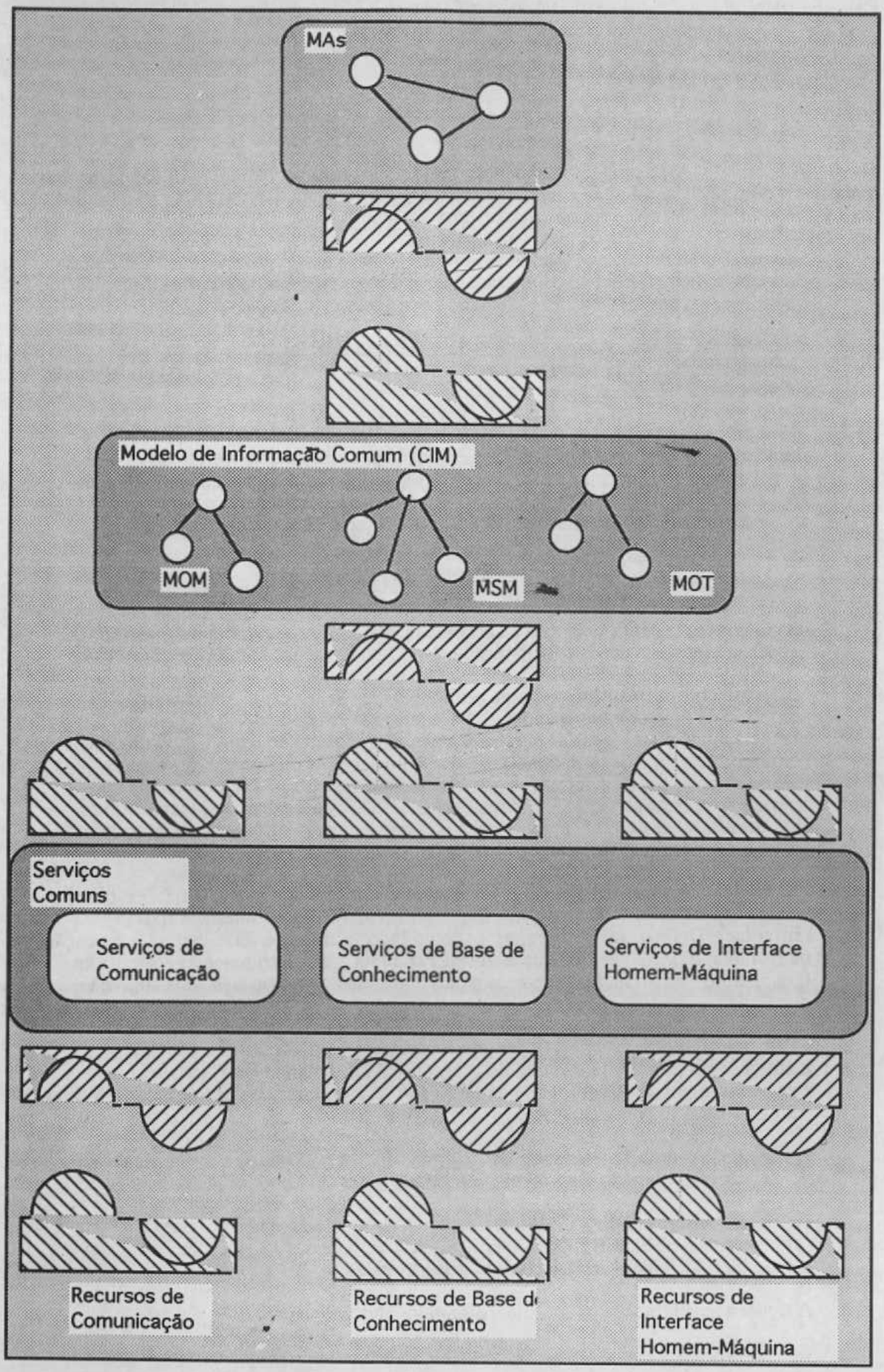
- Entre as MAs e os Serviços Genéricos: Esta interface fornece às MAs mecanismos consistentes de interação com os recursos do sistema

3.6 Plataforma de (Distribuição) Computação

A plataforma de distribuição é o componente da infraestrutura que permite que o NCAS seja executado em unidades de processamento heterogêneas (hosts).

A plataforma oferece a todos os componentes do sistema um apropriado nível de abstração de distribuição.

A figura, a seguir, ilustra a Arquitetura lógica do NCAS:


MAs
Modelo de Informação Comum (CIM)
MOM
MSM
MOT
Serviços Comuns
Serviços de Comunicação
Serviços de Base de Conhecimento
Serviços de Interface Homem-Máquina
Recursos de Comunicação
Recursos de Base de Conhecimento
Recursos de Interface Homem-Máquina

4. PLATAFORMA DE COMPUTAÇÃO

4.1 Sistema ANSAware

ANSAware [13] é uma implementação do Modelo de Engenharia ANSA [5], projetado para servir de suporte ao Modelo Computacional. Alguns conceitos importantes são descritos a seguir (ver também discussão sobre o *trader*, feita no item 2.2)

- **Serviço** é uma função que manipula informação, seja ela de processamento, memória ou transferência. Os objetos que usam um serviço são chamados **clientes** e aqueles que fornecem um serviço são chamados **servidores**.

- **Referências de interface.** Do ponto de vista computacional, os objetos se comunicam através da passagem de **referências de interface**, que são entidades que referenciam instâncias de uma interface.

- **Nó & núcleo:** O termo **nó** é usado para referenciar, por ex., uma estação de trabalho. Uma rede de computadores gerenciada por um sistema operacional distribuído pode também constituir um nó. Os recursos de um nó são gerenciados por um objeto de engenharia chamado **núcleo**. O serviço oferecido pelo núcleo é de construir um ambiente de computação distribuída básico, uniforme e independente de sistemas operacionais, computadores e redes, a partir dos recursos de um nó.

- **Cápsulas ANSA.** Uma Cápsula ANSA é a unidade de operação autónoma dentro do ANSAware. Ela representa um espaço de endereçamento virtual e é mapeada dentro de um correspondente abstração no sistema operacional, por ex., um processo UNIX. A cada componente do sistema (MA, CIM, etc.) está associada uma Cápsula ANSA.

- **Objetos computacionais e de engenharia:** Um Serviço no ANSAware é composto de objetos computacionais, definidos pelo programador. Um objeto computacional pode ter várias interfaces, cada uma oferecendo o mesmo ou diferentes conjuntos de operações (i.e. o mesmo serviço ou serviços diferentes). Os objetos computacionais são transformados por compiladores em objetos de engenharia que interagirão com os serviços de transparência. ANSAware fornece dois compiladores para transformar a especificação de objetos computacionais em objetos de engenharia: **stubc** e **prepc** (descritos a seguir):

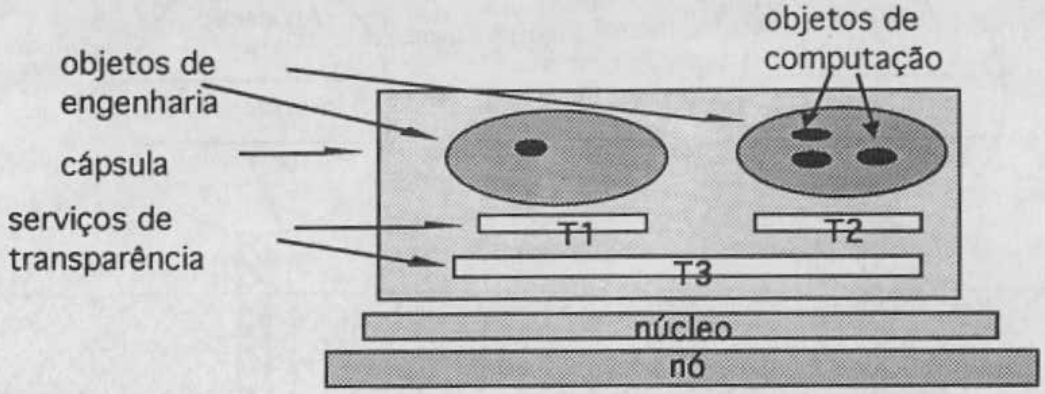
  - **Compilador stub (stubc)**: Os objetos computacionais tem suas interfaces de serviço especificadas utilizando-se a linguagem de definição de interface **IDL** [5] **(Interface Definition Language)**. O compilador **stubc** converte esta especificação num conjunto de rotinas *stub* que realizam a comunicação de argumentos e resultados (*marshalling/ unmarshalling*) entre as partes logicamente separadas de uma aplicação distribuída;

  - **Pre-processador (prepc)**: O **prepc** processa arquivos fontes que contêm instruções próprias da linguagem de programação distribuída **DPL (*Distributed Programming Language*)**, gerando código 'C' com referências apropriadas a rotinas *stub* geradas pelo compilador [5].

A seguir são mostradas exemplos de estruturas dpl, através das quais um Cliente acessa ("Importa") um serviço tornado disponível ("Exportado") por um Servidor [13]:

```
!     {ir} <-traderRef$Import ("GMSInterface", "/CIM", "'Customer' in Classes")
!     {er} <-traderRef$Export ("GMSInterface","/CIM", "Classes {'Customer'
            'Service' 'Network'} Relations {'is_provided_to'}", n).
```

A figura a seguir mostra os conceitos discutidos anteriormente:



4.2 Mecanismo de Entrega GMS_DM (Delivery Mechanismo)

A NCAS é distribuído devido a natureza distribuída das IBCNs. O objetivo da Plataforma de Computação é fazer com que uma mensagem GMS enviada por um componente do NCAS chegue ao componente destino na arquitetura, de tal forma que a distribuição desses componentes seja transparente. A Plataforma é, basicamente, o módulo do NCAS responsável por localizar objetos e gerenciar o fluxo de informações na arquitetura. Esse módulo, denominado Mecanismo de Entrega GMS_DM (*Delivery Mechanism*), define também uma estratégia de como os componentes do sistema poderão ser conhecidos (esquema de nomes) no NCAS. O GMS_DM é baseado no sistema ANSAware, como mostrado na figura a seguir,

Portanto, uma função essencial do GMS_DM é a localização do objeto destino da mensagem GMS, a partir de um *Target Name*. Isso é feito através o através do mecanismo de *trading* do ANSA. O modelo conceitual do GMS_DM é composto dos seguintes três blocos funcionais [10] (*):

- **Gerente de Localização** (LM - "Location Manager"): Responsável pela identificação do componente (local ou remoto) para o qual a mensagem deve ser enviada;
- **Gerente de Sessão** (SM - "Session Manager"): Responsável pelo controle das mensagens, gerenciando assim a comunicação ("time-out", perda de mensagem, etc..);
- **Gerente de Mensagem** (MM - "Message Manager"): Responsável pelo envio e recepção das mensagens do sistema, tanto a nível interno, quanto externo ao GMS_DM.

4.2 Localização de objetos na plataforma

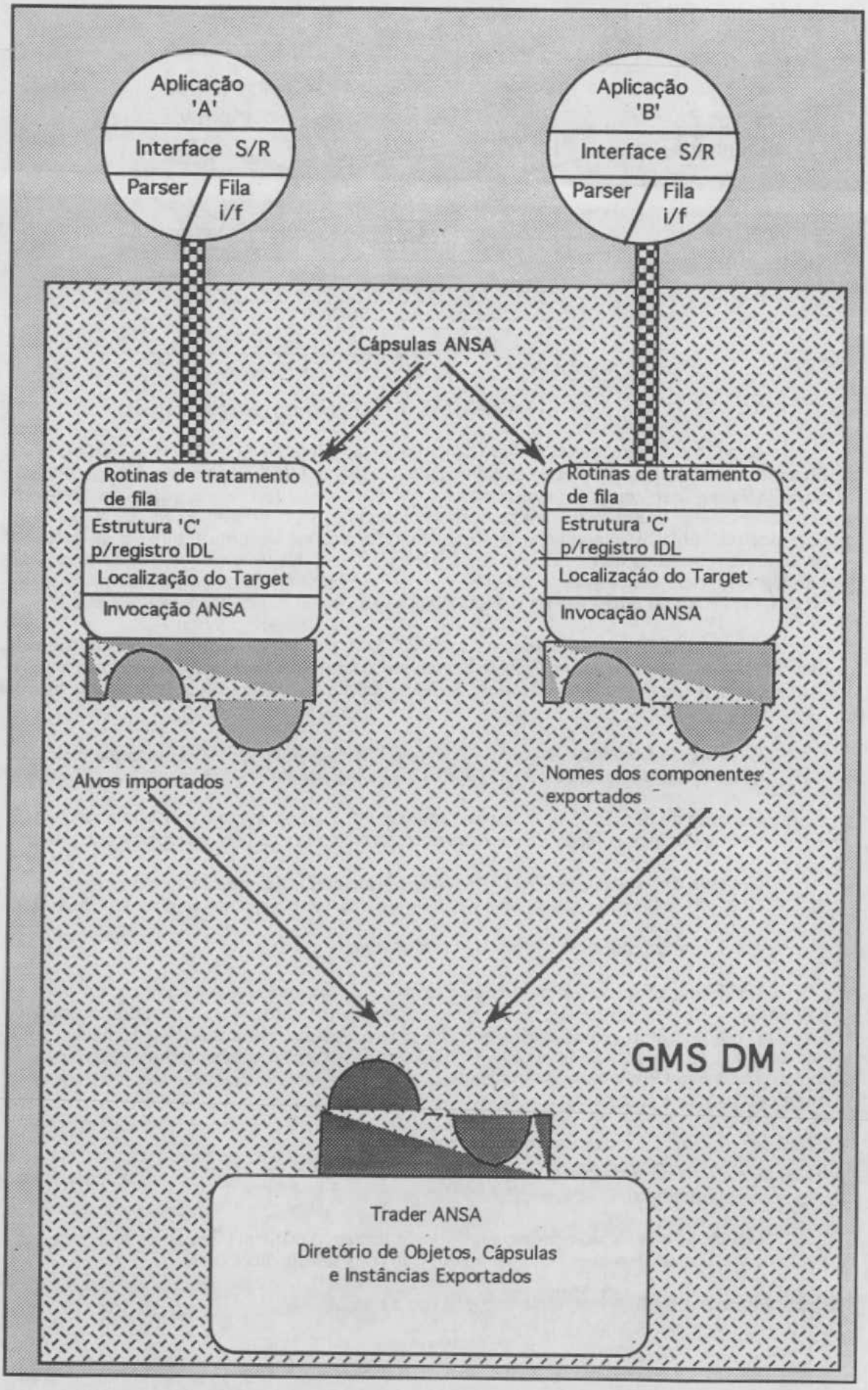
Na definição da arquitetura lógica do NCAS, um módulo é caracterizado por um conjunto de objetos (classes) e pelas relações existentes entre eles. Um módulo contém, também, um conjunto de instâncias de cada classe. Assim, esses módulos fornecem uma interface ao sistema, para efeito de esquema de nomes, através de um conjunto de classes e de relações existentes entre elas, denominado *Container Module* [10]. Para cada interface duas propriedades foram associadas:

Classes = { c | c é uma classe definida no módulo}

Relações = { r | ∃ a relação (ci r cj) no módulo, onde ci e cj são classes definidas no módulo}

Instâncias não foram consideradas porque tornam o esquema de nomes inaceitável do ponto de vista de performance (granularidade) e de consistência dos dados.

(*) Esses blocos (LM, SM, e MM) são implícito na figura a seguir


Aplicação 'A'
Interface S/R
Parser
Fila i/f
Aplicação 'B'
Interface S/R
Parser
Fila i/f
Cápsulas ANSA
Rotinas de tratamento de fila
Estrutura 'C' p/registro IDL
Localização do Target
Invocação ANSA
Rotinas de tratamento de fila
Estrutura 'C' p/registro IDL
Localizaçáo do Target
Invocação ANSA
Alvos importados
Nomes dos componentes exportados
GMS DM
Trader ANSA
Diretório de Objetos, Cápsulas e Instâncias Exportados

5.PROTÓTIPO IMPLEMENTADO

5.1 Descrição do Protótipo no contexto do RM ODP

Neste item nós descrevemos o protótipo, enquadrando-o (sem muito rigor) no contexto do emergente RM ODP (*Reference Model for Open Distribution Processing*) a partir de uma "metáfora" apresentada no Modelo de Empresa (ver a figura mostrada anteriormente):

- **Modelo de Empresa'**

  "Mauro precisa enviar no dia 13/04/92, um 'bonjour' para a secretária eletrônica do 10° SBRC a ser realizado no Recife !!! "
  Aspectos Administrativos:
  - Um contrato MASI-UFPe permite essa atividade;
  - Os Profs. Pujolle e Suruagy concordam com a transação.

- **Modelo de Informação** (*)

  - **Usuário A:** O usuário A precisa acessar informações de administração no recurso B, o qual é modelado como um objeto no CIM.
  - **Recurso:** Fornece um determinado serviço aos usuários do serviço de telecomunicações. Esse Recurso é visível ao Usuário A (de administração) através do CIM (MOM).
  - **CIM:** Fornece às aplicações (Usuário A) de administração uma visão conceitual objetos gerenciados (Recurso) e gerenciadores (MSM).

- **Modelo de Computação**

  - **GMS:** Permite às aplicações acessar e manipular informações no CIM, através de comandos tais como GET, CREATE, ACTION, DELETE e MODIFY.
  - **Aplicação A** : Envia uma mensagem GMS para o CIM via a Plataforma de Computação fazendo uso da Interface S/R.
  - **CIM**: E um bloco funcional que recebe a mensagem GMS da aplicação através da Interface S/R . A Aplicação "B" simula o CIM no protótipo.
  - **Plataforma de Computação:** Fornece mecanismos de transparência de acesso e localização aos componentes (objetos distribuídos) da arquitetura. A Interface S/R permite a conexão desses componentes à Plataforma.
  - **Interface S/R**: A Interface Send/Receive é o único meio pelo qual os componentes da arquitetura podem se comunicar (enviar mensagens GMS). As primitivas disponíveis são: Call, Cast, WeakCast, Reply, StrongCast, etc.

- **Modelo de Engenharia**

  - **Plataforma de Computação:** Corresponde ao Mecanismo de Entrega GMS_DM (*Delivery Mechanism*) descrito no item 4.2. O propósito deste bloco é transferir mensagens GMS e suas respostas ocasionais, entre componentes do sistemas via as Interfaces S/R às quais esses componentes são conectados [12].
  - **Interface S/R:** Conjunto de primitivas e estruturas que permitem a conexão de um componente do NCAS ao GMS_DM. Ela não suporta transações (onde um numero de remote operações são agrupadas juntas tal que se uma operação falha, o todo o grupo retorna ao seu estado original) [11].
  - **GMS_Parser**: Realiza a análise sintática e semântica da mensagem GMS, evitando assim que inválidas mensagens sejam entregues pelo GMS_DM. Ele extrai da mensagem GMS informacoes que facilitam a tarefa do GMS_DM.

(*) E interessante notar os dois tipos de usuários: o do Recurso e o da administração do Recurso.

- **Modelo Tecnológico**

  - **Plataforma de Computação**: O sistema distribuído ANSAware [13], versão 3.0 dá suporte às principais funcões do mecanismo de entrega GMS_DM.
  - **Interface S/R**: Desenvolvida em 'C' [11] ela utiliza um mecanismo de fila (*System V IPC Queue.*) para conectar componentes do NCAS (no ambiente UNIX, por ex.) ao GMS_DM (ANSAWare).
  - **Máquina:** As máquinas utilizadas são estações de trabalho SPARCstation IPC, sistema operacional SunOS versão 4.1.

5.2 Interface do Protótipo

Nesse item é apresentada a interface disponível ao Programador de aplicações de administração (Usúario do NCAS) [11]:

- **Interface S/R:** As primitivas implementadas são:

  Call: E usada para enviar uma mensagem GMS para a qual uma resposta (REPLY) é esperada. O processo usuário desta primitiva fica então bloqueado até a chegada de uma resposta (REPLY or USER_REJECT or SERVICE_REJECT) ou da expiração do relógio (*timeout/blocking factor*).

  Cast: E usada para enviar uma GMS messagem para a qual nenhuma resposta é esperada. A aplicação envia a mensagem e continua seu processamento.

  WeakCast, StrongCast: São variações (bloqueantes) do Cast.

  Reply,UserReject: Usadas pelas aplicações para o envio de resposta.

  IncommingPDU: Usada pela aplicação para receber uma GMS_PDU.

- **Exemplo**: Primitiva Call

```
typedef char   *GMSMessage;

int  Call (Message, Context, Result, TransitDelay)
GMSMessage        Message;
GMSMessage        *Result;
int               Context;          /* CIM, Generic Services, etc*/
int               TransitDelay;     /* timeout-blocking type parameter */
```

- **Estrutura da GMS_PDU:** Para se transferir mensagens GMS entre componentes do sistema, essas mensagens e suas ocasionais respostas são encapsuladas em uma estrutura "C" denominada GMS_PDU (*Protocol Data Unit*)[12]. Cada GMS_PDU contém a mensagem e informações de controle como mostrado a seguir:

```
typedef struct {
                int     PDU_Size;
                int     PDU_Type;
                int     ParsedOrNot;
                int     Context
                char    *TargetName;
                char  *Source.Name;
                int     GMSBitstreamSize;
                char    *GMSBitstream;
                } GMS_PDU;
```

5.3 Fluxo da mensagem GMS no Protótipo

Nesta seção apresentamos a descrição funcional dos componentes que permitem a um componente do NCAS (no caso uma aplicação de administração - MA) se conectar ao Mecanismo de entrega (GMS_DM) da plataforma de computação. Consideramos o mesmo cenário descrito no Modelo de Empresa (item 5.1) : uma Aplicação "A" envia uma mensagem GMS para o CIM, através da Interface S/R (ver a figura mostrada anteriormente).

Aplicação "A" --> Interface S/R : A aplicação "A" envia uma mensagem GMS fazendo uso de uma função disponível na Interface S/R, no caso, a primitiva CAST [11]. CAST cria um "chave" única para acesso ao mecanismo de fila (System V IPC Queue). A aplicação "A" continua o seu processamento normal (CAST é uma primitiva não bloquante).

Interface S/R <--> PARSER : A Interface S/R (primitiva CAST) envia a mensagem GMS ao Parser [6]. Este analisa a mensagem GMS de maneira a rejeitar mensagens inválidas e retorna algumas informações, em especial o *TargetName* (ver acima GMS_PDU) facilitando, assim, a tarefa de localização a ser realizada pelo GMS_DM.

Interface S/R --> Fila I/f : Uma vez recebida de volta a mensagem GMS analisada pelo Parser, a Interface S/R a envia em direção ao GMS_DM, via o componente Fila I/f. Este componente é responsável pela encapsulamento (*packing/ unpacking*) da mensagem numa GMS_PDU [12].

Fila I/f --> Fila de mensagem : O componente Fila I/f ativa uma função (*CallToPlatform*) [12] para enviar a GMS_PDU para a Fila de mensagem, a qual associa a Aplicação uma Cápsula ANSA.

Fila de mensagem --> Rotina de tratamento de fila : Esta rotina (localizada na Cápsula ANSA) lê a estrutura GMS_PDU da fila, permitindo a mensagem GMS enviada pela aplicação "A", acessar o GMS_DM.

Estrutura 'C' --> Registro IDL : Dentro da Cápsula ANSA, a estrutura "C", recebida da fila, deve ser traduzida em uma estrutura IDL (ver item 4.1) [13].

Locatização do Target : Toda mensagem precisa ser tratada, no sentido da identificação de seu destino (ver item 4.2) [10].

Invocação ANSA --> Correntemente, o nome que permite ao GMS_DM identificar o destino da mensagem (*TargetName*) deve especificar um objeto classe dentro do MOM, uma instância dentro do MSM ou um objeto classe dentro do MOT (ver item 3.2) [12].

Trader ANSA --> o serviço de *trading* contém operações que habilitam os objetos de engenharia a registrarem os serviços que eles oferecem (operação **export**) e a recuperarem aqueles que eles eventualmente queiram usar (operação **import**). No caso, o *trading* faz o "matching" entre a Cápsula correspondente a Aplicação A e a Cápsula correspondente ao CIM (simulada pela Aplicação B).

6 CONCLUSÃO

A Equipe *Reseaux et Performance* do Laboratório MASI tem contribuído para os Grupos de Trabalhos do Projeto ADVANCE, desde 1989. O trabalho aqui descrito é resultado de nossa participação no estudo da Arquitetura do NCAS (*Network and Customer Administration System*) [8], nos projetos e implementações de uma Interface para MAs [11] e do Mecanismo de Entrega de mensagens GMS_DM (*Generic Message System Delivery Mechanism*) [10], como partes de um protótipo dessa arquitetura.

Neste Trabalho descrevemos a Arquitetura lógica do NCAS e a Plataforma de Computação desenvolvida, com ênfase nos Modelos de Computação e de Engenharia do RM ODP (*Reference Model for Open Distribution Processing*). O sistema ANSAware que serviu de suporte à Plataforma de computação implementada foi sumariamente apresentado. Ele é ainda um sistema em evolução. Diversas facilidades são previstas para versões futuras, tais como gestão de objetos distribuídos, um protocolo permitindo execuções remotas e a gestão de transações.

A Arquitetura ANSA tem tido uma influência decisiva na padronização de um modelo de referência para processamento distribuído (RM ODP) pela ISO. Esse fato tem sido uma forte motivação para seu uso nos protótipos desenvolvidos no Laboratório MASI.

Um módulo (não descrito neste trabalho) que atualiza o estado do CIM (eventos não solicitados) [14] também foi implementado e integrado ao protótipo aqui apresentado. Atualmente, um outro protótipo para o estudo de NCAS em redes de longa distância (WAN) está em progresso no MASI[15].

REFERÊNCIAS

[1] An Implementation Architecture for the TMN for the CEC RACE Programme.Project R1003. Document 03/CAS/SAR/DS/B/001/b1.

[2] Initial User Requirements. ADVANCE Consortium - Deliverable 1. Document 09/BCM/RD3/DS/C/002/A1. Broadcom, Dublin. Dec 88.

[3] Farley P. A Logical Architecture for NCAS. Document ADBT365. ADVANCE Project. British Telecom. Ipswith UK. January 90.

[4] ISO/IEC JTC1/SC21/WG1 Basic Reference Model of Open Distributed Processing Rapporteur Group, ISO, April 87.

[5] ANSA Reference Manual - Volume A. Architecture Projects Management Limited. Cambridge, U.K. 1989.

[6] Shomaly R. GMS-PV2 Message Parsing Definition. Document ADBT385. ADVANCE Project. British Telecom.Ipswitch UK. April 91.

[7] ISO/IEC 9594-1990. Information Technology - OSI Directory - Part 1: Overview of Concepts, Models and Services.

[8] De Sousa J.N. Information Transfer within the ADVANCE Logical Architecture. ADDN20.Laboratoire MASI.Université Paris 6. July 91.

[9] Brodie M. & al. On Conceptual Modelling. Spring-Verlag. 1984.

[10] Celestino J: & al. The locating Problem in the GMS_DM. ADDN19 document. ADVANCE Project. Laboratoire MASI. Université Paris 6. July 91.

[11] Oliveira AMB. Visible GMS Interface - Installation Notes 1 User's Guide (Release 2.1). Doc ADDN37. ADVANCE Project. Laboratoire MASI. Université Paris 6. July 91.

[12] Harkness D. The GMS_DM Design. ADPL175 document. ADVANCE Project. Roke Manor Research. Hampshire UK. May 91.

[13] ANSAware 3.0. Implementation Manual. Document RM.097.00. Architecture Projects Management Limited. Cambridge, UK 1991.

[14] Lalagne S. & Agoulmine N. The Model Updater. ADDN17 document. ADVANCE Project. Laboratoire MASI. Université Paris 6. June 91.

[15] Oliveira AMB & Agoulmine N An Engineering View on the Inter-TMN Cooperation. ADVANCE Project. ADDN036 doc. Laboratório MASI. Université Paris 6. Sep 91.