Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Tesis de Maestría
en Ingeniería en Computación

Arquitectura para interconexión de Building Blocks en Sistemas de Gestión de Redes y Servicios de Telecomunicaciones

Gerardo Gándara

Director: Dr. Raul Ruggia

Montevideo, Uruguay
Mayo de 2004

Arquitectura para interconexión de
Building Blocks en Sistemas de Gestión de
Redes y Servicios de Telecomunicaciones
Gerardo Gándara

ISSN 1510-7264
Tesis de Maestría en Ingeniería en Computación
Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay, Mayo de 2004

Agradecimientos

Deseo agradecer a todas las personas que colaboraron en forma directa o indirecta con el presente trabajo. En primer al Dr. Raúl Ruggia (Tutor de este trabajo) por sus aportes, guías y sugerencias.

Es de destacar que parte de los contenidos de la presente tesis se desarrollaron en el marco del convenio de Investigación y Desarrollo establecido entre Facultad de Ingeniería y Antel. En este sentido agradezco la colaboración y aportes de todos los integrantes del equipo, fundamentalmente los Ingenieros Pablo Belzarena, Gabriel Gómez, Leonardo Rodríguez, Fernando Rodríguez, Andrés Vignaga, Daniel Perovich, Gonzalo Tejera, Raúl Ruggia (en calidad de integrante del equipo) y estudiantes que participaron por parte de Facultad de Ingeniería. También se destacan los aportes valiosos de los Ingenieros de Antel, Stella Bonino, Eduardo Spremolla, Gonzalo Montilla y Ariel Infantozzi.

Una referencia especial está dedicada al Ingeniero Jorge Gallo (Antel) por ser impulsor de investigaciones en esta área además de brindar aportes concretos al desarrollo del presente trabajo.

Abstract

En el presente trabajo se presenta una arquitectura orientada a la construcción de sistemas para la gestión de redes y servicios en el área de las telecomunicaciones (permitiendo la integración de los mismos en una arquitectura común).

La propuesta presentada pretende dar un marco para la generación de sistemas basados en Building Blocks (componentes de granularidad gruesa que ofrecen servicios a través de Contratos). Se resuelve el problema de interacción entre Building Blocks a través del uso de un "vehículo de comunicaciones común" lo cual permite esconder los detalles de la implementación final. Este vehículo es una abstracción de distintos medios de transporte (desde el punto de vista tecnológico) y modalidades de interacción (sincrónico, asincrónico, etc.).

Este enfoque permite achicar la distancia entre el diseño lógico neutral y su implementación y por otro lado facilitar el reuso a nivel de aplicación además de constituir un refugio ante cambios tecnológicos (al mantener desacoplada la implementación de las particularidades de la plataforma).

Con la finalidad de verificar la aplicabilidad de la arquitectura propuesta se desarrolló un caso de estudio que aborda el problema de la gestión de fallos en la red. Fundamentalmente se atiende el problema del procesamiento de mensajes de alarma que provienen de equipos remotos a los efectos de que dicha información quede disponible para los operadores del sistema. La implementación realizada se apoya en las tecnologías J2EE y CORBA utilizando sistemas de mensajería JMS y "Notification Service" de CORBA.

PALABRAS CLAVES:
GESTION DE REDES Y SERVICIOS, BUILDING BLOCK, INTEROPERABILIDAD, CONTRATO

Índice General

Índice de Figuras

Abreviaturas Empleadas

AA *Application Architecture*
ADSL *Asymetric Digital Subscriber Loop*
ASN.1 *Abstract Syntax Notation 1*
ATM *Asynchronous Transfer Mode*
BB *Building Block*
BML *Business Management Layer*
BPR *Business Proceses Re-engineering*
CAMI *Common Application Management Interface*
CIM *Common information Model*
CMIP *Common Management Information Protocol*
CMIS *Common Management Information Services*
COPS *Common Open Policy Service*
CORBA *Common Object Request Broker Architecture*
COTS *Commercial Off The Shelf*
CP *Contract Providers*.
DCF *Data Communications Functions*
DCN *Data Communication Network*
DCOM *Distributed Common Object Model*
DMTF *Distributed Management Task Force*
EIT *Enterprise Information Tier*
PAT *Process Automation Tier*
HIT *Human Interaction Tier*
EJB *Enterprise Java Beans*
EL *Element Layer*
EML *Element Management Layer*
FAB *acrónimo de Fullfillment, Assurance, Billing*
FCAPS *Fault, Configuration, Accounting, Performance y Security*
GDMO *Guidelines for the Definition of Managed Objects*
IDL *Interface Definition Language*
IM *Iinformation Models*
IRTF *Internet Research Task Force*
ISV *Independent Software Vendors*.
ITU-T *Iternational Telecommunications Union*
JVT *Java Value Types*
JIDM *Joint Interdomain Management Network*
JMS *Java Messaging System*
JNDI *Java Naming and Directory Interface*
JSP *Java Server Pages*
LDAP *Light Weight Directory Access Protocol*
LLA *Logical Layered Architectur*
MBD *Model Based Development*
MD *Mediation Devices*
MDA *Model Driven Architecture*
MDB *Message Driven Bean*
MF *Mediation Functions*
MIB *Management Information Base*
MO *Managed Object*
MOF *Meta Object Facility*
MOM *Message Oriented Middleware*
NE *Network Element*
NEF *Network Elements Functions*
NGOSS *New Generation Operations Support Systems*
NML *Network Management Layer*
OCL *Object Constraint Language*
ODP *Open Distributed Processing*

OMG *Object Management Group*
ORB *Object Request Broker*
OS *Operations System*
OSF *Operation Systems Functions*
OSI *Open Systems Interconnect*
OSI-RM *Open Systems Interconnect-Reference Model*
OSS *Operational Support Systems*
OSS *Operations and Support Systems*
PBNM *Policy Based Network Management*
PIM *Platform Independent Model*
PSM *Platform Specific Model*
QA *Q-Adapters, QA*
QAF *Q-Adaptation Functions*
RMI *Remote Method Invocation*
SDH *Synchronous Digital Herarchy*
SIC *Semanti Integrity Constraints*
SID *Shared Information and Data*
SLA *Service Level Agreement*
SMI *Structure of Management Information*
SML *Service Management Layer*
SNMP *Simple Network Management Protocol*
SOAP *Simple Object Access Protocol*
SOP *Service Oriented Programming*
SP *Service Providers*
TFTP *Trivial File Transfer Protocol*
TINA-C *Telecommunications Information Networking Architecture Consortium*
TMForum *TeleManagement Forum*
TMN *Telecommunication Management Network*
TNCS *Technology Neutral Contract Specification*
TOM *Telecommunications Operations Map*
TSCS *Technology Specific Contract Specification*
UML *Unified Modelling Language*
UCD *User Centered Design*
WS *Workstations*
WSDL *Web Service Description Language*
WSF *Workstations Functions*
XML *Exchange Machine Language*

Capítulo 1

1 Introducción

1.1 Contexto general

Las empresas de telecomunicaciones están afrontando grandes desafíos y oportunidades como nunca. El escenario actual en que se mueven está caracterizado por una competencia creciente (producto de un nuevo marco de desregulación en general) con altas exigencias del mercado.

Muchas empresas intentan automatizar procesos que se realizaban originalmente en forma manual, pasando a esquemas que permitan poner el acento en las necesidades del cliente, calidad del servicio, costos y velocidad de llegada al mercado con nuevos productos *(time to market)*. Esto implica un escenario totalmente nuevo donde surge la necesidad de interoperar con la propia competencia y/o proveedores de servicios externos, además de disponer de estructuras internas de manejo de la información que permitan introducir nuevos servicios con valor agregado y obtener mejoras superlativas en el cuidado y atención de los clientes. En este escenario la gestión adecuada de redes y servicios es primordial.

1.2 La Gestión de Redes y Servicios en Telecomunicaciones

La gestión de redes y servicios se puede definir como el conjunto de procesos y actividades que realiza una operadora para ofrecer a sus clientes los servicios de telecomunicaciones, de tal forma que se cumplan tanto los criterios de calidad y costos establecidos en los objetivos de la empresa, como los reflejados en los correspondientes contratos con los clientes.

A medida que las redes y servicios fueron creciendo en complejidad fue necesario recurrir a la automatización de distintos procesos, estructurándose de esta manera Sistemas de Gestión Automatizados.

En este sentido el área de gestión de redes ha ido evolucionando. Diversos aportes se han realizado, desde la propia ITU-T[1] *(Iternational Telecommunications Union)* a partir del modelo TMN (*Telecommunication Management Network).* El modelo TMN proporciona una arquitectura de referencia para el intercambio de información de gestión entre los sistemas de operación y/o los equipos.

Allí se definen conceptos de las arquitecturas de la red de gestión de las telecomunicaciones (arquitectura funcional, arquitectura de información, arquitectura estratificada lógica y arquitectura física) y sus elementos fundamentales.

1.3 Uso de las Nuevas Tecnologías de Información en la Gestión de Redes y Servicios

Al generalizar la automatización de procesos llevados a cabo por una empresa de telecomunicaciones, las arquitecturas de gestión "históricas" (mencionadas anteriormente) deben ser complementadas con una estrategia adecuada para lograr la integración de los distintos sistemas involucrados.

En este sentido, algunas organizaciones como *TeleManagement Forum* encaran el problema en una perspectiva global de la empresa, analizando la totalidad de los procesos que se llevan a cabo dentro de la misma, considerando desde los procesos que tiene que ver con la interacción con los clientes (que solicitan servicios específicos), hasta la configuración y operación de la propia red y recursos en general involucrados. Este encare va acompañado de la definición de un modelo de información.

---

[1] ITU-T *(Iternational Telecommunications Union)* es el órgano rector de las telecomunicaciones a nivel de ONU.

La construcción de sistemas de operaciones se lleva a cabo mediante la articulación de distintos "bloques" donde cada uno de ellos implementa procesos específicos, brindando servicios a otros bloques y utilizando eventualmente servicios disponibles brindados por terceros.

El diseño de sistemas basados en Bloques Constructivos *("Building Blocks")* no es un concepto nuevo. A modo de ejemplo, desde hace mucho tiempo a nivel industrial, se estructuran máquinas complejas compuestas de partes más pequeñas encapsuladas apropiadamente con interfaces adecuadas de manera de permitir el reuso de las mismas en otros diseños.

La implementación efectiva de este nuevo enfoque ha sido posible por el desarrollo que ha tenido el área de las Tecnologías de Información y *e-business*, fundamentalmente en lo que tiene que ver con el diseño basado en componentes (*Component Based Design*), técnicas de computación distribuida basada en el concepto de objetos distribuidos o el paradigma de programación orientada a Servicios (*Service Oriented Programming)*. Estas técnicas permiten proveer interfaces de programación que esconden en cierto grado la distribución, facilitando el diseño y la implementación de sistemas. Son igualmente valiosas las tecnologías existentes para manejar flujos de procesos (*Workflows*) que permiten separar la definición de procesos de la propia implementación posibilitando la reconfiguración del sistema ante cambios de requerimientos.

A lo largo del presente trabajo se hace referencia al término ***Building Block*** en forma frecuente. Es de destacar que la intención de su uso es la presentada anteriormente. A grandes rasgos, se refiere a un agrupamiento de implementaciones de Servicios que llevan a cabo procesos específicos. Los servicios brindados están caracterizados por "Contratos" donde se especifican las características de las interfaces y la forma en que se deben usar.

1.4 Visión General.

El problema de la construcción de sistemas para soportar la gestión de redes y servicios de telecomunicaciones es complejo. Es un problema de *n*-dimensiones donde se cruzan planos de gestión de procesos, funcional, arquitectónico, etc.

Los encares mas ricos de la actualidad parten del análisis de los procesos extremo a extremo requeridos para llevar a cabo los servicios brindados por la empresa. Se analiza la cadena de valor, visualizándose la interacción con otros actores externos como clientes y proveedores de servicios.

Sin embargo, la realidad de la complejidad y tamaño de sistemas de apoyo y operaciones *(Operations and Support Systems, OSS)* usados en el ámbito de gestión de comunicaciones y servicios hace que no se pueda confiar en componentes particulares, o aún más confiar en un único conjunto de estándares de interoperabilidad. Los sistemas se van construyendo en distintos tiempos usando distintas tecnologías adhiriendo a distintos regímenes.

Es común que las especificaciones de interfaces existentes estén atadas a tecnologías específicas lo cual reduce la longevidad de la especificación y atenta contra la interoperabilidad con componentes existentes en el largo plazo, cuando se produce la evolución del sistema en general.

El problema de las dependencias tecnológicas en el proceso de diseño de un sistema en una fase temprana ha ido más allá del dominio de la gestión de las comunicaciones. Cada vez es mas fuerte la promoción de la idea del desarrollo basado en Modelos *(Model Based Development)* como por ejemplo a nivel del OMG, se ha estandarizado la arquitectura MDA *(Model Driven Architecture)*[15]. MDA promueve la generación de modelos independientes de la tecnología utilizando el lenguaje UML *(Unified Modelling Language)*[13] en distintos puntos del desarrollo del ciclo de vida del software, como por ejemplo, los requerimientos, análisis del sistema, diseño, etc. Esto habilita que la conexión con tecnologías específicas se produzca lo más tarde posible (en la etapa de la propia implementación).

Han existido muchas propuestas para el modelado independiente de la tecnología en distintos niveles de abstracción en el dominio de la gestión de las telecomunicaciones. TINA-C *(Telecommunications Information Networking Architecture Consortium)* encaró el tema basado en la arquitectura ODP *(Open*

Distributed Processing)[10], aplicado tanto al control como al software de gestión. Al usar ODP se produce una perfecta separación de ámbitos, de tal manera que los asuntos vinculados a las tecnologías de procesamiento distribuido utilizados están separados del modelado de requerimientos del negocio, estructuras de información, interfaces de componentes, etc.

El DMTF *(Distributed Management Task Force)* ha estandarizado (con éxito) un modelo de información común CIM *(Common information Model)*[3] para las interfaces de gestión (en el ámbito empresarial). El modelo CIM especifica los objetos gestionados y sus asociaciones utilizando el lenguaje MOF *(Manager Object Format)*, el cual puede ser parcialmente representado como diagramas de clases UML. Otros estándares se han desarrollado para definir como se pueden implementar las interfaces basadas en el modelo CIM, utilizando distintas tecnologías como RPC *(Remote Procedure Call)* en el entorno de procesamiento distribuido (DPE), Servicios de Directorio u objetos codificados usando XML y transportados por protocolo HTTP.

El TMForum ha estandarizado requerimientos y modelos en términos neutrales. Primero capturando modelos de procesos del negocio genéricos en el dominio de las telecomunicaciones, expuestos a través del e-TOM (*enhanced Telecom. Operations Map*). Actualmente el TMForum está en proceso de estandarizar interfaces de componentes y la información que circula por dichas interfaces. Estos trabajos se están llevando a cabo en el ámbito de la iniciativa NGOSS (nueva generación de Sistemas de Soporte para Operaciones), sin embargo estos trabajos se están manejando utilizando conceptos arquitectónicos de alto nivel. Hay pocas pautas para el desarrollo e integración de Building Blocks, o por otro lado guías para reconciliar elementos como Procesos del Negocio, Especificaciones de Contratos y Modelo de Información compartida que han sido estandarizados por separado.

Existe en el ámbito comercial, la propuesta de Sun OSS/J, que recoge los principios de NGOSS y propone un conjunto estandarizado de interfaces para resolver problemas en dominios específicos, como Recolección de información para facturación o"*ticketing*", Provisión de servicios, Gestión de SLA *(Service Level Agreement)*. Esta plataforma está basada en la arquitectura J2EE.

1.5 Motivaciones

Un problema importante a resolver es la integración de aplicaciones. Se busca una solución integral que abarque todos los ámbitos de una empresa de telecomunicaciones: ya sea desde la obtención de alarmas desde los dispositivos de bajo nivel hasta la posibilidad de ofrecerle al usuario final que pueda establecer su solicitud de reclamo o pueda tramitar un nuevo servicio utilizando como interfase un navegador de Internet. Las empresas deben tener sistemas que permitan fácilmente la creación de nuevos servicios o el agregado de nuevos elementos de red (que a su vez deben ser mantenidos y configurados).

Es un dato de la realidad que los sistemas existentes manejan interfaces y tecnologías diversas por lo que se requiere la adaptación de los mismos a una arquitectura común.

Existe una proliferación de componentes, esquemas de interoperabilidad y entornos distribuidos, evolución de lenguajes de programación y diferentes estándares para el modelado de aplicaciones. Es riesgoso implementar soluciones que dependan estrictamente de la plataforma elegida.

En este sentido, desde el punto de vista de la construcción del software, uno de los problemas mas críticos es la habilidad para manejar estas nuevas tecnologías. Resulta esencial que las aplicaciones permanezcan estables en el tiempo de forma de justificar los costos de desarrollo, por lo que se requieren cambios en el proceso de diseño y desarrollo del software.

Para proteger la inversión es necesario encarar el diseño en términos neutrales partiendo de modelos independientes de tecnologías específicas (de manera que los resultados continúen siendo válidos para múltiples escenarios). Complementariamente, se requiere de una estrategia de trabajo que achique la distancia entre un diseño lógico neutral y las múltiples implementaciones que se puedan llegar a llevar a cabo.

Otro de los principales asuntos al encarar aplicaciones de telecomunicaciones es el reuso. Las redes de telecomunicaciones han sido tradicionalmente propietarias, desarrolladas como el resultado de alianzas estratégicas entre productores de equipamiento y operadores de red de gran porte. Es real que aplicaciones similares se han tenido que desarrollar muchas veces en cada plataforma propietaria. La situación no va a cambiar a corto plazo debido a grandes inversiones realizadas en tecnologías de red propietarias. Peor aún, la cantidad de servicios y aplicaciones va creciendo y el problema se hace cada vez más crítico. En este sentido existe la necesidad urgente de posibilitar el reuso de aplicaciones en el contexto de distintas plataformas propietarias.

Para una arquitectura basada en *"Building Blocks"*, es de gran valor el disponer de un "framework común" que permita que las implementaciones realizadas se puedan reusar en el contexto de distintas tecnologías. Esto permitirá derivar rápidamente desde un diseño lógico en términos neutrales a implementaciones concretas.

1.6 Propuesta realizada.

1.6.1 Objetivos del presente trabajo

El objetivo de este trabajo consiste en definir una arquitectura para procesamiento distribuido sobre la cual se pueda implementar procesos de gestión de redes y servicios de telecomunicaciones.

Para ello se deberá:

- Realizar un análisis del estado del arte en requerimientos y arquitecturas generales para Sistemas de Gestión de Telecomunicaciones. Para esto se hará un relevamiento de recomendaciones y estándares existentes promovidos por organizaciones como ITU-T (Órgano rector de las telecomunicaciones a nivel de ONU) y otras entidades como TMForum, TINA-C, OMG, etc.
- Realizar un relevamiento de tecnologías existentes, aplicables al caso.
- Formulación de un diseño de arquitectura que permita el funcionamiento de sistemas basados en Building Blocks, con opciones tecnológicas justificadas.
- Caso de estudio Gestión de Alarmas, donde se intenta aplicar los conceptos manejados.

1.6.2 Aportes del Presente Trabajo

La propuesta arquitectónica de interoperabilidad entre Building Blocks que se presenta pretende fundamentalmente achicar la distancia entre el diseño lógico neutral y su implementación y por otro lado facilitar enormemente el reuso a nivel de aplicación.

Si bien la arquitectura propuesta en el presente trabajo está basada en las recomendaciones NGOSS de TMForum, el encare es diferente. En general las recomendaciones de NGOSS y las implementaciones existentes (como OSS/J) manejan estrategias para implementar los servicios en distintas tecnologías. Algunas propuestas comerciales directamente utilizan una tecnología específica, siendo necesaria la adaptación para interactuar con otra tecnología.

En el presente trabajo la implementación de los servicios ofrecidos por Building Blocks se hace sobre la abstracción de un entorno de ejecución genérico de manera que la misma implementación es reusable directamente en distintos entornos de ejecución. Esto posibilita que los servicios brindados por un Building Block dado puedan ser migrados desde la plataforma de un servidor de aplicaciones J2EE por ejemplo a los confines remotos de la red vía CORBA, o que estén disponibles en la aplicación que utiliza el operador *(front-end)*.

Además de reuso de implementaciones de Building Blocks, existe potencialmente la necesidad de reubicación del mismo en el contexto de otra plataforma. Las razones para ello pueden tener que ver con estrategias para la optimización de los recursos empleados, tolerancia a fallos, madurez de las tecnologías elegidas (pueden surgir problemas en el uso de la tecnología en un escenario de gran escala). Pueden

existir razones de performance. Tampoco es de descartar razones que tienen que ver con los costos de licenciamiento de plataformas de servidores de aplicación.

Si bien la arquitectura propuesta resuelve los problemas de interoperabilidad para tecnologías específicas se plantea como objetivo que el framework (soporte de la arquitectura) sea abierto y que sea muy sencillo extenderlo de manera de cubrir otros patrones de interoperabilidad.

1.6.3 Áreas no abordadas

No está dentro del alcance del trabajo abordar la gestión de servicios de granularidad fina que se llevan a cabo entre gestores y agentes de redes de telecomunicaciones. Inicialmente se manejaron protocolos específicos como CMIP y SNMP, luego CORBA (JIDM), actualmente surgió JAIN como una plataforma basada en java para la gestión de agentes entre otros. TMForum tiene la visión de NGOSS de granularidad fina fuertemente apoyada en JINI[2]. En un sentido más amplio suele referirse a estos servicios con las denominaciones *"Ubiquitous computing"* o *"Pervasive Computing"* al aplicarse a dispositivos distribuidos de uso general.

Tampoco está abordada la problemática de gestión de redes IP, que responde a un tema amplio con muchas aristas. Estas redes originalmente fueron pensadas sobre la base de esquemas del mejor esfuerzo, que no son adecuados para brindar servicios de calidad sobre la red. Actualmente existe toda una área de trabajo vinculada a los soportes de transmisión, utilizando nuevos protocolos de ruteo (MPLS por ejemplo) que permiten la aplicación de políticas adecuadas para la gestión de servicios de calidad aceptable. Existen trabajos en el área del paradigma de gestión basada en políticas o PBNM *(Policy Based Network Management)* y proponiendo nuevas funcionalidades a través del protocolo COPS *(Common Open Policy Service)*.

1.7 Organización del documento

El trabajo está organizado de la siguiente manera:

En el Capítulo 2 se realiza un relevamiento del estado del arte del área de gestión de redes y servicios en el área de las telecomunicaciones, fundamentalmente en lo referente a la construcción de OSS, basados en servicios de granularidad "gruesa" agrupados en Building Blocks.

En el Capítulo 3 se presenta una arquitectura de integración de Building Blocks que permita articular en forma flexible y dinámica la interoperabilidad entre Building Blocks.

En el Capítulo 4 se aborda un caso de estudio de la gestión de fallos en la red donde se intenta verificar la aplicabilidad de la arquitectura propuesta.

Finalmente en el Capítulo 5 se presenta un resumen de lo expuesto en el trabajo y consideraciones sobre trabajos futuros a realizar.

Complementariamente, existe un Anexo donde figura el análisis de casos de uso del caso de estudio.

[2] JINI. Tecnología de Red desarrollada por Sun basada en Java. Es una arquitectura abierta que permite el desarrollo de servicios centrados en la red (network centric services) que se implementan en software o el propio hardware (dispositivos embebidos) .

Capítulo 2

2 Estado del Arte

2.1 Introducción

El área de la Gestión de Redes y Servicios es compleja, existiendo distintas visiones y dimensiones del problema a abordar. En este capítulo se intenta hacer un relevamiento muy sintético del estado del arte en esta área.

Se presenta inicialmente conceptos básicos y el contexto de la problemática del área de Gestión de Redes y Servicios. Se presenta una visión general (resumida) de aportes realizados por organizaciones como ITU-T y TMForum.

Posteriormente se presentan requerimientos para la construcción de sistemas basados en Building Blocks y las características de las Arquitecturas requeridas. Se presenta como referencia la Arquitectura NGOSS de TMForum.

Complementariamente se presentan algunos enfoques existentes respecto al análisis y diseño basado en modelos (en términos neutrales) aplicado al área de gestión de redes y servicios.

Finalmente se hacen consideraciones sobre las elecciones tecnológicas posibles a la hora de la implementación de los sistemas.

2.2 Los Sistemas de Gestión de Redes y Servicios

2.2.1 Contexto General

2.2.1.1 Gestión de Redes y Servicios
2.2.1.2 Clientes y Servicios son el eje de procesos de gestión en la actualidad
2.2.1.3 Contexto de las empresas de Telecomunicaciones
2.2.1.3.1 Análisis de procesos de gestión
2.2.1.3.2 El desarrollo de software
2.2.1.4 Las plataformas de desarrollo, ejecución de aplicaciones distribuidas y Servidores de Aplicaciones.

2.2.2 TMN (ITU-T)

2.2.2.1 Introducción
2.2.2.2 Marco de trabajo propuesto por TMN
2.2.2.2.1 Arquitectura funcional (punto de vista lógico)
2.2.2.2.2 Arquitectura de información (punto de vista de la gestión funcional)
2.2.2.2.2.1 Visión de “Recursos” como objetos gestionados
2.2.2.2.2.2 La base de datos de información de gestión
2.2.2.2.2.3 Mecanismos de Interoperabilidad Gestor-Agente
2.2.2.2.2.4 Plano de funciones de aplicación de gestión (FCAPS)
2.2.2.2.3 Arquitectura Física (punto de vista de comunicaciones entre componentes).
2.2.2.2.4 Arquitectura estratificada lógica
2.2.2.3 TMN vs. Gestión en Internet

2.2.3 TM Forum

2.2.3.1 Introducción
2.2.3.2 Otra visión de TMN
2.2.3.3 Áreas de trabajo del TMForum
2.2.3.4 Mapa de Procesos en las Empresas de Telecomunicaciones: TOM (Telecom Operations Map)

2.2.4 Conclusiones

2.2.1 Contexto General

2.2.1.1 Gestión de Redes y Servicios

La gestión de redes y servicios se puede definir como el conjunto de procesos y actividades que realiza una operadora para ofrecer a sus clientes los servicios de telecomunicaciones, de tal forma que se cumplan tanto los criterios de calidad y costos establecidos en los objetivos de la empresa, como los reflejados en los correspondientes contratos con los clientes.

A medida que las redes y servicios fueron creciendo en complejidad fue necesario recurrir a la automatización de distintos procesos, estructurándose de esta manera Sistemas de Gestión Automatizados.

El disponer de sistemas de gestión por sí solo no es garantía de éxito. La posibilidad de supervisar las alarmas de los equipos de red permite determinar de forma sencilla el equipo que ha fallado, pero esto no soluciona el problema. Es necesario definir los mecanismos y procedimientos adecuados con la finalidad de llevar a cabo las acciones necesarias para resolver el problema. En este sentido, hay que definir los procesos de negocio: ¿qué hacer ante una reclamación de usuario?, ¿cómo actuar ante la aparición de una alarma?, etc., y hacer que los sistemas presten el soporte adecuado a estos procesos.

La gestión de las redes inicialmente se desarrollaba bajo el paradigma de la "telemetría" bajo el cual los elementos de red enviaban notificaciones continuas a unidades centrales de monitoreo.

A mediados de los 80 ITU-T *(Internacional Telecommunications Union)* formaliza el TMN *(Telecommunications Management Network)*[11], como un modelo para estructurar en forma lógica las actividades del negocio. Este modelo inicialmente proporciona una arquitectura de referencia para intercambio de información de gestión entre los sistemas de operación y equipos. El modelo TMN está basado en el modelo OSI[3] para la interconexión de sistemas abiertos, que adopta el modelo gestor - agente para las relaciones entre sistemas o entre sistemas y equipos. El modelo inicialmente considera la conexión de sistemas desde tres aspectos:
1. *Funcional*. Define las actividades que hay que realizar y la organización de las mismas.
2. *De información*. Modela la información de gestión que se intercambia entre el gestor y el agente. Este modelo depende de las funciones que se realicen y de los recursos que se quieran gestionar.
3. *De comunicación*. Especifica los protocolos de comunicaciones utilizados para el intercambio de información entre sistemas. Su objetivo es permitir la transferencia e interpretación correcta de la información de gestión.

En 1996 ITU-T agrega un cuarto aspecto que ha sido reconocido por todos los foros y constructores:

4. *De estratificación lógica.* Que divide la empresa de telecomunicaciones en estratos de responsabilidad de las actividades.

A finales de los años 90 en la mayoría de los mercados de servicios de telecomunicaciones se produjo una evolución desde una situación de monopolio hasta un régimen de competencia. En este entorno, ya no es suficiente reducir los costos de explotación, sino que hay que aumentar los ingresos; para ello es fundamental que los clientes asocien la imagen de calidad de los servicios con el nombre de la operadora. Este hecho ha obligado a que las operadoras de telecomunicaciones modifiquen sus estrategias y revisen sus procesos de negocio.

Ahora, los procesos de negocio, deben poner el hincapié en la atención de los clientes, y esta atención requiere la coordinación de todos los recursos involucrados en la prestación de los servicios. Esta

[3] OSI *(Open Systems Interconnect)*.El modelo de referencia OSI es un modelo de arquitectura de red y un conjunto de protocolos (stack de protocolos) que la implementan. Fue desarrollado por ISO en 1978 como un marco para la construcción de estándares internacionales para arquitecturas de redes de computadoras heterogéneas.

coordinación se debe apoyar en la integración de los sistemas de gestión, de forma que se pueda aumentar el grado de automatización de los procesos de negocio, con el objetivo de reducir los errores y los tiempos de resolución de problemas y de provisión. De esta forma se consigue, en esencia, mejorar la calidad de los servicios y a la vez optimizar los costos.

En esta última línea de trabajo están situados los trabajos del TeleManagement Forum y el consorcio TINA-C. De la misma manera existen experiencias destacables como los proyectos de EURASCOM y RACE (proyectos ACTS) que apuntan en el mismo sentido.

2.2.1.2 Clientes y Servicios son el eje de los procesos de gestión en la actualidad

Actualmente las empresas de telecomunicaciones están afrontando nuevos desafíos. El escenario actual es de altísima competencia (en parte producto de la liberalización progresiva del mercado), donde conservar y conseguir nuevos clientes es crucial para la supervivencia. En este marco, el lanzamiento de nuevos servicios es una forma de conseguir nuevos clientes y posicionarse mejor en el mercado. Para lograr esto con eficacia es esencial el tener el dinamismo y la suficiente flexibilidad para dar respuestas rápidas.

En muchos casos los servicios son mejoras de los existentes o se apoyan en servicios básicos que ya están disponibles. En este sentido la gestión de servicios de estas características impone nuevas necesidades de integración de las distintas plataformas de gestión existentes.

La calidad es otro factor esencial para mejorar la posición en el mercado.

La gestión de la calidad implica hacer el seguimiento de la forma en que los servicios se están prestando a los clientes. Requiere recolección y análisis de datos que provienen de la supervisión de los recursos que soportan dichos servicios.

Hoy día donde la infraestructura de las telecomunicaciones es esencial para muchas empresas, la calidad se torna crítica (por ejemplo, las empresas financieras que procesan transacciones electrónicas pierden mucho dinero al no disponer del servicio de comunicación). Estos clientes exigen garantías sobre la calidad de los servicios contratados. Para ello se realizan acuerdos de Nivel de Servicio Contratado *(Service Level Agreement o SLA)* donde se especifican las características mínimas de calidad que debe cumplir el proveedor de servicios.

Para poder instrumentar este tipo de acuerdos con el cliente es necesario poder controlar los parámetros de calidad del servicio brindado. Hay que procesar la información que proviene de los distintos sistemas de gestión de red. Esto requiere un elevado nivel de integración entre los sistemas de gestión.

2.2.1.3 Contexto de las empresas de Telecomunicaciones

2.2.1.3.1 Análisis de procesos de gestión

Es frecuente la existencia de muchos sistemas de gestión (que por razones históricas se van agregando). En general no existe una integración de todos los sistemas que permita tener una visión global de la red. La integración entre los sistemas es manual y cada operador introduce en su sistema los datos que recibe por teléfono o desde otro sistema mediante algún mecanismo diseñado especialmente. Esta metodología de gestión conduce a errores además de generar demoras en la detección de un problema lo que se traduce en una mala calidad del servicio y un aumento de costos.

Es frecuente (cada vez más) que los servicios brindados por empresas de Telecomunicaciones se apoyen en forma horizontal sobre varios sistemas y procesos existentes. Esto plantea nuevos requerimientos de integración de sistemas y coordinación de procesos de gestión.

2.2.1.3.2 El desarrollo de software

La problemática actual del desarrollo de software esta caracterizada por:

- Requerimientos de automatización y/o soporte de nuevos servicios
- Acceso a información compartida
- Elevada velocidad en los cambios tecnológicos

Surge claramente la necesidad permanente de nuevos desarrollos de software. Es necesario disponer de una metodología de trabajo que permita optimizar los esfuerzos de desarrollo de software, para bajar costos y reducir los tiempos de respuestas. De la misma manera se plantea la necesidad de dar respuesta a la problemática de cambios tecnológicos frecuentes a través de metodologías de desarrollo que permitan una relativa independencia del marco tecnológico vigente.

La tecnología basada en componentes reusables es vista como una ayuda importante en el desarrollo de software en la industria de las telecomunicaciones. Construir sistemas con componentes que interactúan a través de interfaces bien definidas ofrece un camino para el re-uso de software en proyectos de desarrollo de sistemas de telecomunicaciones y la posibilidad de integrar componentes estandarizados de terceros en el sistema.

2.2.1.4 Las plataformas de desarrollo, ejecución de aplicaciones distribuidas y Servidores de Aplicaciones.

Hoy las tecnologías de Información están desarrollando técnicas de computación distribuida basada en el concepto de objetos distribuidos. Esto provee una interfaz de programación que esconde en cierto grado la distribución y facilita el desarrollo. Algunas de las experiencias mas conocidas son: CORBA *(Common Object Request Broker Architecture)*[49] impulsado por OMG *(Object Management Group)*, Java RMI *(Java TM Remote Method Invocation)* de Sun Microsystems y DCOM *(Distributed Common Object Model)*[25] de Microsoft, luego llamado COM+^{TM} y cuyo sucesor actual es .NET. Es de destacar la aparición reciente en escena del modelo WebServices apoyado en XML.

Los Servidores de Aplicaciones son sistemas informáticos de base que proveen infraestructura para la ejecución, y habitualmente para el desarrollo, de aplicaciones o componentes en el contexto de arquitecturas cliente-servidor de múltiples capas.

A través de uso de Servidores de Aplicaciones es posible ejecutar componentes de software que se invocan mutuamente, aún residiendo en diferentes máquinas. Esta posibilidad es la conocida como de soporte a componentes u objetos distribuidos.

Los Servidores de Aplicaciones se consideran generalmente como middleware de tipo ORB *(Object Request Broker)*, debido a que permiten la conexión entre aplicaciones y son por tanto soluciones tecnológicas de alto nivel para la comunicación de software. Más concretamente, los principales productos en la gama Servidor de Aplicación implementan componentes y mecanismos para la interacción entre los mismos y con aplicaciones externas.

Sin embargo, la tendencia actual es a que los Servidores de Aplicaciones incluyan también servicios y mecanismos no sólo de tipo ORB, sino también de tipo MOM *(Message Oriented Middleware)*, gestión de transacciones, integración débilmente acoplada basada en XML, etc. Asimismo, los actuales productos de Servidor de Aplicaciones incluyen ambientes para desarrollo y conexión con Servidores Web.

En resumen, los Servidores de Aplicaciones son un tipo de productos en plena expansión tecnológica e industrial que tienden a cubrir el conjunto de requerimientos generados en el desarrollo de sistemas en múltiples capas.

En la actualidad existen tres grandes familias de productos tipo Servidor de Aplicaciones: CORBA, J2EE, y Productos de Microsoft (COM+, .NET).

Más allá de la disponibilidad de estas nuevas tecnologías de distribución de objetos existe la necesidad de contar con una arquitectura común de las aplicaciones, un conjunto de requerimientos que permitan soportar operabilidad, interoperabilidad y puesta en funcionamiento de módulos (componentes) con interfaces apropiadas. Esto permitiría a un desarrollador construir paquetes de una manera consistente pudiendo interoperar con paquetes similares en un entorno de ejecución bien definido.

2.2.2 TMN (ITU-T)

2.2.2.1 Introducción

El TMN *(Telecommunications Management Network)* establece un marco para lograr la interconexión de sistemas de operaciones heterogéneos y redes de telecomunicaciones. Fue desarrollado por ITU-T *(International Telecommunications Union)* como una infraestructura para soportar la gestión de servicios dinámicos en el área de las telecomunicaciones.

La rápida evolución de la industria de las telecomunicaciones sumado a las tecnologías emergentes, el proceso de desregulación y la creciente demanda de servicios por parte de los usuarios, determinaron la necesidad de un modelo de gestión de las redes de telecomunicaciones.

Este modelo de gestión tiene que resolver la gestión integrada de diversidad de equipos de diferentes fabricantes y con múltiples protocolos de gestión. Asimismo debe permitir la interacción entre diferentes empresas proveedoras de servicios. En este sentido, el marco TMN, proporciona una arquitectura de referencia para el intercambio de información de gestión entre los sistemas de operación y/o los equipos.

2.2.2.2 Marco de trabajo propuesto por TMN

Los conceptos de TMN *(Telecommunications Management Network)* se definen a partir de la Recomendación M3010 de ITU-T [11].

El TMN intenta promover un framework de trabajo que sea flexible, escalable, confiable. Intenta definir formas estandarizadas para la realización de tareas vinculadas a la gestión de la red y comunicaciones a lo largo de distintas redes. TMN permite que el procesamiento se realice en forma distribuida, utilizando niveles razonables de escalabilidad, performance y eficiencia en la comunicación.

Una red de telecomunicaciones esta compuesta de sistemas de conmutación, equipos de transmisión, terminales, etc. En la terminología de TMN, se utiliza el termino de elementos de red o NE *(Network Element)* para referirse a los recursos de la red en general[9].

Desde el punto de vista de la gestión, los sistemas de conmutación, transmisión y otros recursos de la red se conectan a través de una red de comunicaciones de datos o DCN *(Data Communication Network)* a uno o más sistemas de operaciones (OSS). Los sistemas de operaciones realizan la mayoría de las funciones de gestión. Estas funciones se pueden llevar a cabo por operadores o en forma automática. Es posible que una operación de gestión termine llevándose a cabo por múltiples sistemas de operaciones. En este sentido la red de comunicaciones de datos (DCN) se usa para intercambiar información de gestión entre sistemas de operaciones. La DCN, se usa también para conectar estaciones de trabajo *(Workstations o WS)* las cuales permiten interpretar la información de gestión. Estas estaciones tienen interfaces humanas las cuales escapan el alcance de TMN (en este sentido las WS están en el borde de TMN como se puede ver en la Figura 1) [24].

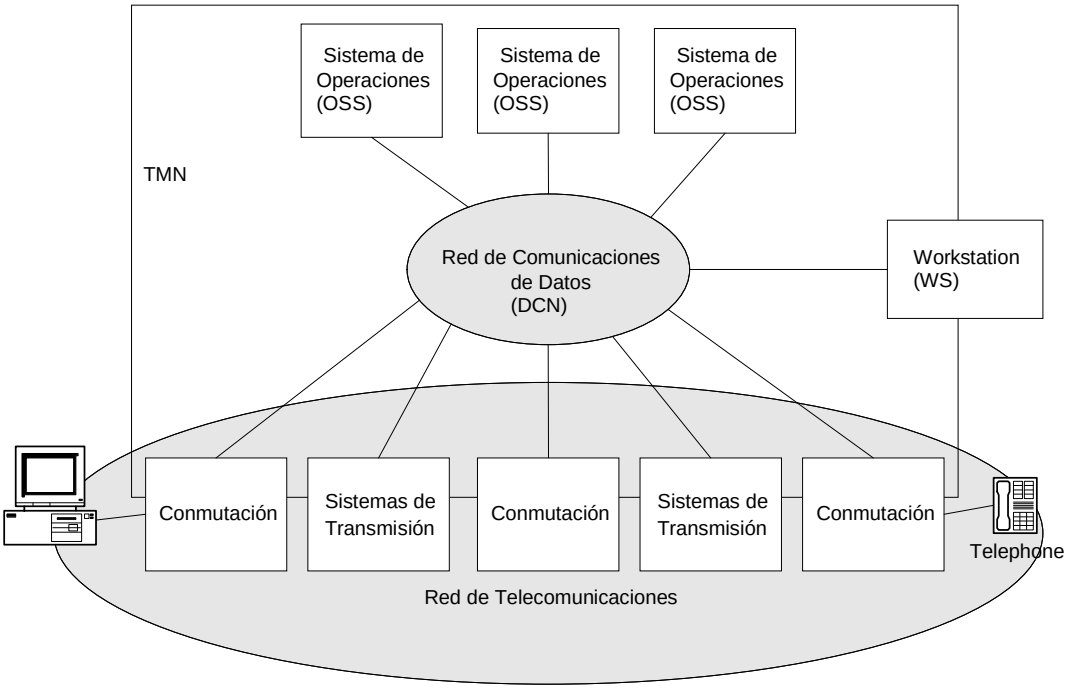


Figura 1 – Relación entre TMN y red de telecomunicaciones.

Es posible encarar la arquitectura desde distintos puntos de vista:

La arquitectura funcional (punto de vista lógico), que proporciona la visión de funciones generales de TMN, distinguiendo entre sus distintos bloques funcionales.

La arquitectura de información (punto de vista de la gestión funcional), que orienta al uso del paradigma orientado a objetos para el intercambio de información utilizando los conceptos de gestor-agente y sus relaciones: operación vs. notificación. Desarrolla las 5 áreas de funciones de gestión FCAPS *(Fault, Configuration, Accounting, Performance y Security)* que se describen en 2.2.2.2.2.4.

La arquitectura física (punto de vista de comunicaciones entre componentes), que facilita la comunicación de gestión entre los equipos de la red de telecomunicaciones y desarrolla el concepto de "interfaces" a partir de los puntos de referencia entre funciones lógicas.

La arquitectura estratificada lógica divide la gestión de las telecomunicaciones en capas lógicas mostrando cómo la gestión puede ser estructurada de acuerdo a diferentes responsabilidades.

2.2.2.2.1 Arquitectura funcional (punto de vista lógico)

La norma M3010 de ITU-T define la arquitectura funcional TMN definiendo sus bloques de funciones lógicas (ver Figura 2). No todas las funciones tienen que estar presentes en un sistema TMN. Como se puede observar en la figura, algunas de las funciones no están dibujadas completamente dentro del recuadro etiquetado como TMN, esto significa que no están totalmente especificadas por TMN.

Los bloques de funciones son:

Bloques de funciones del sistema de operaciones *(Operation Systems Functions, OSF).* Responsables de la gestión del elemento de red. Estos bloques funcionales inician operaciones de gestión sobre elementos de red específicos y reciben notificaciones de los mismos. En términos del modelo gestor-agente, estos bloques funcionales desempeñan el papel de "gestor".

Es de destacar que dentro del TMN pueden existir múltiples bloques funcionales de operaciones. Estos también pueden dialogar entre sí, siguiendo un esquema similar al modelo gestor-agente.

Bloques de funciones de un elemento de red *(Network Elements Functions, NEF).* Una red típica de telecomunicaciones está equipada con sistemas de conmutación y sistemas de transmisión. Estos sistemas son ejemplos de "elementos de red" o NE *(Network Element)*. Los NE llevan a cabo dos tipos de funciones (NEF):

- Funciones Primarias (de gestión de las telecomunicaciones): son las funciones que permiten el intercambio de datos de los usuarios finales de la red de telecomunicaciones.
- Funciones de Gestión: que permite que el elemento de red (NE) actúe en calidad de agente. Esto es, puede trabajar en una relación gestor-agente con sistemas de operaciones.

Es de destacar que las funciones primarias no son especificadas por TMN (por esta razón el bloque funcional NEF en la Figura 2, no está totalmente contenido en el recuadro TMN).

Bloques de funciones de adaptación Q *(Q-Adaptation Functions,QAF).* Brinda la funcionalidad de enlace *(gateway)* a los efectos de poder gestionar elementos de red no TMN, o heredados de tecnologías más antiguas,

Bloques de funciones de estación de trabajo *(Workstations Functions, WSF).* Funciones encargadas de presentar al humano la información de entidades TMN.

Bloques de funciones de mediación *(Mediation Functions, MF).* Actúan en el pasaje de información entre OSF y NEF, almacenando, adaptando, filtrando, estableciendo umbrales, condensando la información.

Bloques de funciones de comunicaciones de datos *(Data Communications Functions, DCF).* No son estrictamente bloques funcionales. Posibilitan la comunicación entre los distintos componentes. DCF provee las capas 1 a 3 del modelo de referencia de OSI (OSI-RM)[4].

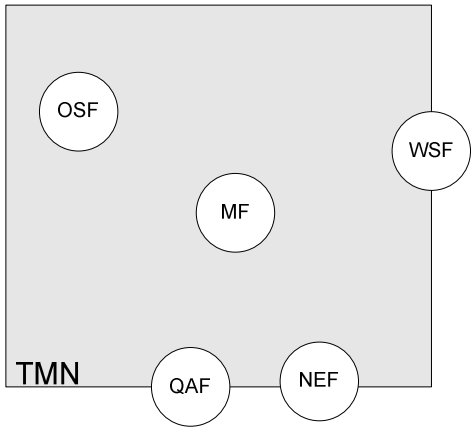


Figura 2 - Bloques de funciones lógicas TMN

[4] OSI-RM *(Open Systems Interconnect-Reference Model)*.El modelo de referencia OSI es un modelo de arquitectura de red y un conjunto de protocolos (stack de protocolos) que la implementan.

2.2.2.2.2 Arquitectura de información (punto de vista de la gestión funcional)

2.2.2.2.2.1 Visión de "Recursos" como objetos gestionados

La arquitectura de información de TMN utiliza un enfoque orientado a objetos y está basado en el modelo de Información de Gestión de OSI. En este modelo cada recurso es representado por un objeto gestionado MO *(Managed Object)*[24].

Desde el punto de vista de la gestión de los recursos, solo es visible la frontera del objeto gestionado. Esta frontera (visión de Gestión del Objeto) está caracterizada por:

- Atributos. Propiedades o características del objeto gestionado (MO)
- Operaciones. Que se pueden realizar sobre el objeto (operaciones de gestión sobre el MO)
- Comportamiento. Que se exhibe como respuesta a las operaciones efectuadas
- Notificaciones. Emitidas por los objetos

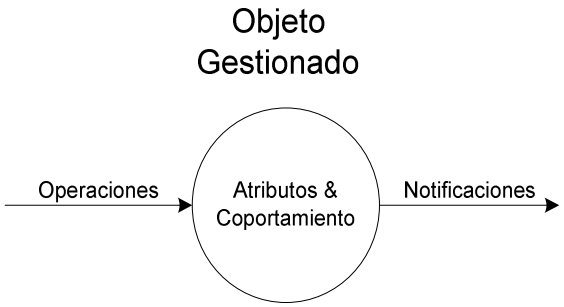


Figura 3 – Objeto Gestionado(Managed Object)

La comunicación se lleva a cabo utilizando el paradigma Gestor-Agente, donde:

- el gestor: pide operaciones al agente y solicita se le notifique del resultado, o bien recobra o establece un atributo
- el agente: notifica de los resultados de operaciones al gestor (a solicitud de éste), o informa espontáneamente los eventos que ocurren.

Para que exista interoperabilidad entre el Gestor y Agente debe existir un lenguaje común y estructuras de información estandarizadas.

2.2.2.2.2.2 La base de datos de información de gestión

La información de gestión reside en la base de información de gestión *(Management Information Base, MIB)* . En ella se coleccionan las clases correspondientes a los objetos gestionados (MO). La definición de clase de objeto incluye:

- una lista de atributos,
- operaciones,
- acciones,
- una descripción de sus relaciones con otras clases de objetos, y
- eventos asociados con el elemento de red gestionado.

Por cada recurso de red que deba ser gestionado o monitoreado, se instancia en la base de gestión (MIB) un objeto de la clase requerida. Cada elemento de red que se agrega debe ser instanciado. En este sentido existe una relación uno a uno entre el recurso de red físico y el objeto de gestión instanciado.

Los MO son una abstracción de los recursos reales gestionados que hay en la red, ejemplo de estos son:

- alarmas que afectan el estado de los objetos.
- una descripción de los atributos que afectan la función que se le determinó a cumplir.
- acciones necesarias para efectuar una cross-conexión en el elemento de red.

Cada vez que se produce un cambio en la red, la MIB cambiará reflejando el nuevo estado de la red.
Cuando un operador modifica un parámetro operacional de la red, también producirá un cambio en el atributo del MO que representa el recurso de la red en cuestión. Estos cambios pueden arrastrar otros y así sucesivamente.

2.2.2.2.2.3 Mecanismos de Interoperabilidad Gestor-Agente

Los objetos gestionados responden al paradigma orientado a objetos y la definición de los mismos se lleva a cabo utilizando las pautas para definir los mismos de OSI *(Guidelines for the Definition of Managed Objects, GDMO)* y el lenguaje de especificación ASN.1[5]*(Abstract Syntax Notation 1)*.

Los gestores y agentes se comunican a través del protocolo CMIP *(Common Management Information Protocol)* utilizando elementos CMIS *(Common Management Information Services)* (que son protocolos y modelos de información de OSI recogidos en X.711 de ITU).

Los servicios de información CMIS son utilizados por parejas de procesos para intercambiar información y ejecutar comandos a los efectos de llevar a cabo la gestión de la red. CMIS define un conjunto de mensajes (GET, CANCEL-GET, SET, CREATE, DELETE, EVENT-REPORT y ACTION), y la estructura y contenido de los mismos de manera que puedan ser utilizado en el contexto de sistemas abiertos. Conceptualmente es similar a SNMP, pero más potente (y por lo tanto más complejo).

En la jerga a ésta situación se la conoce habitualmente como el protocolo OSI, TMN CMIP/GDMO, en contraposición con el concepto de gestión de TCP/IP, SNMP, de Internet. El conjunto de GDMO y ASN.1 actuan en contraposición de la Estructura de la Información de Gestión (SMI) de Internet.

2.2.2.2.2.4 Plano de funciones de aplicación de gestión (FCAPS)

Las siguientes funciones de aplicación de Gestión son desarrolladas en la norma ITU-T M3400 y constituyen otra visión al concepto de gestión de servicios, pues si bien tratan de la gestión de la red se perfilan hacia los servicios por ella soportados.

- Función de gestión de fallo (F)
  Procesos para aislar, detectar y corregir errores que causan fallos de los sistemas de comunicaciones.
- Función de gestión de configuración (C)
  Procesos para preparar, inicializar, y operar recursos de red con el propósito de gestión de la capacidad, o la provisión /re-provisión de servicios.
- Función de gestión de costeo (A)
  Procesos que habilitan cargos a ser establecidos por el uso de recursos de red.
- Función de gestión de desempeño (P)
  Procesos para mantener el desempeño de la red al nivel de conseguir requerimientos de servicios que lleven a costos efectivos.
- Función de Gestión de seguridad (S)
  Procesos que aseguran acceso a recursos que son permitidos sólo por usuarios autorizados. Y a los niveles asignados a ellos.

Cada una de estas funciones dará lugar luego a una serie de actividades y procedimientos operacionales que deberán tener lugar en el seno de la empresa. Aquellas podrán estar organizadas según distintas formas pero todas ellas deberán tener un denominador común, esto es: una visión de cara a las líneas de servicios que la empresa provea [12].

---

[5] ASN.1 es un standard de ISO/ITU-T. Define la sintaxis (en términos abstractos) de la información sin restringir la forma en que se codifica. Adicionalmente existen reglas de codificación ASN.1 *(transfer syntax)* , que posibilitan la representación concreta de valores cuya sintaxis abstracta se describe en ASN.1. Estos estándares facilitan el intercambio de datos e información estructurada especialmente entre aplicaciones que operan sobre redes, mediante la descripción de estructuras de datos formuladas independientemente de arquitecturas de máquinas concretas y lenguajes de implementación específicos.

2.2.2.2.3 Arquitectura Física (punto de vista de comunicaciones entre componentes).

La Arquitectura Física muestra como se implementan cada una de las funciones TMN (definidas en la arquitectura funcional) en términos de equipos físicos (*Building Blocks*) con sus respectivas interfases.

La Figura 4 muestra de manera simplificada la arquitectura física o plano de comunicaciones de TMN, según ITU-T. Allí se distinguen, los *Building Blocks* y las interfaces.

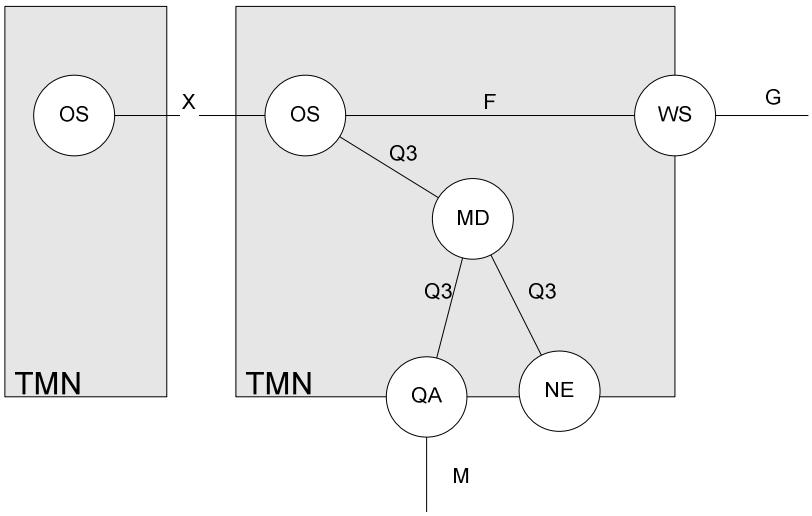


Figura 4 - Arquitectura física de TMN y sus interfaces

El sistema de operaciones *(Operations System, OS)* representa los sistemas de supervisión y control del TMN. Los OS pueden generar otras estructuras de gestión jerarquizadas en capas, interconectándose a través de la interfaz X.

El elemento de red *(Network Element, NE)* es el único nodo que reside en la red gestionada (la propia red de telecomunicaciones). Su trabajo primario es manejar tráfico, carga útil y no gestión. Por ello es origen o destino de la supervisión de gestión o control.

La estación de trabajo *(Workstation, WS)* es donde el humano se coloca. Ella provee la función de presentación. Los nodos se intercomunican a través de la Red de Comunicación de Datos (DCN) (red de soporte operacional, en su parte enfocada a los elementos de red) que es el medio usado en el mundo TMN.

Los bloques de mediación *(Mediation Devices, MD)* actúan en el pasaje de información interna, almacenando, adaptando, filtrando, estableciendo umbrales, condensando la información, etc.

Finalmente los bloques de adaptación-Q *(Q-Adapters, QA)* ofician de enlace con elementos de red legados (que no responden al TMN).

En cuanto a las interfaces Q3, conecta un OS con un NE, con un QA, con un MD, y aún 2 OS si pertenecen a un mismo TMN.

Finalmente la interfaz F permite la interconexión de WS con los otros nodos. La metodología para especificar las interfaces está dada en la norma M3020. La norma M3100 por su parte contiene los MO que son genéricos para describir la información intercambiada entre las interfaces TMN [6].

2.2.2.2.4 Arquitectura estratificada lógica

Adicionalmente al descomponer los bloques funcionales en actividades, se puede separar la información gestionada por los bloques de Funciones del Sistema de Operaciones (OSF) en niveles de abstracción que a veces se les llama capas, o estratos. Estos son:

- **Nivel de información de Gestión de Negocios BML** *(Business Management Layer)*
- **Nivel de información de Gestión de Servicios SML** *(Service Management Layer)*
- **Nivel de información de Gestión de Red NML** *(Network Management Layer)*
- **Nivel de información de Gestión de Elemento EML** *(Element Management Layer)*
- **Nivel de información del Elemento EL** *(Element Layer)*

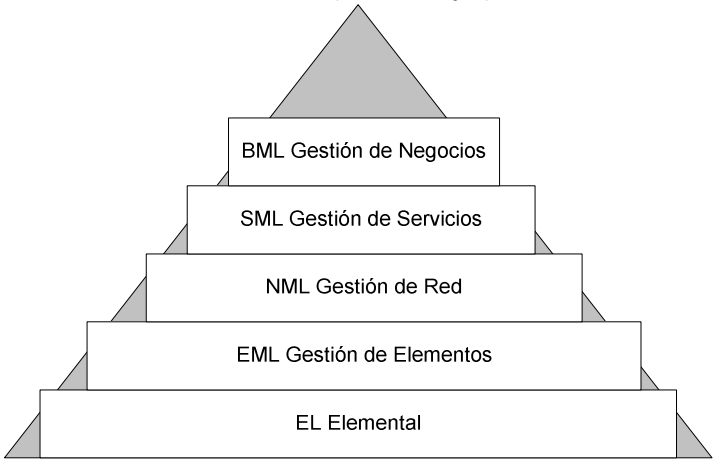


Figura 5 - Niveles de Abstracción de la gestión

Se construye así la pirámide que separa la visión de una empresa en niveles (Figura 5). Esta pirámide es la primera visión concreta que ITU-T dispuso de una separación en actividades de una empresa de telecomunicaciones, donde se perfila al menos al cliente a través de la capa de servicios, y al dueño de la empresa a través de la visión de los Negocios.

2.2.2.3 TMN vs. Gestión en Internet

Una diferencia importante entre TMN y la gestión de redes a nivel de Internet es que el esquema TMN se concentra en la especificación de arquitecturas de gestión mientras que en el caso de Internet, el acento está puesto en la implementación de protocolos de gestión. Como resultado, existen solamente un número muy limitado de productos TMN en el mercado, mientras que existen muchas alternativas (comerciales y de dominio público) para la gestión en Internet. Concretamente en la comunidad de Internet se hizo una apuesta muy fuerte por el protocolo de gestion SNMPv3 el cual responde a una arquitectura muy simple si se compara con la riqueza de conceptos arquitectónicos definidos por TMN[24].

La integración entre TMN y SNMP siempre ha sido un punto de investigación importante. La integración se obtiene por el uso de Adaptadores-Q (Función QAF). El QAF se intercala, dialogando CMIP hacia un extremo, y SNMP hacia el otro lado. La parte más crítica, constituye la traducción del modelo TMN que utiliza la propuesta OSI para la definición de objetos gestionados *(Guidelines for de Definition of Manager Objectso GDMO)* y la estructura de información utilizada en Internet SMI *(Structure of Management Information).*

A diferencia de lo que sucede con la gestión en Internet, la especificación TMN sugiere una separación conceptual entre la red que se gestiona (la red de telecomunicaciones) y la red que traslada la información

de gestión *(Data Communications Network, DCN)*. Los miembros del grupo de gestión de Internet tomaron otra dirección: prefirieron usar los mismos componentes para la red siendo gestionada y la red utilizada para el transporte de información de gestión.

Probablemente uno de los conceptos más valiosos de TMN es el concepto de arquitectura estructurada en capas lógicas *(Logical Layered Architecture,LLA)*. Esta arquitectura distingue entre elementos de la red, la propia red, los servicios y la gestión del negocio. Tradicionalmente en Internet se puso énfasis en la gestión de los elementos y la propia red, actualmente se están llevando a cabo experiencias e investigaciones en el área de gestión de servicios sobre Internet a cargo de grupos que trabajan en el contexto de IRTF *(Internet Research Task Force)*.

2.2.3 TM Forum

2.2.3.1 Introducción

TMForum (*TeleManagement Forum*) es un consorcio global de proveedores de servicios y fabricantes que apunta a lograr que la industria de las telecomunicaciones provea mecanismos efectivos para mejorar las redes públicas y la gestión de los servicios asociados.

El *TMForum* [43] tiene la siguiente misión:

- Facilitar a los SP *(Service Providers)* u Operadores de Red moverse a bajo costo, hacia operaciones de negocios con mayor grado de automatización.
- Facilitar la capacidad de gestionar servicios al cliente a través de cadenas de suministros complejas.
- Facilitar la amplia disponibilidad de sistemas de soporte operacional a bajo costo del tipo de componentes comercialmente disponibles o COTS (*Commercial Off The Shelf*).

El Foro suministra a la industria los siguientes servicios y facilidades:

- una rápida, más resistente y barata vía de gestionar redes/servicios y de obtener el software para lograr tales objetivos a través de una colaboración estructurada (entre los interesados),
- bajo riesgo y alto retorno en inversiones en desarrollos de OSS *(Operational Support Systems)*,
- un foro para acuerdos,
- un método para influir en la industria,
- un mercado para vendedores y compradores,
- compartir conocimiento de la industria.

2.2.3.2 Otra visión de TMN

El TMForum ha tomado una visión de arriba hacia abajo *(Top-Down)* de la propuesta TMN. Usa al modelo como una manera de dirigir el problema del negocio mirando desde el punto de vista de los operadores de telecomunicaciones y los SP *(Service Providers)*. En este sentido, el negocio es el centro del análisis.

El TMForum ha adoptado el término TMN Inteligente *(SMART TMN)* para capturar el valor potencial de TMN en la industria de las telecomunicaciones. *SMART TMN* defiende una proximidad pragmática a las normas, con un énfasis en la automatización operacional "extremo a extremo" *(end to end)*, por encima de soluciones puntuales eventuales y la aplicación de tecnologías prácticas que sean económicas, estén disponibles para su aplicación y resuelvan el problema.

El valor del TMForum para la industria está en ayudar a los SP de comunicaciones y proveedores a tomar decisiones adecuadas con respecto a la implantación de sistemas usados para manejar su negocio. Ayuda a identificar cadenas de valor dentro de la empresa, analizando los procesos involucrados e identificando actividades donde se genera (o no) un valor agregado.

2.2.3.3 Áreas de trabajo del TMForum

Uno de los objetivos primarios de TMForum es tratar de crear un escenario donde sea posible el uso de tecnologías de componentes COTS para desarrollar el modelo TMN. El mismo consiste básicamente en disponer de elementos en el mercado de software y hardware que sean capaces de interoperar, a fin de poder ser incorporados a voluntad y puedan funcionar en una empresa de telecomunicaciones (cualquiera).

A estos efectos se estableció un marco de trabajo para articular soluciones que contempla los siguientes aspectos:

Procesos del negocio. Se modelan las interacciones de los procesos del negocio de las telecomunicaciones y los modelos de información asociados. Esta especificación es conocida como TOM *(Telecom Operations Map)* [44] que actualmente se ha sustituido por una versión más moderna denominada e-TOM [38] (ver 2.2.3.4).

Modelos de Información. Se modelan los "conceptos" manejados en el negocio.

Marco de Reglas para la construcción de Sistemas. Las interacciones de procesos de negocios son modeladas en términos de componentes de aplicaciones. Se especifican interfaces apoyadas en los modelos de información propuestos. Se apuesta a una formulación independiente de la tecnología en primera instancia.

Un aspecto importante de este marco de referencia será la especificación de criterios de diseño que permitan luego desarrollar aplicaciones re-usables. Estos componentes deberán ser desarrollados a través de múltiples escenarios de telecomunicaciones, donde habrá influencias de muchos factores sobre la colección de criterios de diseños. Así se determinará la dirección de la aplicación correcta, necesaria para soportar las interacciones de los procesos de negocios de capas superiores.

Proyectos Catalizadores y COTS. Los proyectos catalizadores son las instancias del TMForum donde se "prueban" Modelos de Información, Procesos e Interfaces Estandarizadas. Los componentes que han sido probados exitosamente pasan a ser ofrecidos como COTS por ISV *(Independent Software Vendors)*.

La construcción de soluciones surge de la sinergia de interacción entre las cuatro áreas señaladas anteriormente (ver Figura 6.).

La lógica de los procesos se expresa en función de un modelo de información. Se definen posteriormente servicios brindados por componentes que llevan a cabo los procesos requeridos. Estas interfaces de servicios son definidas inicialmente en términos neutrales apoyados sobre el modelo de información. Finalmente, en función de interfaces específicas es posible la construcción de componentes COTS.

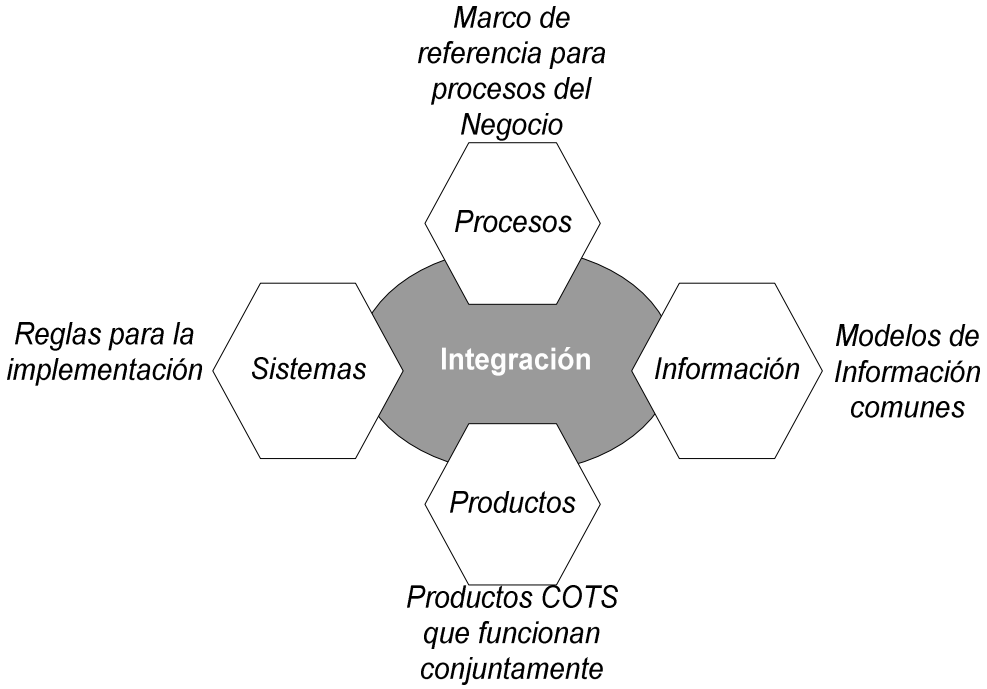


Figura 6 - Integración TMForum

2.2.3.4 Mapa de Procesos en las Empresas de Telecomunicaciones: TOM *(Telecom Operations Map)*

Describe los procesos que se llevan a cabo en una empresa de telecomunicaciones.

- Identifica la cadena de valor de gestión, parte desde los clientes, pasando por la gestión de servicios, por la gestión de red, incluyendo la fuente de los activos de red y los servicios propios de comunicación y de información .
- El modelo se conduce de una manera independiente de la tecnología, resultado con ello que las definiciones de los procesos de negocios tendrán más larga vida que la arquitectura de la tecnología de la información y la tecnología que soporta el modelo.
- El modelo, por fuera de la empresa considerada, es usado como punto de referencia para modelar los procesos de negocios reflejando la cooperación entre los SP *(Service Providers)* a efectos de cumplir con los requerimientos de los clientes.

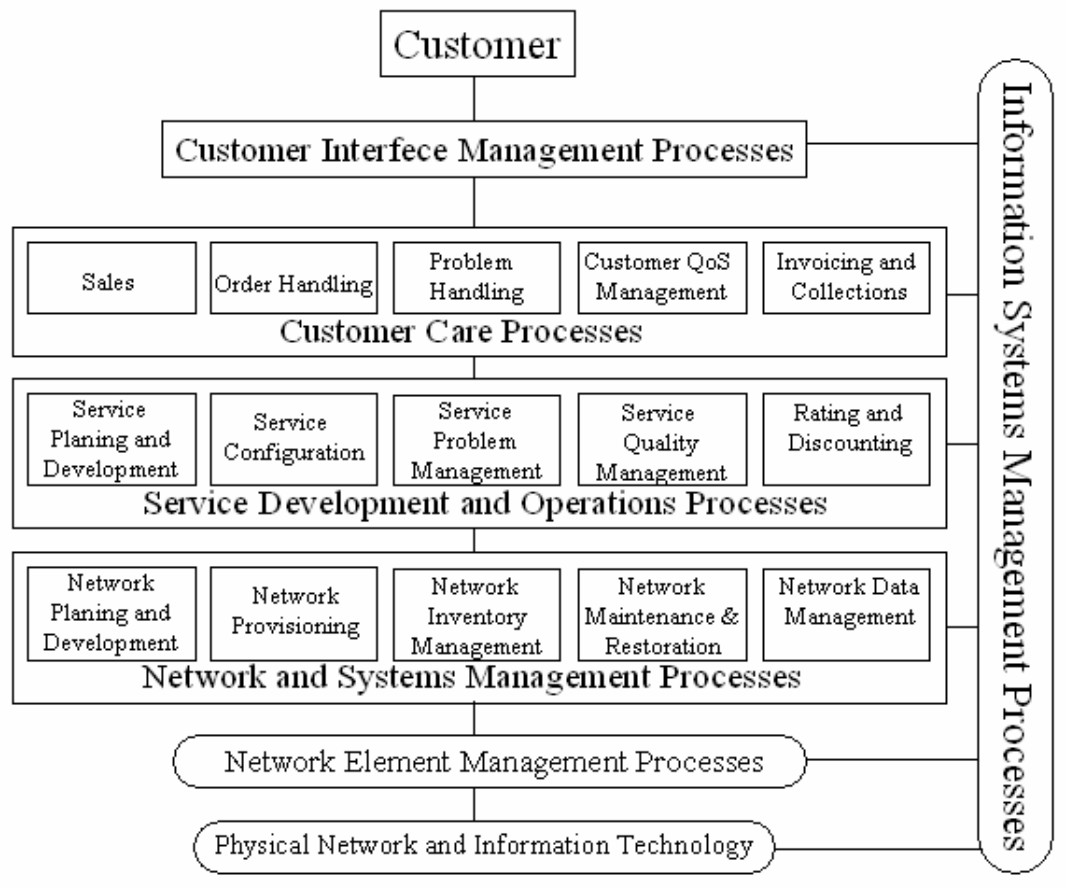


Figura 7 - Marco de trabajo TOM, Proceso del negocio (ref. [44])

En la Figura 7 se muestra una síntesis del Mapa de Procesos. Se puede observar que el mismo está dispuesto de manera que todos los procesos vinculados al "cuidado" de los clientes están situados en la parte superior *(Customer Care Processes)*. Estos procesos a su vez interactúan con procesos vinculados al área de desarrollo y operación de servicios *(Service Development and Operation Processes)*. Finalmente, estos procesos interactúan con aquellos definidos a nivel del área de gestión de la propia red y los sistemas de telecomunicaciones *(Network and Systems Management Processes)*. Estos últimos son los que realmente interactúan con los sistemas y dispositivos de la red (dibujados en la parte inferior del diagrama).

Otra visión presente en el diagrama, es la visión de procesos “extremo a extremo” *("end to end")* (tomando en consideración otros SP u operadores y los puntos de interconexión). Esta visión complementaria, se expresa en una dinámica de procesos cuyas interacciones actúan de izquierda a derecha en el diagrama (ver Figura 8).

Algunas características del mismo son:

- Desde el punto de vista de un cliente final, sus interacciones con un SP se pueden desarrollar en las siguientes áreas: provisión (*FullFillment*), aseguramiento (*Assurance*), y facturación (*Billing*) para el servicio. Esta parte del modelo es conocida como modelo FAB *(acrónimo de Fullfillment, Assurance, Billing)* , y es una visión de procesos *"end to end"*, donde se incluyen otros *SP*,

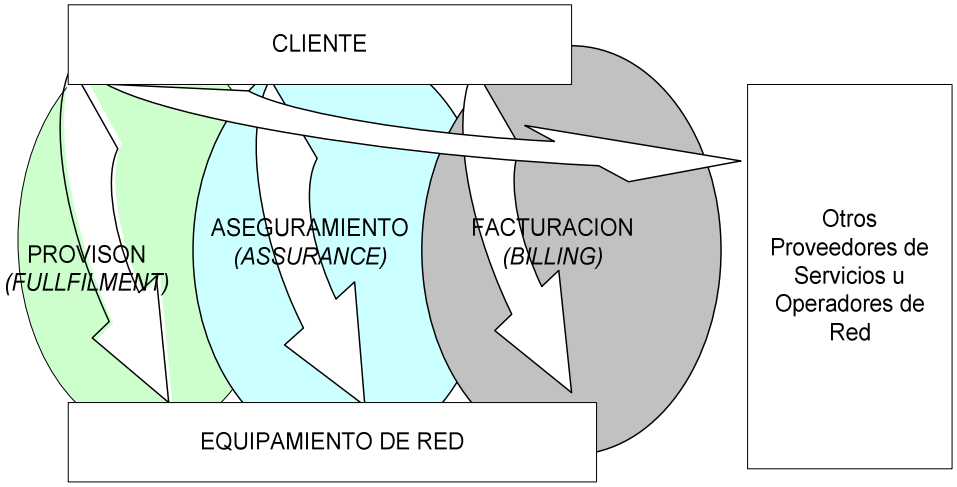


Figura 8 - Flujo del Proceso FAB *"end to end"* y el flujo a través

- El modelo hace reflejar fuertemente el punto de vista del cliente y se focaliza en su gestión, y en los procesos de flujo de información desde el cliente a la red y a los sistemas de operación.
- La visión del TOM a través del FAB también permite ayudar a mostrar como una actividad distribuida puede separarse en sub-procesos a través de dominios. Un caso es el del Acuerdo de Nivel de Servicio o SLA (*Service Level Agreement*) donde Cliente y Proveedor se ponen de acuerdo en cuanto las características del servicio contratado, fundamentalmente la calidad del mismo.
- Los cambios de perspectiva que produce son vitales para que la empresa considerada sobreviva y pueda abordar instancias como la Re-ingeniería de Procesos de Negocios o BPR (*Business Proceses Re-engineering*)**,** estableciéndose luego pautas para efectuarlo, así como métodos para transformar las empresas de acuerdo a los conceptos del TMForum.

2.2.4 Conclusiones

A medida que las redes y servicios fueron creciendo en complejidad fue necesario recurrir a la automatización de distintos procesos, estructurándose de esta manera Sistemas de Gestión Automatizados.

ITU-T formalizó el modelo TMN *(Telecommunications Management Network)*[11] como un modelo para estructurar en forma lógica las actividades del negocio. Este modelo inicialmente proporciona una arquitectura de referencia para intercambio de información de gestión entre los sistemas de operación y equipos. Las normas asociadas al modelo TMN están basadas en el esquema OSI de gestión de red, donde se definen servicios básicos en capas inferiores que se brindan a capas superiores de mayor nivel de abstracción.

Este enfoque (TMN) se ha visto enfrentado con formas diferentes de plantearse el transporte y el encaminamiento de la carga útil de información (TCP/IP), que a su vez han definido sus propias

estrategias para la gestión de redes (SNMP). En este sentido no existe aún una plataforma de gestión genérica que pueda gestionar la totalidad de las redes por lo que continua la búsqueda de una plataforma de gestión común que sea bastante independiente de la tecnología de red a la cual se gestiona y y lo suficientemente genérica como para poder dar soporte a los nuevos servicios que el mercado competitivo exija[5].

Es de destacar que el modelo TMN se ha constituido en una referencia ineludible para otros modelos y sistemas de gestión. No obstante, las características del enfoque de TMN (de resolver detalles de bajo nivel para subir en nivel de abstracción posteriormente), puede producir estructuras de gestión verticales orientadas a un servicio o ciertas funciones. En este tipo de estructuras la comunicación entre distintos sistemas de gestión es escasa. Este modelo no es viable cuando los servicios se multiplican y se apoyan sobre otros servicios más sencillos preexistentes, requiriéndose un intercambio ágil de información entre los sistemas involucrados.

Actualmente es más interesante abordar el problema de arriba hacia abajo, viendo los procesos que es necesario realizar y como debe fluir la información entre los distintos procesos y actividades.

Este enfoque (que parte desde arriba) permite abordar el problema desde una perspectiva independiente de la tecnología. Inicialmente se manejan aspectos técnicos muy básicos, con información vinculada al cliente. A medida que se baja en los niveles TMN, se incorporan los datos necesarios.

Para poder lograr lo anterior es esencial disponer de un lenguaje común y un modelo de datos para articular la comunicación entre sistemas. Esto simplifica el desarrollo de interfaces e interconexión. En este sentido el TMForum ha sintetizado (a través del conocimiento de distintas empresas de telecomunicaciones) un modelo que describe los procesos que se llevan a cabo en una empresa de telecomunicaciones típica *(Telecomunications Operations Map o TOM)* que ofrece una visión de alto nivel de los procesos de gestión, abarcando actividades que cubren los aspectos de atención al cliente, gestión de servicio y gestión de red. Este punto de partida posibilita la construcción de Sistemas de Operaciones basados en Building Blocks en correspondencia con los distintos procesos que se llevan a cabo en la empresa.

2.3 Sistemas de Gestión de Redes y Servicios basados en Building Blocks

2.3.1 Introducción

2.3.2 Building Blocks: Requerimientos generales para la gestión de telecomunicaciones (recomendación GB909[44])

2.3.2.1 Introducción
2.3.2.2 Características de los Building Blocks
2.3.2.3 Arquitectura de Aplicación
2.3.2.3.1 Objetivos de la Arquitectura
2.3.2.3.2 Requerimientos
2.3.2.3.2.1 Requerimientos Básicos
2.3.2.3.2.2 Trading Service
2.3.2.3.2.3 Independencia de Versión
2.3.2.3.2.4 Resistencia a Fallos
2.3.2.3.2.5 Requerimientos transaccionales
2.3.2.3.2.6 Seguridad
2.3.2.3.2.7 Common Application Management Interface
2.3.2.4 Construcción de Sistemas Lógicos basados en BB
2.3.2.5 Separación de ámbitos de los BB
2.3.2.5.1 Process Automation Tier
2.3.2.5.2 Human Interaction Tier
2.3.2.5.3 Enterprise Information Tier
2.3.2.5.3.1 Modelos de Información

2.3.3 Arquitectura NGOSS

2.3.3.1 Metas y Objetivos
2.3.3.2 Principios
2.3.3.2.1 Separación de los procesos del Negocio de la implementación de componentes
2.3.3.2.2 Interfaces definidas a través de "Contratos"
2.3.3.2.3 Gestión basado en políticas
2.3.3.2.4 Roles de Usuarios y Seguridad
2.3.3.2.5 Vehículo de Comunicaciones
2.3.3.2.6 Información compartida
2.3.3.2.7 Adaptación de Información
2.3.3.2.8 Sistemas distribuidos escalables débilmente acoplados
2.3.3.2.9 Framework Services
2.3.3.2.10 Separación de la "Arquitecturas Neutral" y la "Arquitectura dependiente de la tecnología"
2.3.3.3 Descripción de la Arquitectura
2.3.3.3.1 Vista Estructural de la arquitectura
2.3.3.3.2 Modelos de Información
2.3.3.3.3 Especificaciones de Contratos
2.3.3.3.4 Uso de contratos
2.3.3.3.4.1 Ubicación de Servicios (Registry Service).
2.3.3.3.4.2 Servicio de Invocación (Invocation Service)
2.3.3.3.5 Gestión de procesos
2.3.3.4 Gestión de Información Compartida

2.3.4 Conclusiones

2.3.1 Introducción

En la sección 2.3.2 se presentan requerimientos generales para la construcción de sistemas basados en Building Blocks. Se presentan aspectos de la Arquitecturas requeridas para posibilitar la construcción y operación de sistemas de estas características.

En la siguiente sección (2.3.3) se presenta la arquitectura NGOSS *(New Generation of Operation and Support Systems)* propuesta por TMForum, que sigue las pautas señaladas en la sección 2.3.2.

2.3.2 Building Blocks: Requerimientos generales para la gestión de telecomunicaciones (recomendación GB909[44])

2.3.2.1 Introducción

Dado que la tecnología de objetos distribuidos tiende a estandarizarse, parece importante que la industria de gestión de telecomunicaciones migre hacia este nuevo paradigma de sistemas distribuidos por un tema de costos y soporte.

Este apartado se organiza de la siguiente manera. En primer lugar se introduce el concepto de Building Block (BB) como una noción abstracta de una entidad "utilizable" de software con múltiples interfaces y como se inserta éste en el modelo de tres capas de sistemas distribuidos.

En segundo lugar se presenta el concepto de Arquitectura de Aplicación o AA *(Application Architecture)*. La misma es un conjunto de restricciones genéricas para poder soportar operabilidad, interoperabilidad y empaquetamiento en sistemas distribuidos. Esta arquitectura provee al programador de un marco consistente para el ensamblado de objetos en paquetes que puedan ser utilizados *(deployable package)* y que pudieran fácilmente interoperar en tiempo de ejecución con paquetes similares[41].

2.3.2.2 Características de los Building Blocks

Un BB puede definirse como una unidad *"deployable"* de software interoperable (se utiliza la expresión *"deployable"* para referirse a las actividades vinculadas a la instalación, actualización, arranque y detención de una aplicación o componente).
Otras definiciones podrían ser: "un paquete de objetos", una entidad con una granularidad más grande que la de objetos o una especie de contenedor de objetos.

Características y conceptos de los BB[41]:

- Ningún BB puede por si solo resolver un problema de negocio.
- Un conjunto de BB puede formar parte de un Sistema Lógico *(Logical System)* conectado por medio de un bus (de tipo *"plug and play"*) llamado Entorno Distribuido de Procesamiento o DPE *(Distributed Processing Environment)*.
- Los BB deben tener roles bien definidos en un sistema lógico.
- TOM y TMN pueden ser usados para determinar la mejor división de roles *(separation of concerns)*.
- Un BB puede ser reusado por múltiples sistemas lógicos, y una sola instancia de un BB puede participar concurrentemente en un número de sistemas lógicos que usen sus funcionalidades.
- Un BB tiene la propiedad de encapsulamiento *(object encapsulation)* permitiendo acceso solo a través de sus interfaces públicas reservando sus detalles de implementación interna (como información privada). Un BB puede estar formado por un conjunto de componentes distribuidos que corren en diferentes nodos de ejecución (hosts) pero la interfaz entre estos componentes es privada en este caso.

- Las interfaces públicas de un BB se llaman contratos *(Contracts)*. Además de la metadata asociada a las interfaces se especifican requerimientos y condiciones de uso.
- Un BB separa sus interfaces públicas en:
  - Contratos de servicio, interfaces que proveen un servicio.
  - Contratos de gestión, interfaces que proveen la capacidad de gestionar y configurar un servicio.
- Las interfaces de un BB están disponibles (con restricciones de seguridad) en un sistema distribuido por medio de un DPE negociación de servicio *(Trading Service)* (páginas amarillas para contratos).
- Los BB usan el *Trading Service* (ver 2.3.2.3.2.2) para ubicar contratos disponibles en otros BB.
- La funcionalidad de un BB debe residir totalmente dentro de una de las tres capas de sistemas distribuidos definidas como: EIT *(Enterprise Information Tier)*, PAT *(Process Automation Tier)* y HIT *(Human Interaction Tier)*.

  - **EIT**
    Esta capa esta relacionada con el almacenamiento y mantenimiento de los datos de la empresa *(enterprise data)*, por ejemplo, datos usados por muchos procesos del negocio.
  - **PAT**
    Esta capa esta relacionada con operación y gestión del negocio
  - **HIT**
    Esta capa esta relacionada con la interacción humano/computadora. Se siguen principios de diseño centrado en el usuario *(User Centered Design, UCD)*.

- Un BB es gestionable por medio de una "Interfaz Común de Gestión de la Aplicación" o CAMI *(Common Application Management Interface)* permitiendo al administrador del sistema tener control sobre él y estar informado acerca de su funcionamiento y rendimiento. Esta interfaz debe ser implementada por todos los BB.

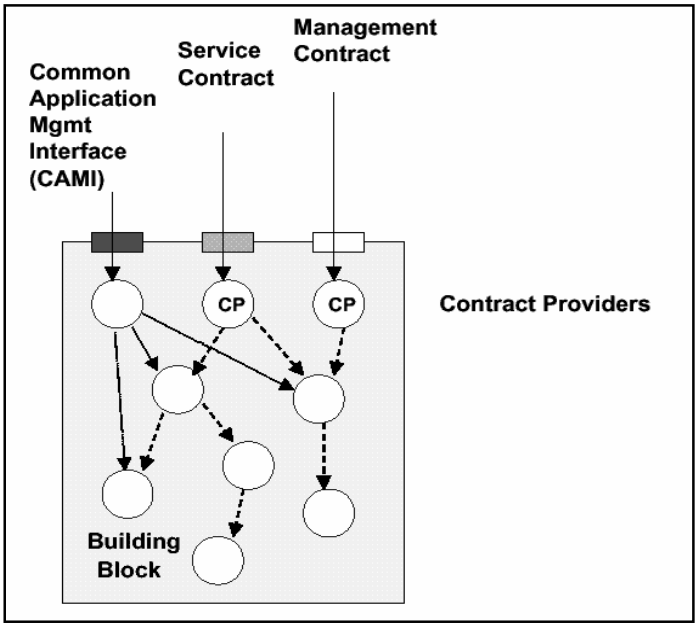


Figura 9 - Esquema general de un BB (ref.[44])

En la Figura 9 se muestra el esquema general de un BB. En la parte superior están señalados los servicios brindados. Se puede observar la Interfaz de Gestión Común, Contratos de Servicio y Contratos de Gestión. En su interior se visualizan objetos que implementan cada uno de los servicios ofrecidos a través de Contratos, denominados CP *(Contract Providers)*.

Uno de los objetivos finales de la definición de BB es promover un escenario que permita establecer un mercado de componentes para los objetos de negocio comunes *(Common Business Objects)*.

En la industria de gestión de telecomunicaciones, emergerán BB genéricos para satisfacer los requerimientos basados en el análisis de los procesos de negocios señalados en el TOM *(TeleManagement Forum's Telecom Operations Map)* y documentos relacionados.

2.3.2.3 Arquitectura de Aplicación

El concepto de Building Block no es implementable en forma directa y aislada. La idea de construcción de software basado en Building Blocks está fuertemente atada a la existencia de un marco de trabajo apropiado para soportarlo. En este sentido se requiere definir una arquitectura básica que permita el funcionamiento de los mismos[41].

2.3.2.3.1 Objetivos de la Arquitectura

Operabilidad. Debe existir la habilidad de manejar y controlar eficientemente y a costo razonable el desarrollo, la administración, la ejecución y el acceso de usuario a una colección débilmente acoplada *(loosely coupled)* de productos de software, independientemente de sus proveedores, para alcanzar los objetivos del negocio en cuanto a rendimiento, disponibilidad, confiabilidad y seguridad se refiere.

Interoperabilidad. Se define como la habilidad para interconectar productos de software que proveen una funcionalidad de negocio, independiente de su proveedor y de la versión, para proveer acceso a los datos y funcionalidades, a un usuario autorizado, y mantener interconexión y acceso aún cuando haya cambios en el proveedor o versiones.

Las interacciones deben ocurrir vía interfaces bien definidas y bien documentadas las cuales son publicadas a través de un *Trading Service*, que permite a un cliente buscar servidores de contratos por su tipo sin saber el nombre o ubicación de la instancia específica.

Commoditization. Los Building Blocks comunes deben estar disponibles y deben ser fáciles de conseguir. Foros industriales, como TeleManagement Forum, deben definir y probar *(testing)* contratos para BB comunes. Un BB de un proveedor debe poder interoperar con otro BB de otro proveedor por medio de interfaces conocidas y bien documentadas.

Gestión Común. Los BB deben ser gestionables como aplicaciones de forma común.
Una empresa debe poder gestionar BB de proveedores diversos con un único sistema de gestión.

2.3.2.3.2 Requerimientos

2.3.2.3.2.1 Requerimientos Básicos

En primer lugar existe un conjunto básico de requerimientos que deben cumplir siempre todos los BB y que están relacionados con:

- Los objetivos de la Arquitectura de Aplicación
- El acceso externo de los BB (diseño de contratos)
- Requerimientos que no son fácilmente cambiables o modificables en etapas posteriores

2.3.2.3.2.2 Trading Service

El conjunto de requerimientos relacionados con la existencia de un servicio para ubicación de Contratos (como el *Trading Service*) es importante cuando la empresa tiene un gran sistema de red distribuido y es difícil conectar todo en forma estática por medio de un servicio de nombres. El *Trading Service* hace posible que los BB se encuentren unos a otros en forma dinámica en tiempo de ejecución por medio de características del servicio.

2.3.2.3.2.3 Independencia de Versión

Este requerimiento se refiere al hecho que una instancia de un BB debe poder ser instalada, actualizada, cambiada, y/o activada sin requerir actividades similares en otras instancias de sí mismo o de otros BB. Cuando una nueva versión de un BB es instalada, esta no puede ocasionar una pérdida en alguna funcionalidad requerida por un cliente instalado.

2.3.2.3.2.4 Resistencia a Fallos

Este grupo de requerimientos es necesario en servicios de alta confiabilidad. Ellos permiten que los servicios detecten problemas y se restablezcan automáticamente de los mismos cuando el fallo en la red es reparada.

Un BB debe poder sobrevivir en una amplia gama de escenarios de fallo. En la fase de fallo el BB debe emplear estrategias para resistir el mismo.

2.3.2.3.2.5 Requerimientos transaccionales

Este grupo de requerimientos es necesario cuando un servicio no puede ser implementado en forma independiente, o cuando los BB van a ser usados por clientes transaccionales.

2.3.2.3.2.6 Seguridad

Este grupo de requerimientos es necesario cuando la seguridad se vuelve crítica, cuando un servicio traspasa el límite administrativo de un proveedor de servicios, permitiendo el acceso al sistema de otros proveedores de servicio o clientes.

2.3.2.3.2.7 Common Application Management Interface

La gestión de aplicación es importante para el proveedor de servicios que esta ejecutando sus BB en múltiples máquinas físicas interconectadas por red.

Cuanto más compleja es la red distribuida, más importante es disponer de un diagnóstico preciso y capacidad de gestión de todos los BB que en ella se están ejecutando. Todos los BB deben tener interfaces idénticas para proveer capacidad general de gerenciamiento.

El requerimiento de una interfaz de aplicación de gestión provee a los BB de la posibilidad de que sean operables. Debe ser posible para un administrador del sistema de poder arrancar, parar y monitorear el estado de un BB en tiempo de ejecución.

2.3.2.4 Construcción de Sistemas Lógicos basados en BB

Se denomina sistema lógico a una colección de BB que colaboran para resolver un proceso del negocio. En un sistema lógico interactúan *n* instancias de *m* tipos de BB. Un tipo de BB puede ser usado en diferentes sistemas lógicos. El sistema lógico es no encapsulado y no es un objeto. El BB es la mayor entidad en la que es posible encapsular.

2.3.2.5 Separación de ámbitos de los BB

La funcionalidad de un BB debe residir totalmente dentro de una de las tres capas: EIT *(Enterprise Information Tier)*, PAT *(Process Automation Tier)* y HIT *(Human Interaction Tier)*.

Cada BB tiene un ámbito de acción definido y no existe superposición de tareas en este sentido.

2.3.2.5.1 Process Automation Tier

Esta capa implementa las reglas del negocio a partir de un modelo de los procesos del negocio. No se asume ningún tipo particular de presentación o interacción humana.

La información de la empresa se obtiene de EIT.

2.3.2.5.2 Human Interaction Tier

La capa HIT describe los componentes relacionados con la interfaz humana.

Las tres funcionalidades básicas de los HIBB (BB en HIT) son:

- Implementación de la interfaz con el usuario que estructura la presentación al usuario de las funcionalidades del sistema.
- Gestión de tareas vinculadas a las metas y tareas del usuario. Contiene las funciones necesarias para ejecutar las reglas del negocio que requieren interfaz humana.
- Gestión de seguridad. Identifica y autentifica al usuario determinando su nivel de acceso a las funciones de los BB.

2.3.2.5.3 Enterprise Information Tier

La capa EIT, describe la estructura del negocio, así como las reglas por las cuales éste se rige. En particular EIT es la capa responsable del almacenamiento y mantenimiento de la información empresarial permitiendo el acceso a toda la información empresarial como un todo interconectado.

Al igual que la HIT y la PAT, esta capa es implementada a través de un conjunto de BB.

Las principales funcionalidades de los EIBB son:

- Acceso abierto desde otros BB, bajo controles de seguridad.
- Manejo transparente de la redundancia y distribución de la información.
- Asegurar la integridad de los datos.

Los EIBB no están ligados a un proceso u operación en particular. Su construcción está basada en la estructura de la información y las reglas de negocio definidas y no en los requerimientos específicos de las demás capas.

2.3.2.5.3.1 Modelos de Información

Concretamente, los EIBB (BB dentro de la capa EIT) implementan modelos de información *(information models)* que representan accesos a datos, tanto predefinidos como ad-hoc, así como controles a las restricciones de integridad SIC *(semantic integrity constraints)* definidas para los datos.

Cada conjunto cooperativo de EIBBs, o un EIBB que no sea parte de un conjunto cooperativo de EIBBs, deben asegurar que se cumplen las SIC definidas sobre los datos asociados al EIBB. También se incluye los casos de datos compartidos o replicados entre varios EIBB. Este control se realiza efectivamente en las operaciones de modificación de datos. Por lo tanto, en los casos de múltiples EIBB manejando un mismo conjunto de datos, se deben prevenir las inconsistencias en las SIC controladas.

2.3.3 Arquitectura NGOSS

2.3.3.1 Metas y Objetivos

Se intenta proveer un "framework" para la integración rápida y flexible de Sistemas de Soporte de Operaciones y del negocio en las telecomunicaciones e incluso más allá de la industria de las telecomunicaciones[40].

Se pretende abordar el problema en una forma independiente de la tecnología. No se apoya sobre ninguna tecnología en particular, en lugar de eso, apunta al soporte de una federación de diferentes tecnologías (donde cada una de las cuales ofrece ventajas en diferentes niveles). Se busca que los conceptos y los principios del negocio manejen el diseño del sistema y su arquitectura (y no al revés). Esto se puede implementar usando las tecnologías de servicios de información disponibles. Es crítico en este punto la posibilidad de usar y compartir información y datos comunes.

La meta es definir una arquitectura del sistema distribuida, basada en componentes.

La especificación de una arquitectura "neutral" facilita el uso de distintas tecnologías el desarrollo de sistemas para la gestión coordinada (de uno o varios OSS). De la misma manera facilita la adopción de nuevas tecnologías, a medida que vayan adquiriendo la madurez requerida.

2.3.3.2 Principios

Se han adoptado un conjunto de principios que surgen de *"best practices"* de la industria en general [40] que se enumeran a continuación.

2.3.3.2.1 Separación de los procesos del Negocio de la implementación de componentes

La arquitectura en general se caracteriza por la separación del comportamiento *"hard coded"* de los componentes y el software que automatiza los procesos del negocio sobre estos componentes. Un sistema se caracteriza por "Descripciones de su comportamiento" expresadas en una forma independiente de la tecnología empleada.

2.3.3.2.2 Interfaces definidas a través de "Contratos"

Un sistema NGOSS se caracteriza por el hecho de que todas las entidades de software que proveen servicios a otras entidades, lo hacen a través de interfaces denominadas "Contratos" debidamente especificadas. Se incluye metadata del servicio provisto, condiciones bajo las cuales se implementa el servicio, excepciones que se generan eventualmente, etc.

2.3.3.2.3 Gestión basado en políticas

Esta arquitectura utiliza la gestión basada en políticas. Las políticas constituyen "instrucciones" que definen objetivos y formas que especifican como se va a gobernar el sistema (sin requerir de mecanismos específicos en la implementación del sistema).

2.3.3.2.4 Roles de Usuarios y Seguridad

Un sistema NGOSS se caracteriza por que el usuario realiza el *"log-on"* al sistema en forma general. Se definen las posibilidades de acceso a los distintos servicios en función del perfil del usuario.

2.3.3.2.5 Vehículo de Comunicaciones

Un sistema NGOSS se caracteriza por la existencia de un servicio de comunicaciones (por ejemplo un "bus" de comunicaciones) o alguna forma que permita a las entidades "interoperar" entre sí. Cada servicio de comunicaciones ofrece uno o más mecanismos de transporte. Es importante destacar que puede existir más de un servicio de comunicaciones dentro de una misma implementación y en estos casos cada servicio de comunicación representa distintos mapeos hacia tecnologías específicas.

2.3.3.2.6 Información compartida

El sistema en general se caracteriza por el uso de un único modelo de información común, denominado Modelo de "Servicios de Información y Datos compartidos" o SID (*Shared Information and Data*). Este modelo está diseñado para que sea más que una simple representación de los datos que maneja la empresa. Define la semántica, comportamiento e interacción entre las entidades gestionadas. En general toma dos formas: Diagramas UML *(Unified Modelling Language)* de clases [2] y un diccionario de datos que define la sintaxis, semántica y el uso de clases y atributos en general.

Los contratos no forman parte de la SID. Por el contrario, sí es posible que las especificaciones usen información del SID.

2.3.3.2.7 Adaptación de Información

La arquitectura describe un framework "neutral" de servicios, componentes y modelo de información. En el mundo real no siempre se posible establecer una conexión directa con este modelo ideal. Se requiere

del concepto de "adaptación". "Adaptación" es un mecanismo para convertir una entidad en otra con diferente representación, de manera de poder lograr la interoperación entre ellas.

2.3.3.2.8 Sistemas distribuidos escalables débilmente acoplados

En general los OSS contemporáneos se pueden describir como altamente acoplados, compuestos de un número de aplicaciones monolíticas, cada una de las cuales ofrece un conjunto de servicios dentro de un dominio particular. Estos sistemas legados se pueden integrar utilizando "gateways" de mediación (usando un acoplamiento débil). El sistema NGOSS se caracteriza por el hecho de que las entidades de software, también llamadas componentes o *"Building Blocks"*, están débilmente acopladas entre sí y además distribuidas.

2.3.3.2.9 Framework Services

La arquitectura se caracteriza por la existencia de un conjunto de servicios básicos (del propio framework) que facilitan la operación de los componentes del sistema. Algunos de estos servicios son:

- *Servicio de Invocación.* Permite que las entidades puedan invocar servicios provistos por otras entidades de acuerdo a las reglas del contrato del servicio y las políticas del sistema.
- *Gestión de Información y Datos compartidos*. Gestión de integridad de los datos requeridos por una o más entidades.
- *Trading*. Servicio que posibilita encontrar otros servicios en función de sus atributos característicos.
- *Policy*. Servicios que definen, gestionan y ejecutan las definiciones de "Políticas" del sistema.
- *Seguridad.* Mecanismos que posibilitan la autenticación, autorización y auditoría.

2.3.3.2.10 Separación de la "Arquitectura Neutral" y la "Arquitectura dependiente de la tecnología"

Un sistema NGOSS se caracteriza por el hecho de que todos los contratos, ya sea que estén vinculados a servicios de negocios o servicios básicos *(Framework Services)*, están documentados en una especificación que es neutral con respecto a cualquier tecnología. Existen distintos trabajos en curso para producir el mapeo entre esta arquitectura neutral y distintas tecnologías aplicables[40].

2.3.3.3 Arquitectura de NGOSS

El sistema se construye sobre una colección de componentes débilmente acoplados, cada uno de los cuales implementa uno o más servicios del negocio. Los servicios del negocio están soportados por un conjunto de componentes bien definidos, débilmente acoplados, que brindan servicios básicos *(Framework Services)*.

El modelo de información y datos compartido (SID) se usa para representar:

- Atributos y operaciones de entidades gestionadas.
- Relaciones que la entidad gestionada mantiene con otras entidades del sistema.
- Comportamiento o Semántica que gobierna la forma en que se comporta una entidad gestionada.
- Meta-información de componentes en general.

A continuación se discute la estructura de un sistema NGOSS desde distintos puntos de vista.

2.3.3.3.1 Vista Estructural de la arquitectura

La Figura 10 muestra la arquitectura de NGOSS. Se visualiza la relación entre Servicios Básicos, Servicios de Políticas y Servicios del Negocio. Los servicios básicos brindan funcionalidad al nivel más bajo. Estos son usados por los Servicios del Negocio para satisfacer las especificaciones de "Contratos".

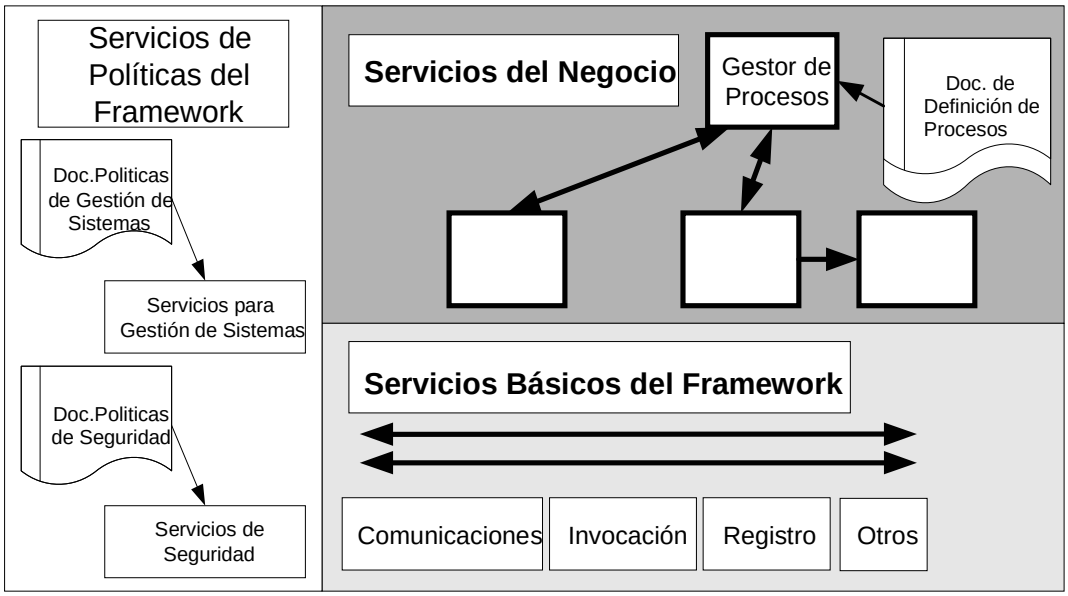


Figura 10 - Arquitectura NGOSS

Notar el uso del "Gestor de Procesos" que permite integrar los servicios del Negocio. Notar también que dos procesos de Negocios se comunican entre sí eventualmente sin el uso del "Gestor de Procesos". La definición de los procesos del negocio reside en el documento de definición del modelo de procesos (usado por el Gestor de Procesos para ordenar la ejecución de servicios en forma consistente con el modelo). De la misma manera los Servicios de Seguridad y de Gestión de Políticas usan los documentos de definición de Seguridad y Políticas respectivamente.

Los Servicios Básicos pueden ser utilizados por otros servicios. Por ejemplo, el documento de definición de procesos puede ser accedido desde el servicio de Registro General *(Registry)*. De la misma manera, el uso del servicio de Invocación, puede requerir del servicio de Seguridad para determinar si es posible llevar a cabo la ejecución del servicio.

La arquitectura especifica dos clases fundamentales de servicios:

- Servicios del negocio. Brindan elementos de la funcionalidad del negocio a través de la especificación de los contratos.
- Servicios básicos. Brindan servicios básicos para soportar la interacción entre componentes que implementan los servicios del negocio. Por ejemplo: gestión de procesos, soporte de transacciones, seguridad, etc. Estos servicios se brindan también a través de especificaciones de contratos apropiadas.

Los Servicios Básicos incluyen Comunicaciones, Registro General, Naming Service, Trading Service, Gestión de Información y Datos Compartidos, Gestión de Políticas, etc. Estos servicios surgen de las necesidades específicas de los servicios típicos de negocios y de los servicios en general provistos en ambientes distribuidos.

2.3.3.3.2 Modelos de Información

Nos referimos aquí a la especificación externa de las características y comportamiento de las entidades gestionadas. Existen dos tipos de modelos:

- El *modelo de información* es una abstracción y representación de las entidades en el dominio gestionado. Esto incluye la definición de atributos, operaciones y relaciones entre ellos. Es independiente del tipo de repositorio, software o protocolo utilizado.

- El *modelo de datos* es una implementación concreta del modelo de información, en términos de las plataformas específicas involucradas.

Los modelos pueden tomar muchas formas, incluyendo las especificaciones de diagramas de clases, metadata, objetos del negocio, contratos, procesos, políticas. Es importante destacar que los modelos se usan para posibilitar un sistema extensible, en lugar de dejar información *"hard-coded"* dentro de la aplicación.

Se intenta hacer pública está información de una manera "neutral" respecto a las tecnologías. Se puede acceder a esta información a través del servicio de Registro General.

Nos referimos a Metadata como datos que describen a otros datos. Se provee información de cómo utilizar datos e interfaces.

Se ha definido usar UML *(Unified Modelling Language)*[2] para definir información y datos compartidos. Para expresar comportamiento puede utilizarse OCL siempre que sea posible (para mostrar "expresiones" que afectan a las entidades gestionadas).

2.3.3.3.3 Especificaciones de Contratos

Un contrato es un acuerdo de provisión de información o realización de una operación. Una especificación de un contrato define la semántica de la invocación y comportamiento del servicio que se ofrece. Se provee de una interfase bien definida para acceder a los componentes.

La especificación de un contrato asociado con el servicio del negocio incluye:

1. Una descripción de la operación provista (con entradas y salidas).
2. Referencias a Servicios Básicos *(Framework Services)* requeridos por el servicio del negocio de manera de cooperar con otros servicios de acuerdo al patrón de interacción deseado (por ejemplo, interactuar con otro servicio de negocio en el contexto de una misma transacción).
3. Una descripción de posibles excepciones y errores que pueden ocurrir en la ejecución del servicio.
4. Metadata que describe cómo las entidades afectadas por la ejecución del servicio reaccionan e interactúan con otras entidades gestionadas.
5. Una lista de precondiciones que se deben cumplir previo a la ejecución del servicio.
6. Una lista de poscondiciones que se van a cumplir luego de la ejecución exitosa del servicio.
7. Una lista de excepciones y notificaciones que pueden ser generadas por la ejecución del servicio.
8. Una descripción de los Objetos del ámbito de Información compartida que son alterados por la ejecución del servicio.

Una especificación de contrato invoca conceptos manejados en el "modelo". Por ejemplo, la especificación de entrada y salida, usan la metadata que describe los "tipos" de datos.

Las especificaciones de precondiciones y poscondiciones usan facilidades de la lógica del negocio para describir las condiciones a verificar al comienzo y fin de la ejecución del servicio. Se espera que estas condiciones se expresen en términos de lógica de predicados, de manera de poder automatizar la verificación de las mismas. Es responsabilidad del entorno que invoca el servicio el verificar la validez de las precondiciones. No se podrá invocar el servicio si no se verifican las precondiciones. Luego la implementación del contrato debe verificar que las entradas son correctas. Finalmente si la implementación no puede verificar las poscondiciones, se termina el servicio con una Excepción.

2.3.3.3.4 Uso de contratos

2.3.3.3.4.1 Ubicación de Servicios (Registry Service).

Tanto los clientes como los componentes necesitan ubicar otros contratos y los componentes que los implementan sin conocimiento de su ubicación física en la red (transparencia en la distribución de los mismos).

La entidad esencial que soporta la transparencia de la Distribución de componentes a lo largo de la red es el Registro General. En este sentido se brindan los servicios de:

- *Naming Service*: Permite manejar objetos en base a nombres.
- *Directory Service*: Establece vínculos entre nombres y Objetos.
- *Trading Service*: Permite encontrar objetos en base a atributos que han sido registrados.

El *Trading Service* extiende la funcionalidad del servicio de Registro soportando la posibilidad de vincular atributos a objetos. De la misma manera permite ubicar dentro de un conjunto de objetos aquel que mejor se aproxima a un criterio de búsqueda especificado (en función de los atributos). Este servicio es útil para ubicar especificaciones de contratos, dando las propiedades de contratos en lugar de las identificaciones de los mismos.

A continuación se visualiza un diagrama que visualiza el registro de un componente y como más tarde es encontrado por otros componentes del sistema.

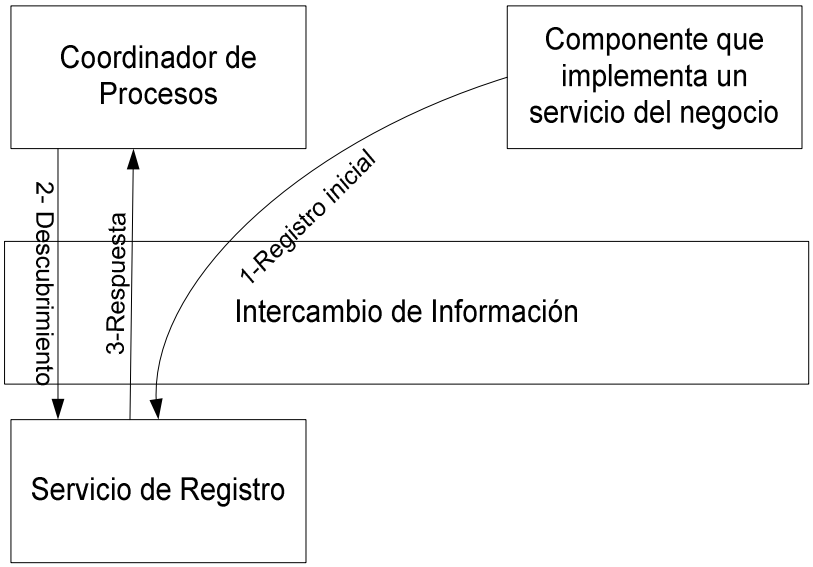


Figura 11 – Ubicación de Servicios

2.3.3.3.4.2 Servicio de Invocación (Invocation Service)

Existen varios pasos asociados a la ejecución de un servicio en un componente remoto, siendo el paso más crítico asegurar que las precondiciones y poscondiciones de los contratos sean verificadas. Complementariamente, la seguridad debe ser aplicada en cada uno de los pasos del proceso verificando las autorizaciones requeridas. La existencia del servicio de invocación permite llevar a cabo distintas operaciones, asegurando la integridad de las mismas, en el contexto del sistema y del cliente que invoca el servicio.

Los pasos para procesar la invocación son:

- Ubicar la implementación de un Contrato de Negocio.
- Interrogar al Servicio de Seguridad para determinar si la operación está autorizada.
- Si la autorización falla, se reporta una excepción, de lo contrario continúa.
- Verificar las precondiciones del Contrato de Negocio. Si estas condiciones no se cumplen se genera una excepción.
- Invocación efectiva. Según sea el mecanismo de implementación, se arma el "Request" y se espera por la respuesta.
- Recepción de la respuesta. Se verifican las poscondiciones del Contrato. Si no se cumplen se genera excepción, de lo contrario continúa.
- Log eventual de la invocación.
- Retornar la respuesta al solicitante de la invocación.

2.3.3.3.5 Gestión de procesos

La premisa básica es poder construir sistemas OSS altamente escalables y flexibles que posibiliten el desarrollo de nuevos servicios.

El servicio de gestión de procesos actúa como un coordinador de las actividades realizadas en el sistema (operaciones llevadas a cabo por componentes). El medio para expresar la lógica de ejecución del servicio de gestión de procesos es diferente al empleado en la implementación de componentes. Básicamente consiste de definiciones en un lenguaje neutral de las invocaciones de contratos requeridas.

Esto facilita el reacomodar o alterar los pasos del proceso del negocio (modificando la interacción entre componentes) siempre que sea necesario. La forma en que se ordenan las ejecuciones de contratos de negocio no está fija en el sistema (como lo sería en un sistema monolítico) sino que está expresada en las definiciones manejadas por el gestor de procesos.

La Figura 12 muestra al gestor de procesos con un plan de procesos que puede sugerir distintos caminos en lo que tiene que ver con la invocación de contratos. En la parte inferior se observan paquetes de contratos agrupados en *Building Blocks* (los rectángulos grises representan contratos, mientras que las lineas ovaladas representan *Building Blocks* que agrupan Contratos). Las flechas bidireccionales muestra contratos que se invocan y retornan la indicación de su finalización.

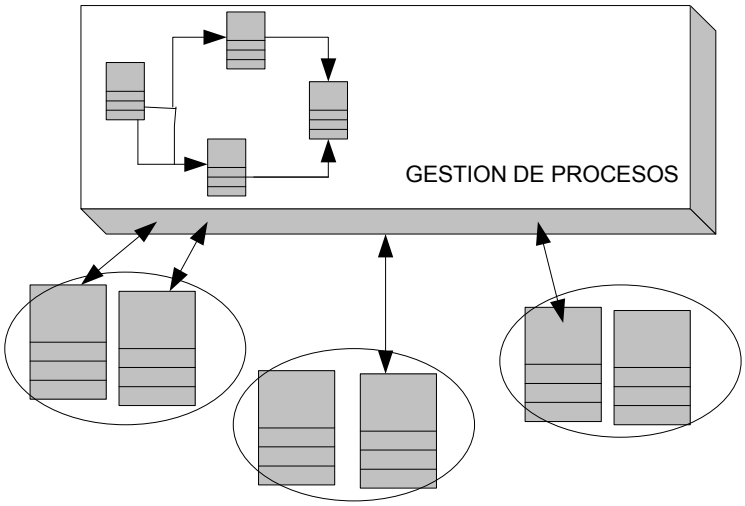


Figura 12 - Gestión de Procesos

2.3.3.4 Gestión de Información Compartida

El propósito es definir una estructura común de información, características, comportamiento, y relaciones entre las distintas entidades. Un modelo de información es independiente del tipo de repositorio, protocolo de acceso, implementación de software o plataforma usada. El modelo SID *(Shared Information and Data)* es federado por dos razones. Primero, habilita al SID a funcionar como un framework, donde los detalles de bajo nivel pueden ser brindados por la SID o por otros modelos existentes de otras comunidades. Segundo, habilita que las aplicaciones que visualicen al SID como un conjunto extensible de *Building Blocks*. No necesitan usar el SID en toda su extensión, sino que usan solo aquello requerido por la aplicación [39].

Este modelo de información unificado se traduce a un conjunto de modelos de datos. Cada modelo de datos se optimiza para una combinación específica de repositorio y protocolo de acceso usada para gestionar tipos particulares de datos. Esto es necesario, porque la gestión de datos es naturalmente diversa, y ningún repositorio o protocolo de acceso tienen la potencialidad para gestionar todos los datos en forma concurrente. Esto ilustra la necesidad de tener un único modelo de información. Si hubiera múltiples modelos de información la coherencia de los datos de una entidad dada no se podría asegurar a lo largo de diferentes implementaciones en repositorios diferentes.

El SID habilita que dos servicios del negocio se pongan de acuerdo en definiciones comunes de los datos compartidos (a los efectos de que puedan ser usados en las aplicaciones respectivas). Cada uno es libre de refinar los datos compartidos para adecuarse a sus propias necesidades, pero pueden interoperar solamente utilizando la representación de los datos compartidos.

El concepto de SID es fundamental en estos sistemas de operaciones. Representa el medio por el cual la información útil, relevante para un conjunto de procesos, pueda ser factorizada, compartida y gestionada por múltiples procesos de negocio a lo largo del sistema general.

Las instancias de elementos definidas por la SID son accedidas por las distintas entidades a través de interfaces bien definidas (Contratos) denominados “Proveedores de Información”. Los proveedores de información soportan los contratos de servicios de información. Colectivamente el conjunto de servicios de información provisto a través de estas interfaces proveen acceso a la totalidad de la información compartida.

Para que el concepto lógico de SID pueda ser implementado, se recurre al soporte de capas inferiores de la siguiente manera (ver Figura 13):

- A nivel más alto están los Especificaciones de contratos (soporte del propio SID).
- La capa de Servicios de Información agrega un valor agregado en términos del negocio, transformando datos crudos en información útil para el negocio (puede incluir *Data Mining* y filtrado). Esta capa también es responsable de la adaptación y/o mediación desde el formato común de la empresa al formato particular expresado en el contrato ofrecido.
- La capa de distribución de datos da soporte a los servicios de información, brindando la posibilidad de acceder y compartir datos ubicados en repositorios distribuidos (desde el punto de vista físico). Se incluye soporte para la sincronización de datos y mantenimiento de la integridad referencial y transaccional.
- A nivel más bajo están los repositorios físicos que guardan y mantienen los datos.

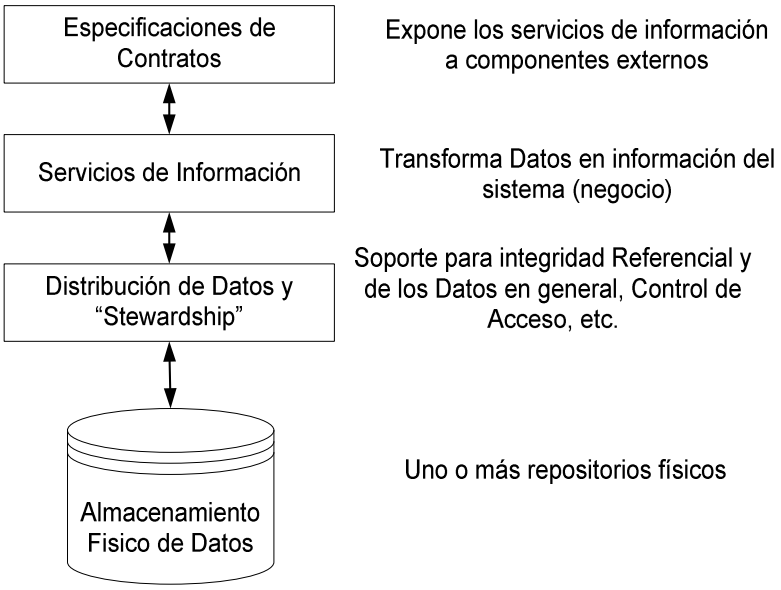


Figura 13 – Información Compartida

El SID es un modelo de información construido refinando el metamodelo UML. El SID consiste de un conjunto de entidades que están organizadas en conjuntos de dominios de gestión.

Los componentes que ofrecen servicios, lo hacen en términos de tipos de datos comunes, Éstos están definidos en el SID. Se usan mecanismos de extensión de UML (Estereotipos, "*tagged data*", OCL) para extender el metamodelo UML.

El beneficio de este enfoque es que logra desacoplar los datos del productor y consumidor. En lugar de que cada componente quede fuertemente acoplado a otro a través del uso de estructuras de datos propietarias, se acopla utilizando la especificación común SID. Por lo tanto cuando un componente requiere ser quitado otros componentes no se ven afectados.

2.3.4 Conclusiones

La recomendación GB909 establece un marco desde el cual, partiendo de los procesos del negocio de una empresa de telecomunicaciones, define la arquitectura de un sistema de Gestión de Redes de Telecomunicaciones.

Está basada en algunos principios básicos que deben cumplirse para adaptarse a los cambios tecnológicos y del negocio de la empresa. Estos principios básicos son: Interoperabilidad, procesamiento distribuido y acoplamiento débil entre componentes, reutilización, alineación con el modelo de negocio de la empresa (TOM).

Para implementar estos principios se basa en los siguientes conceptos fundamentales:

- Uso de BB, componentes de software con alta cohesión y que interactúan entre sí para alcanzar un conjunto de funcionalidades del sistema (sistema lógico).
- Separar la arquitectura en tres capas: interfaz humana, procesos del negocio, información de la empresa.
    - Una metodología de análisis y diseño para los diferentes sistemas lógicos existentes.

La norma GB909 establece un marco de trabajo que se basa en conceptos ampliamente aceptados en el desarrollo de este tipo de sistemas. Los conceptos fundamentales que se manejan no están ligados a un proveedor o tecnología, permitiendo a una aplicación evolucionar y adaptarse a diversas tecnologías.

Por otra parte, ante los fuertes cambios tanto tecnológicos como en el negocio que sufren hoy en día las empresas del área de Telecomunicaciones, hacen pensar que este marco de trabajo permita un desarrollo altamente flexible e interoperable y que se puede adaptar a las evoluciones que pueda sufrir la empresa en el futuro.

La arquitectura NGOSS de TMForum (presentada muy sintéticamente en la sección anterior), constituye un marco para el desarrollo y puesta en marcha de sistemas de soporte a nivel de operaciones y del negocio de las Telecomunicaciones en general.

Desde el punto de vista técnico algunas de sus características son:

- Las implementaciones NGOSS están constituidas por sistemas débilmente acoplados entre si.
- Las implementaciones NGOSS usan datos compartidos de la empresa.
- Se busca la separación entre el control de flujo de procesos del negocio de la operación de los componentes del negocio.
- Las implementaciones NGOSS utilizan un mecanismo de comunicaciones común.
- Se posibilita el reuso de Componentes NGOSS a través de definiciones claras de Contratos de acceso a los mismos.

2.4 Análisis, Diseño y Modelado en términos neutrales aplicado al área de gestión de redes y servicios.

2.4.1 Introducción

Es común que las especificaciones de Interfaces existentes estén atadas a tecnologías específicas lo cual reduce la longevidad de la especificación y atenta contra la interoperabilidad con componentes existentes en el largo plazo, cuando se produce la evolución del sistema en general.

El problema de las dependencias tecnológicas en el proceso de diseño de un sistema en una fase temprana ha ido más allá del dominio de la gestión de las comunicaciones. Cada vez es mas fuerte la promoción de la idea del desarrollo basado en Modelos *(Model Base Development)* como por ejemplo a nivel de OMG, se ha estandarizado la arquitectura MDA *(Model Driven Architecture)*[15]. MDA promueve la generación de modelos independientes de la tecnología utilizando el lenguaje UML *(Unified Modelling Language)* en distintos puntos del desarrollo del ciclo de vida del software, como por ejemplo, los requerimientos, análisis del sistema, diseño, etc. Esto habilita que la conexión con tecnologías específicas se produce lo más tarde posible (en la etapa de la propia implementación). La arquitectura MDA define mecanismos para intercambiar modelos de software basados en un metamodelo Standard, el MOF *(Meta Object Facility)*, el cual permite que modelos (sin dependencias tecnológicas específicas) pueden ser intercambiados y reusados entre organizaciones distintas independientemente de las tecnologías empleadas.

Han existido muchas propuestas para el modelado independiente de la tecnología en distintos niveles de abstracción en el dominio de la gestión de las telecomunicaciones. TINA-C *(Telecommunications Information Networking Architecture Consortium)* encaró el tema basado en la arquitectura ODP *(Open Distributed Processing)*[10], aplicado tanto al control como al software de gestión. Al usar ODP se produce una perfecta separación de ámbitos, de tal manera que los asuntos vinculados a las tecnologías de procesamiento distribuido utilizados están separados del modelado de requerimientos del negocio, estructuras de información, interfaces de componentes, etc.

El DMTF *(Distributed Management Task Force)*[3] ha estandarizado (con éxito) un modelo de información común CIM *(Common information Model)* para las interfaces de gestión (en el ámbito empresarial) .

El modelo CIM especifica los objetos gestionados y sus asociaciones utilizando el lenguaje Manager Object Format (MOF), el cual puede ser parcialmente representado como diagramas de clases UML. Otros estándares se han desarrollado para definir como se pueden implementar las interfaces basadas en el modelo CIM, utilizando distintas tecnologías como Remote Procedure Call en el entorno de procesamiento distribuido (DPE), Servicios de Directorio LDAP u objetos codificados usando XML y transportados por protocolo http.

El TMForum ha estandarizado requerimientos y modelos en términos neutrales. Primero capturando modelos de procesos del negocio genéricos en el dominio de las telecomunicaciones, expuestos a través del eTOM(presentado anteriormente en 2.2.3.4). Actualmente el TMForum esta en proceso de estandarizar Interfaces de componentes y la información que circula por dichas interfaces. Estos trabajos se están llevando a cabo en el ámbito de la iniciativa NGOSS.

Otras visiones complementarias como las del proyecto FORMS, o el modelo "Arquitectural" *(Lewis, Wade y Cullen del Trinity Collage, Dublin)*[15] se apoyan también en TINA, CIM de DTMF y fundamentalmente sobre NGOSS. Esta propuesta no pretende definir un nuevo framework para el desarrollo de sistemas de gestión basados en componentes. Por el contrario, se sugiere una estructura de modelos que en combinación con metodologías basadas en UML, posibilite el estudio de las relaciones entre modelos del negocio, modelos de interfaces de contratos y modelos de información. Además de poder visualizar como estos modelos se pueden desarrollar en una manera independiente de la tecnología a emplear y como se pueden estructurar de la mejor forma para facilitar el intercambio de modelos y productos entre los "accionistas" o "depositarios" (actores involucrados) del desarrollo de un sistema de gestión basado en componentes.

2.4.2 Desarrollo basado en componentes y el enfoque MDA

Uno de los enfoques más interesantes en el área de desarrollo basado en componentes, es el de expresar soluciones en términos de nociones abstractas de componentes (independientes de la tecnología), enfoque que se denomina "desarrollo basado en modelos".

Un ejemplo importante de este enfoque es la propuesta MDA *(Model Driven Archtecture)* de OMG. Esta propuesta busca separar la lógica del negocio, de la tecnología de la plataforma subyacente. De esta manera se diseñan aplicaciones (con independencia de la plataforma) y se realizan posteriormente en distintas plataformas específicas. El concepto del modelo es central en MDA. Se distinguen dos tipos de modelos: el PIM *(Platform Independent Model)* o modelo independiente de la plataforma y el PSM *(Platform Specific Model)* o modelo específico de la plataforma. En MDA se transforma un PIM generado para un sistema en un PSM. Finalmente, se puede implementar directamente el PSM en la plataforma elegida[18].

Las aplicaciones MDA son portables. Esto es porque las herramientas MDA implementan mapeos estandarizados desde plataformas independientes a plataformas específicas utilizando UML profiles. Una aplicación MDA se puede portar a distintas plataformas partiendo del Modelo original independiente de la plataforma (PIM)[34].

Al igual que en PIM, se usan "profiles" específicos para la formulación de Modelos para Plataformas Específicas *(Platform Specific Models,PSM)*. Por ejemplo OMG ha estandarizado un "profile" UML para CORBA y EJB y está trabajando para el profile "Web Services".

Se definen mapeos estandarizados que definen el camino del modelo PIM hacia el modelo de la plataforma específica considerada. Este framework de "profiles" y mapeos estandarizados permiten que las herramientas MDA automaticen en gran parte la transformación del PIM al SIM, que termina en la propia generación del código de la aplicación.

En la tercera y última etapa,definida por MDA, se genera el código y archivos asociados a partir del PSM. Sin embargo, la mayoría de estas herramientas no genera todo el código de la aplicación, de manera que los desarrolladores tendrán que escribir secciones faltantes y chequear el código generado.

2.4.3 TMForum y NGOSS

La arquitectura NGOSS posibilita la construcción de sistemas distribuidos que soportan procesos del negocio asociados con información de gestión y dominios de servicios de comunicaciones. Se provee una forma neutral de describir la estructura de estos sistemas basados fundamentalmente en componentes y un conjunto de servicios básicos sobre los que se apoyan.

Existen distintos roles asociados al diseño de la infraestructura, diseño de componentes y diseño del sistema, cada uno con sus propias responsabilidades, como se describe a continuación:

- "Planificador y Analista de los procesos del negocio". Son los encargados de definir e identificar los flujos de procesos del negocio.
- Analista de Dominio *(Domain Analyst)*. Se enfoca en la definición del modelo de Análisis *(Analysis Model)* para un área acotada de la problemática de gestión para la cual se buscan componentes NGOSS.
- Diseñador de Contratos *(Contract Designer)*. Se enfoca en el diseño y publicación de contratos que pueden ser utilizados y compartidos por otros.
- Constructor de Componentes *(Component Builder)*. Se enfoca en el análisis, diseño e implementación, prueba y liberación de software basado en componentes.
- Constructor del Sistema *(System Builder)*. Se enfoca en analizar las necesidades del negocio, diseñar e implementar el sistema apropiado.
- Administrador del Sistema *(System Administrator)*. Responsable del monitoreo y gestión de un sistema que contiene componentes de manera que funcione dentro de los parámetros requeridos.

El Planificador de Procesos del Negocio *(Business Process Planner)*, define los procesos del negocio que son entregados al Analista de Procesos del Negocio *(Business Process Analyst)* a los efectos de desarrollar los requerimientos. Estos requerimientos se entregan al Analista del Dominio *(Domain Analyst)* de manera que con la ayuda de un experto *(Business Process Analyst)* desarrollan un modelo a nivel del dominio *(Domain Analysis Model)*. Este modelo del Dominio se utiliza a su vez para crear las especificaciones de contratos, información compartida y tipos de datos involucrados. El proceso para llegar a las especificaciones de contratos se ilustra en la Figura 14.

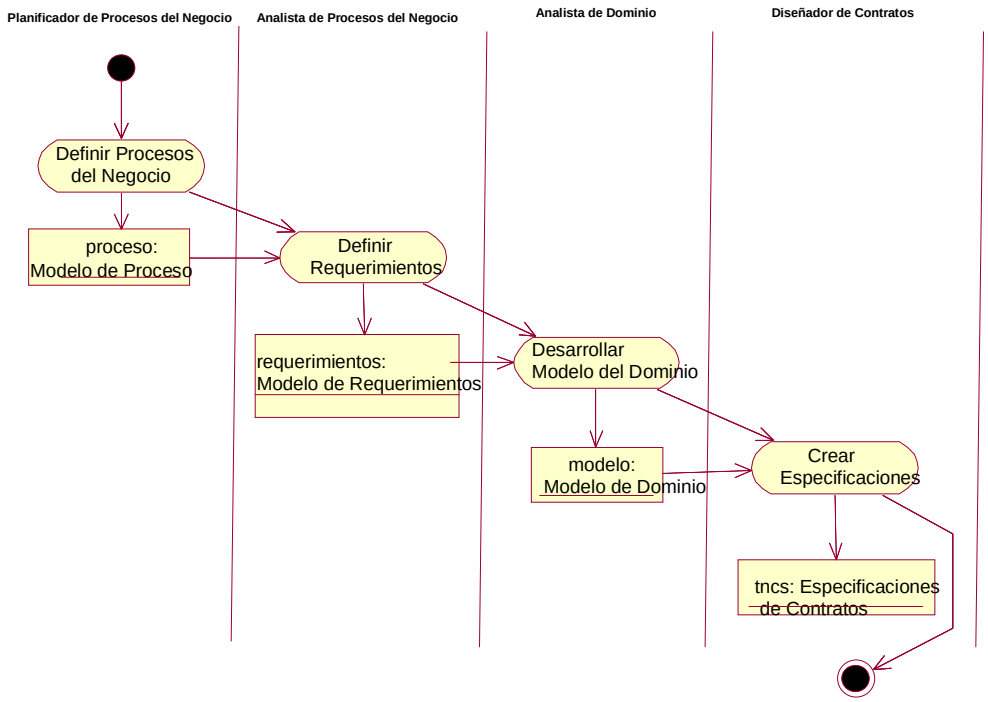


Figura 14 - Creación de Especificaciones de Contratos (NGOSS)

La tarea de crear las Especificaciones de los Contratos involucra: especificar un nombre, definir comportamiento, definir parámetros, excepciones y condiciones bajo las cuales se brinda el servicio. Todas estas tareas implican simultáneamente la definición de información y datos compartidos. Todas las operaciones involucran parámetros y objetos devueltos cuyos tipos de datos deben ser definidos. Se busca que exista reuso de las definiciones de datos de manera de compartir estas definiciones en el contexto de Dominios o sistemas.

La definición real del Contrato y la información compartida asociada depende de un conjunto de modelos. En la Figura 15 se muestra la estructura estática de estos modelos y sus interdependencias. Es inevitable que los usuarios relacionen contratos entre si. En este sentido se incorpora el concepto de Paquete de Contratos *(ContractSet)* como un agrupamiento lógico de Contratos que comparten algunas características o están relacionados por algún motivo.

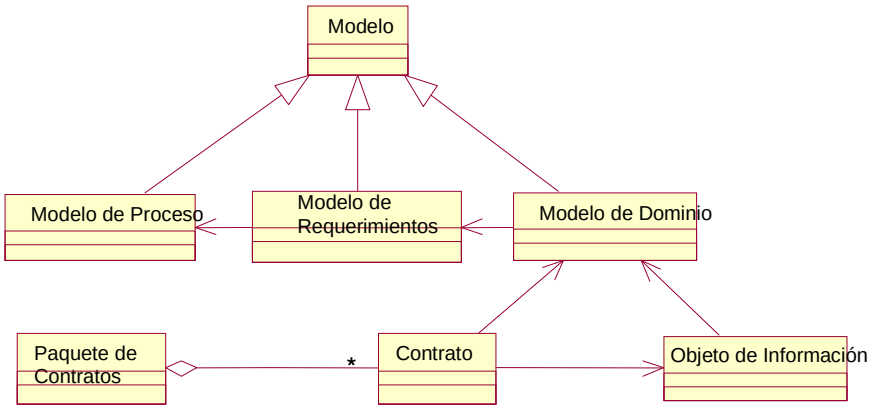


Figura 15 – Dependencias de Modelos (NGOSS)

2.4.4 El modelo "Arquitectural"

El enfoque combina conceptos vinculados al desarrollo de software basado en componentes, técnicas MBD *(Model Based Development)* de TINA-C, OMG, DMTF y TM Forum a los efectos de formular "guías" para el desarrollo de modelos que requieren ser intercambiados entre los actores involucrados en el desarrollo de componentes de software y los sistemas de gestión en que se van a usar[15].

La estructura de modelos definida en el modelo "Arquitectural" esta gobernada por el análisis de interacciones entre los actores involucrados en el desarrollo de sistemas de gestión basados en componentes.

Estos "actores" son:

- Cuerpos de Estándares *(Standard Bodies)*, generan estándares, como resultados de acuerdos en el ámbito industrial.
- ISV *(Independent Software Vendors)*, los cuales producen y comercializan componentes de software.
- Integradores de Sistemas *(System Integrators)*, los cuales producen sistemas de gestión construidos utilizando componentes propietarios y de terceros (Múltiples ISV).
- Proveedores de Servicios *(Service Providers)*, usuarios de los sistemas de gestión (OSS), son los operadores. Son los que poseen los requerimientos del negocio.

El modelo "Arquitectural" captura la estructura de los modelos que se requieren intercambiar entre los distintos actores en el desarrollo de componentes, sistemas y estándares. Esto incluye la identificación de relaciones entre elementos de los modelos que deben ser gestionados desde el propio desarrollo del modelo, a través de su liberación, uso y reuso. El modelo arquitectural brinda guías para la formulación de los modelos, apoyándose en el uso de UML y XML, siguiendo pautas del MDA de OMG.

2.4.4.1 Estructura del modelo "Arquitectural"

Un componente de software se modela como un *Building Block*, el cual ofrece una o más interfaces llamadas contratos. Los contratos se empaquetan con los requerimientos del negocio, modelos de análisis del sistema y modelos de información explícita que se pasa a través del contrato.

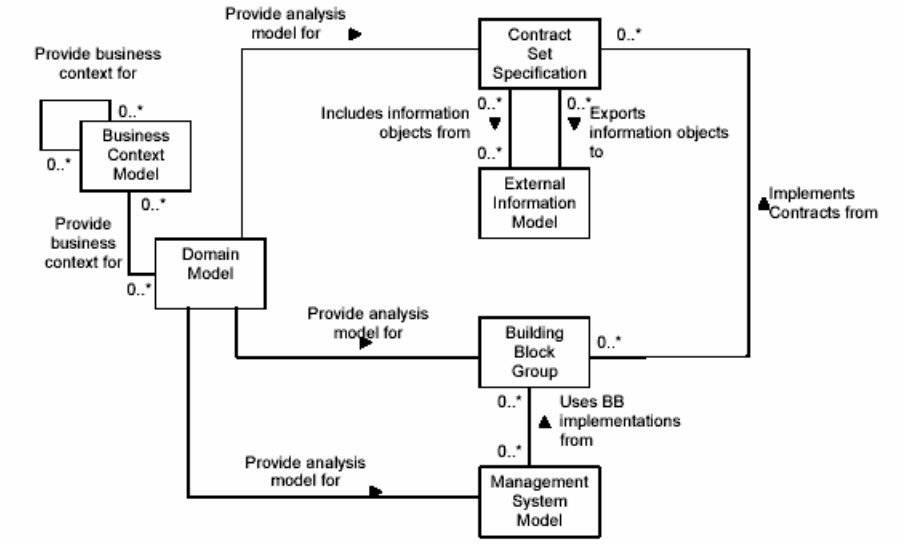


Figura 16 – Modelo Arquitectural

2.4.4.1.1 Modelo del Contexto del Negocio *(Business Context Model, BCM)*

Captura los requerimientos bajo un dominio de interés y modela aspectos de entorno del negocio sobre los cuales opera el dominio.

Modelos Utilizados:

Requerimientos. Colección de requerimientos impuestos por el dominio de interés.
Modelo de Roles del Negocio. Representa los distintos tipos de organizaciones presentes en el dominio de interés y su entorno a nivel del negocio y de "Puntos de Referencia", cada uno de los cuales representa un posible conjunto de interacciones que puede existir entre un par de Roles del Negocio *(Business Roles)* (basado en el modelo de TINA).Los Roles pueden representarse como clases de un diagrama UML y los "Puntos de Referencia" se representan como asociaciones entre dichas clases.
Modelo de Organización del Negocio. Modelo que captura el escenario general del negocio que recoge las organizaciones típicas y usuarios que caracterizan el dominio de interés y su entorno. Las organizaciones manejan uno o más "Roles del Negocio" (del modelo de Roles visto anteriormente), de manera que las Organizaciones pueden ser vistas como instancias de **Roles del negocio**. En este sentido se pueden representar como instancias de objetos en el diagrama UML.
Modelo de Casos de Uso del Negocio. Es una expresión de la funcionalidad observada del dominio de interés por un conjunto de Actores del Negocio, los cuales pueden incluir Organizaciones y usuarios del modelo de Organización del Negocio que son externos al dominio de interés.
Modelo de Procesos del Negocio. Es una expresión de los procesos del negocio que pueden operar en el dominio de interés (como lo ha hecho TMForum a través del eTOM). El modelo contempla áreas de procesos, flujos de procesos extremo a extremo, modelados como diagramas de actividad UML, identificando actividades específicas en las distintas áreas de procesos que están encadenadas para realizar un caso de uso.

2.4.4.1.2 Modelo del Dominio *(Domain Model, DM)*

Se realiza el análisis de requerimientos y entorno del negocio capturado por el Modelo del Contexto del Negocio (Business Context Model). Provee una expresión detallada a nivel de sistema del entorno con el cual interactúa el dominio de interés y la funcionalidad que debe exhibir a dicho entorno.
Se provee análisis de requerimientos para una especificación de un "Grupo de Contratos", un grupo de Building Blocks, un Sistema de Gestión o una combinación de estos.

El modelo consiste de los siguientes sub elementos:
Modelos de Casos de Uso del Dominio y Modelo de Procesos del Dominio, los cuales son refinamientos de los correspondientes modelos a nivel del Modelo de Contexto del Negocio.
Modelo de Análisis del Dominio. Se deriva de los anteriores. Se usa el "pattern" de diseño "model-view-controller" para proveer modelos estáticos y dinámicos de la estructura lógica del dominio. Estos se modelan usando diagramas de clases UML, con estereotipos "boundary", "control" y "entity" complementados con diagramas de interacciones que muestran las interacciones entre instancias de estas clases requeridas para llevar a cabo los casos de uso del dominio[13].

2.4.4.1.3 Especificación de "Grupo de Contratos" *(Contract Set Specification, CSS)*

Especifica una o más especificaciones de interfaces relacionadas que pueden ser usadas en la implementación de contratos a nivel de *Building Blocks*. Permite generar especificaciones de contratos independientemente de los *Building Blocks*, de esta manera tratando de que la especificación pueda ser reusada en diferentes implementaciones de Building Blocks.

Contiene un conjunto de especificaciones de contratos y un Modelo de Información del Grupo de Contratos *(Contract Set Information Model, CSIM)*. Las especificaciones de contratos se derivan del Modelo del Dominio. El modelo de información es la agregación de los modelos de información pasada a través de los contratos individuales. El modelo de Información del grupo de contratos puede referenciar objetos en un modelo de Información externa. El modelo puede derivarse de entidades del Modelo de Análisis del dominio.

Se requieren lenguajes de especificación de Contratos en forma independiente de la tecnología *(Technology Neutral Contract Specification, TNCS)* que puedan captar los modelos de interoperabilidad

en términos neutrales. Este lenguaje debe soportar una transformación determinística a un lenguaje de especificación de contratos específico a una tecnología *(Technology Specific Contract Specification, TSCS)* resultando por ejemplo en especificaciones IDL o GDMO.

Como un primer paso para el diseño de un lenguaje TNCS, el modelo "Arquitectural" ha definido un formato neutral para la parte del modelo de información de un Contrato, la cual puede ser usada para el CSIM. Esto está originado en el meta-esquema definido en el CIM de DMTF, pero simplificado a los efectos de definir modelos de información compartida en UML.

2.4.4.1.4 Modelo de Información Externa *(External Information Model, EIM)*

Expresa la información que puede ser pasada a través de contratos, pero documentada en forma separada de las especificaciones de contratos de manera de posibilitar el reuso de especificaciones de información contenida a lo largo de distintos diseños de contratos.

El uso progresivo conduce a una población federada de Modelos de Información externa originados en distintas fuentes, permitiendo la comparación entre ellos y cuando sea apropiada la integración de los mismos.

2.4.4.1.5 Grupo de Building-Blocks *(Building Block Group, BBG)*

Se expresan para cada uno de los Building-Blocks, los contratos soportados por el BB, otros Contratos sobre los cuales se apoya el BB, y el comportamiento visible externamente. Incluye el propio software (deployable) del BB.
Uno de los propósitos de la existencia de este grupo, constituye la posibilidad de reusar BBs en el desarrollo de distintos Sistemas de Gestión.

A nivel del BBG, cada *Building Block* está caracterizado por un Descriptor de *Building Block* que provee de la información requerida para la instalación del BB tales como detalles propios de la plataforma especifica considerada y propiedades que modifican el comportamiento en tiempo de ejecución. Referencia al propio software del BB y la especificación del contrato soportada.

2.4.4.1.6 Modelo del Sistema de Gestión *(Management System Model, MSM)*

Expresa el diseño e implementación de un sistema concreto de gestión que realiza actividades OSS típicas. Constituye el desarrollo de un único Integrador de Sistemas.

2.4.4.2 Uso del modelo Arquitectural

Desde la perspectiva de un Proveedor de Servicios, éste especifica "Requerimientos del Negocio" (en términos de BCM) y especificaciones de interfaces especificas al Integrador de Sistemas para resolverlos. El Proveedor de Servicios puede optar por expresar su análisis de los requerimientos del negocio en términos derivados de BCMs estándar disponibles, de manera de mejorar el entendimiento de los requerimientos y facilitar la adaptación a soluciones existentes. Las especificaciones de Interfaces del Sistema también pueden ser tomadas de Estándares Existentes. Un Proveedor de Servicios necesita intercambiar Requerimientos del Negocio y especificaciones de Interfaces del Sistema con otros Proveedores de Servicios cuando se ensamblan requerimientos para un Sistema de Gestión que interactúa con aquellos de otros Proveedores de Servicios.

Los Integradores de Servicios intentarán proveer sus Sistemas de Gestión a la máxima cantidad de Proveedores de Servicio de manera de maximizar ventas. Al hacer eso, harán uso también de BB obtenidos de distintos ISVs de manera de obtener la funcionalidad a un precio óptimo y calidad. Los Integradores de Sistemas pueden intentar alinear los requerimientos de los Proveedores de Servicios usando BCMs construidos por "Cuerpos de Estándares". Esto ayuda en el diseño de posibles soluciones utilizando Especificaciones de Contratos y EIM estándares y así incrementar la posibilidad de encontrar algún BB de algún ISV que implementa esos estándares.

Los ISV intentarán proveer BBG a un amplio rango de Integradores de Sistemas. ISV pueden intentar usar los BCMs estándar disponibles en el desarrollo de nuevos BBG de manera de poder implementar Contratos Estandarizados, los cuales pueden ser usados en productos basados en BB. Los ISV pueden publicar sus CCS y EIMs para estimular el uso de sus productos.

2.4.5 Conclusiones

Son valiosos los aportes del TMForum en cuanto a estandarizar requerimientos y modelos en términos neutrales. El eTOM [38], brinda modelos de procesos del negocio genéricos en el ámbito de las telecomunicaciones. El SID define un modelo de datos e información compartida entre distintos dominios de gestión de redes y servicios de las telecomunicaciones.

El TMForum ha estandarizado requerimientos y modelos en términos neutrales. Primero capturando modelos de procesos del negocio genéricos en el dominio de las telecomunicaciones, expuestos a través del eTOM. Actualmente el TMForum esta en proceso de estandarizar interfaces de componentes y la información que circula por dichas interfaces. Estos trabajos se están llevando a cabo en el ámbito de la iniciativa NGOSS.

Es destacar sin embargo que estos trabajos se están manejando utilizando conceptos arquitectónicos de alto nivel. Hay pocas pautas para el desarrollo e integración de *Building Blocks*, o por otro lado guías para reconciliar elementos como Procesos del Negocio, Especificaciones de Contratos y Modelo de Información compartida que han sido estandarizados por separado.

El Modelo Arquitectural muestra una visión valiosa en lo que tiene que ver con el modelado en términos neutrales de componentes de gestión interoperables. Intenta proveer un conjunto de guías de diseño y modelado que posibilite el intercambio y comprensión de modelos entre las distintas organizaciones involucradas en el desarrollo de OSS basado en componentes integrados.

Un lenguaje se requiere aún para definir los contratos en términos neutrales *(Technology Neutral Contract Specification, TNCS)*. WSDL *(Web Service Description Language)* ofrece una solución promisoria a investigar. Usa XML para definir interfaces en forma independiente del mapeo a interfaces que responden a protocolos específicos.

Es de destacar que no se ha intentado el uso de este esquema para las interfaces gestor-agente todavía, esto podría requerir un encare más estructurado para definir el contenido de la información de las interfaces WSDL, actualmente expresadas usando Schemas XML , los cuales se requerirían que fueran compatibles con la expresión orientada a objetos de los modelos de Información del modelo Arquitectural.

2.5 Implementación y Elecciones Tecnológicas Posibles

2.5.1 Necesidad de Integrar múltiples tecnologías

La tendencia actual a una evolución cada vez más rápida de los servicios de comunicación y tecnologías asociadas, requiere de una estrategia conjunta de los vendedores de infraestructura de red y de software de gestión de redes, para permitir a los proveedores de servicios adaptarse a los requerimientos y necesidades existentes. En la actualidad no existe ninguna estrategia que cubra estas necesidades, sino que existen muchas estrategias eventualmente inconsistentes entre sí que no cubren todas las necesidades.

Los proveedores de servicios deben disponer de sistemas que posibiliten la creación de nuevos servicios y permitan la incorporación de nuevos elementos de red, facilitando su creación, mantenimiento y configuración.

Este problema admite varias soluciones. Algunas se centran en la construcción de un marco de trabajo "ideal" que intentan definir todos los capas que lo integran, las interfaces existentes entre ellos y las tecnologías que se utilizan para su implementación, para luego recién construir las aplicaciones que están sobre el mismo. En general estas estrategias no se adecuan a las necesidades actuales ya que el marco de trabajo tiene un alto tiempo de desarrollo que lleva a que cuando se termina su construcción, los requerimientos y las tecnologías básicas ya caducaron.

Una solución que se adecua más a las necesidades actuales es el desarrollar un marco de trabajo centrado en las aplicaciones de los proveedores. En lugar de esperar a automatizar servicios básicos en forma masiva, se apunta a resolver rápidamente servicios que tengan que ver con el cliente en forma directa. El estado del arte actual, es que han aparecido muchas tecnologías que facilitan la creación de aplicaciones de alto nivel que operan sobre motores de workflow, fáciles de crear y mantener. Es posible que puedan quedar procesos de bajo nivel no automatizados en primera instancia, pero esto no debe ser obstáculo para utilizar las aplicaciones disponibles de los proveedores. Paulatinamente, se irán automatizando e incorporando los servicios faltantes.

Una primera visión sobre un marco de trabajo con dichas características, basado en la arquitectura TMN / TOM, es el que se muestra en la Figura 17. Esta estrategia planea llegar a un acuerdo para que los proveedores de software construyan software específico y especializado para un nivel (red, servicio o presentación). El software desarrollado debe permitir un alto grado de control desde el exterior del mismo a través de interfaces bien definidas y documentadas, de forma de poderlos integrar al marco de trabajo fácil y rápidamente.

Esta estrategia debe satisfacer todas las necesidades de *Fulfillment* (pedido de servicio, configuración, aprovisionamiento y activación), *Assurance* (disponibilidad, performace, tiempos de respuesta y cumplimiento del SLA) y *Billing.*

Este marco de trabajo debe de tener determinadas características (propuesta *"Implementing a Management System Architecture Framework"* [7] de Lucent):

- La interfaz front-end con el cliente basada en tecnologías Web. La salida debería ser preferentemente en XML permitiendo interfaces flexibles que pueden ser visualizadas de diferentes formas y por diferentes dispositivos. En la Figura 17 esto está representado por el portal i-TakeCare que resuelve el *fullfillment*, *assurance* y *billing*.
- Se sugiere como un mecanismo posible de autenticación de los usuarios con el portal utilizado LDAP *(Lightweight Directory Access Protocol)*, determinando así los recursos y aplicaciones a las que tiene acceso cada usuario. Este portal lo utilizan tanto clientes como vendedores y operadores, para realizar sus respectivas tareas.
- Cada aplicación a integrar en este marco de trabajo debe ser encapsulada dentro de un nuevo componente para resolver la interacción con los buses a los que acceda.

- La interoperabilidad entre las aplicaciones de las capas de Servicio y de Red, se resuelve por medio del llamado "Service Workflow-Oriented Bus". El mismo esta orientado a servicios y utiliza un motor de workflow para hacer el seguimiento de los distintos procesos (incidente, configuración, etc.). Debido a las necesidades de escalabilidad y flexibilidad requeridas, se sugiere el uso de EJBs *(Enterprise Java Beans)* como tecnología para implementarlo. En este sentido puede optarse por desarrollar enteramente dicho motor con esta tecnología, o tomar un middleware y un motor de workflow propietario (de algún proveedor) y encapsularlo dentro de EJB para que soporte, futuras aplicaciones de administración. El acceso a las bases de datos de clientes y de servicios se hace a través de éste mismo bus utilizando JDBC o ODBC (este caso para fuentes de datos legadas).

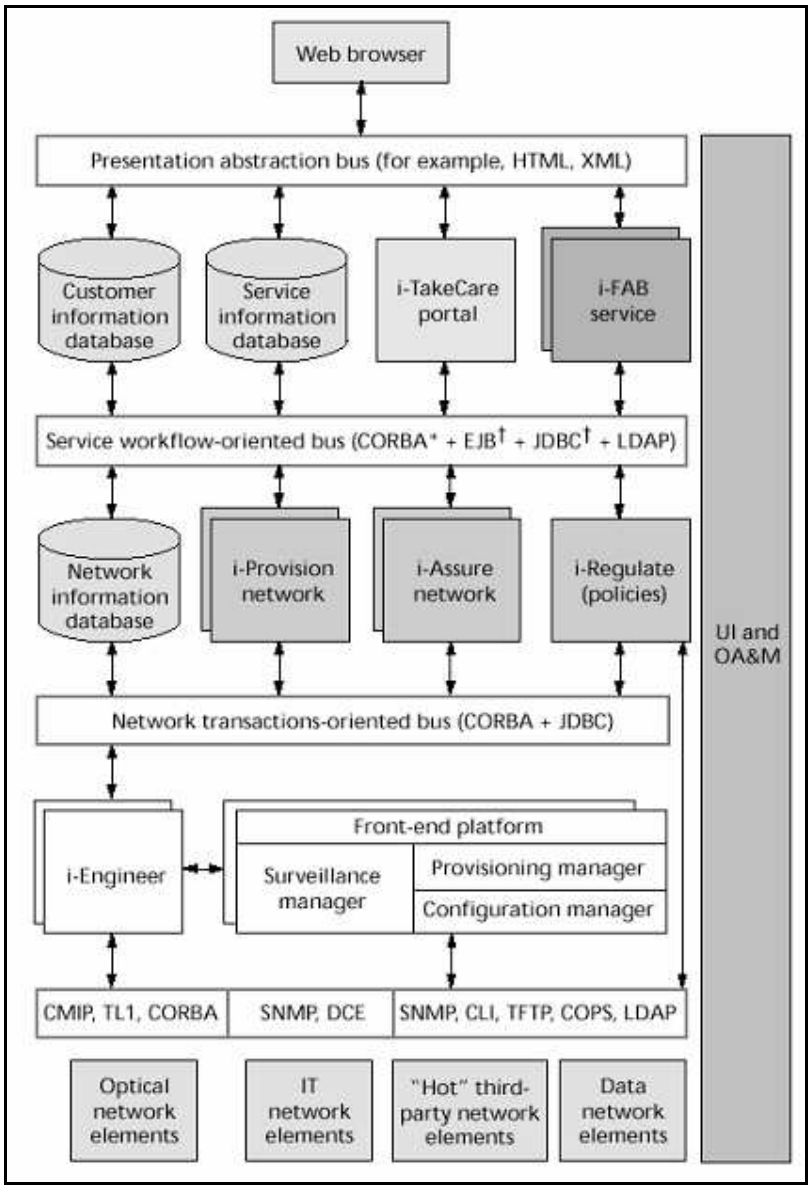


Figura 17 - Arquitectura de marco de trabajo centrada en las aplicaciones.

- De la capa de Red hacia abajo se encuentran las aplicaciones que implementan las funcionalidades de FCAPS[6], para el manejo de los elementos de red. Debido a la alta variación

[6] FCAPS es el acrónimo de Fault, Configuration, Accounting, Performance, Security

de los elementos de red, estas aplicaciones requieren de un bus estándar para poder simplificar su manejo. Este bus es llamado "Network Transactions-Oriented Bus", y debido a la naturaleza de las interacciones (comunicaciones de 2 partes) entre los participantes del mismo, se eligió CORBA para definir las interfaces debido a lo estándar y estable de ésta tecnología, y su alta difusión entre los proveedores de empresas de telecomunicaciones.

- En lo relativo con la comunicación entre la capa de administración de los elementos de red y los propios elementos, se recomienda el uso de SNMP *(Simple Network Management Protocol)*, CLI y TFTP *(Trivial File Transfer Protocol)*, mientras que se descarta CMIP/Q3 y TL1 que paulatinamente se están dejando de usar.
- Utiliza tres modelos de datos: el de Información de Cliente, el de Información de Servicios y el de Información de Red.
- Se proveen un conjunto de herramientas para la administración y mantenimiento común de todas las aplicaciones a todos los niveles que se encapsula en un bloque llamado "Operations, Administration and Maintenance" (OA&M).

2.5.2 Propuesta OSS/J

OSS/J es una alianza de vendedores y desarrolladores (liderado por Sun Microsystems) que definen APIs basadas en J2EE para ser usadas por los Integradores de Sistemas de Gestión de Telecomunicaciones.

Esta iniciativa define un entorno de trabajo (a través de API estandarizadas) de manera de posibilitar una plataforma concreta para la implementación de componentes que brindan servicios aplicados a la gestión de telecomunicaciones [35].

Definen un conjunto de servicios comunes propios de la especificación y se estandarizan interfaces para algunos módulos del negocio. Particularmente en las áreas de "Trouble Ticketing", "Billing" y "Quality of Service", "Provision de Servicios".Esta propuesta se apoya fundamentalemente en el uso de la plataforma EJB/J2EE.

Los servicios se implementan como componentes EJB *(Enterprise Java Beans)* [30] que proveen:

- Interfaces del Negocio de Grano Grueso:
- Clustering, Scalability y "Fail Over" a través del uso de Application Server
- Mensajería para minimizar acoplamiento
- Uso de conectores JCA para la integración de sistemas legados

El concepto de *Building Block*, es un concepto clave para OSS/J. Es una colección de componentes empaquetados a los efectos de posibilitar su "deployment" y que responden a una necesidad (o conjunto de necesidades) concretas del negocio. Un Building Block OSS/J está compuesto de :

- EJB Session Beans with Java Value Types (JVT)
- Message Driven Beans
- EJB Entity Beans

Un OSS/J Building Block es un componente de software (en el sentido amplio) responsable de la gestión de entidades de telecomunicaciones tradicionales gestionadas u objetos del negocio de las telecomunicaciones en general.

En la propuesta OSS/J una entidad gestionada u objeto del negocio se representa como un EJB EntityBean un Objeto JVT (Java Value Type) o un Documento XML (XVT Document). La funcionalidad se implementa dentro de cada componente, soportando las interfaces definidas a nivel de contratos que representan la lógica visible externamente. Por ejemplo, las interfaces OSS/J son EJB Session Beans que brindan una visión (de grano grueso) de las entidades gestionadas.

2.5.3 Algunas aplicaciones comerciales disponibles

Empresa	Producto	Caracteristicas
IP VALUE www.ip-value.de	PREMIOSS	Service Provisioning.
INTRACOM www.intracom.com	Janction Service Activator	Service Provisioning. Servicio transaccional basado en java para la activación de servicios. Construido sobre el microkernel JMX, permite la extensión dinámica de su motor usando adaptadores que encapsulan el conocimiento de "Provisioning Service" en distintos tipos de redes usando distintos tipos de tecnologías.
DIGITAL FUEL www.digitalfuel.com	Service Flow 3.0	Quality of Service API.
IBM www.ibm.com/services	Mobile Workforce Management OSS/J Solution	aTrouble Ticketing, Qualty of Service/SLA, Service Activation. Soporta monitoreo de fallos en la red. Se integra con aplicaciones de gestión de movilidad.Basado en estándares J2EE y OSS/J
ILOG www.ilog.com	ILOG JTGO	Suite de componentes OSS. Quality of Service APIs
METASOLV www.metasolv.com	Automated Service Activation Program ASAP 4.6 Order Management System OMS 2.5	Plataforma dirigida a proveedores de servicios de telecomunicaciones para "Service Activation". El sistema ASAP *(Automated Service Activation Program)* recibe solicitudes de servicio desde diferentes orígenes y transmite la información de activación a cualquier dispositivo de la red. ASAP se puede comunicar con distintas tecnologías y redes heterogéneas. Soporta el estándar OSS/J, Service Activation.
NOKIA www.nokia.com	NOKIA NetAct	OSS Service Activation,OSS Quality of Service, OSS Trouble Ticket, OSS Inventory
HERSCHEL TECH. www.herscheltechnologies.com	Herschel Integration Suite (HIS)	Framework de Integracion. Unifica J2EE y CORBA a nivel de servicios y mensajeria.Es capaz de integrar sistemas de gestión basados en CORBA al sistema de Trouble Ticketing apoyado en OSS/J
VERTEL www.vertel.com	M*Ware Ticket Exchange, 6.2	Provee conectividad a través de interfaces de la API OSS Trouble Ticketing con varios vendedores de sistemas de Trouble Ticketing permitiendo transacciones entre sistemas legados y aplicaciones OSSJ/TT.

2.6 Reflexiones sobre el estado del Arte

2.6.1 Multidimensionalidad del problema de la gestión

El foco de gestión de las redes tradicionales ha sido a nivel del elemento, en relación con sus protocolos de comunicaciones y sus Bases de Información de Gestión (MIB).

Cuando la red crece en tamaño y los servicios son más complejos, los tópicos de escalabilidad y de gestión de ella se vuelven más importantes.

Los nuevos requerimientos existentes en la actualidad como la necesidad de definir y gestionar servicios rápidamente sobre la infraestructura de red, han creado requerimientos de técnicas más sofisticadas de gestión de red. Estas técnicas se deben extender también al area de servicios por lo que el problema se vuelve cada vez más complejo.

La enumeración estructurada de funciones de TMN ha sido de gran utilidad para permitir la identificación de las necesidades de gestión al implantar nuevos servicios. Los distintos modelos y esquemas de clasificación TMN se han constituido en elementos valiosos para comunicar requerimientos e información entre distintos actores involucrados en la puesta en marcha y funcionamiento de los nuevos sistemas de gestión e ilustran de manera clara la complejidad de un sistema de gestión.

La gestión de redes y servicios en la industria de telecomunicaciones se encuentra en medio de un espacio de *n*-dimensiones con sus reglas que tiene que resolver. En este sentido se puede enfocar el problema desde distintos puntos de vista como ser:

Funcional. Se definen actividades que hay que realizar y la organización de las mismas.
De información. Se modela la información de gestión que se intercambia entre el gestor y el agente. Este modelo depende de las funciones que se realicen y de los recursos que se quieran gestionar.
De comunicación. Se especifican protocolos de comunicaciones utilizados para el intercambio de información entre sistemas.
De estratificación lógica. Se identifican en la empresa de telecomunicaciones distintos estratos de responsabilidad de las actividades (que surgen de la propia estratificación lógica de la gestión y la de las capas jerárquicas organizacionales de la empresa).

Además, dentro de cada dimensión se puede establecer otra clasificación en asuntos, áreas de interés, responsabilidades, granularidad, etc. En síntesis, se pueden identificar pequeñas porciones del espacio *n*-dimensional, susceptibles de ser organizadas y resueltas en forma independiente, pudiendo encapsular el comportamiento de las mismas en Bloques Constructivos (Building Blocks). Esto va acompañado de la definición de interfaces normalizadas entre Building Blocks de manera que los flujos de trabajo y las inter-relaciones tengan una organización coherente.

Se intenta que visión del negocio sea el elemento guía para organizar las relaciones entre los Building Blocks, identificando aquellas actividades que aportan valor, organizando la (las) cadena(s) de valor de la empresa.

El conjunto de actividades organizadas permiten un incremento de la confiabilidad, la seguridad y los procesos del flujo a través en la implantación de servicios, posibilitando una mejor calidad de servicio brindada al cliente. También se reducen los costos al optimizar las distintas actividades con un criterio de organización uniforme[5].

2.6.2 Desarrollo basado en Modelos.

La mayoría de los enfoques anteriormente presentados resaltan la necesidad de poder diseñar sistemas inicialmente en términos neutrales (despojándose de consideraciones tecnológicas prematuras). Esta

separación asegura la validez y continuidad de las especificaciones a lo largo del tiempo, aún cuando ocurran cambios tecnológicos.

Los sistemas lógicos se construyen definiendo los servicios requeridos a través de definiciones de Contratos (en términos neutrales). Las especificaciones de los contratos son el producto del trabajo de distintos actores que formulan modelos (utilizando distintas herramientas y manejando distintos niveles de detalle). En general el análisis parte del enfoque de requerimientos y procesos que se llevan a cabo, derivando en la definición de servicios requeridos y un modelo de información asociado. El modelo de información está íntimamente asociado a las especificaciones de servicios en el sentido de que la descripción de los mismos solo es posible hacerlo sobre la base de "Tipos de datos" de base adecuados. Es común agrupar el análisis de servicios que mantienen características comunes o están relacionados por una razón a nivel de "Dominios".

Tanto TINA-C como TMForum prestan una ayuda muy importante en la definición inicial de modelos. El TOM *(Telecom Operations Map)* del TMForum constituye un modelo de procesos de alto nivel para la gestión de telecomunicaciones que puede ser utilizado como referencia inicial.

TMForum está en el proceso de definir modelos de información y un conjunto de "Contratos" requeridos (atendiendo a los requerimientos del TOM).

2.6.3 Implementaciones y Visión tecnológica.

Del relevamiento realizado surgen nuevos requerimientos para los proveedores de servicios para sus sistemas de gestión. Las nuevas necesidades tienen que ver con:

- *Distribución.* Las aplicaciones de gestión son esencialmente distribuidas dada la propia naturaleza de distribución geográfica de los recursos gestionados.
- *Escalabilidad*. Los recursos de red crecen en cantidad y complejidad.
- *Diversidad.* La plataforma de gestión debe ser capaz de trabajar con la diversidad de recursos y sistemas operativos que existen en un dominio de gestión.
- *Prestaciones*. Los modelos de gestión incluyen, cada vez más, objetos gestionados que las aplicaciones deben manejar adecuadamente y sin menoscabo de sus prestaciones.
- *Fiabilidad*. Los sistemas de gestión deben ser capaces de trabajar las 24 horas del día.
- *Integración de servicios*. Los sistemas deben ofrecer facilidades para la integración y creación de nuevos servicios.
- *Interoperabilidad*. Se logra a través de estándares de gestión, que permita automatizar procesos internos como la colaboración entre operadores, proveedores y clientes.
- *Acceso a la información*. Los usuarios implicados en la gestión deben poder acceder a la información que necesitan (con las consideraciones de seguridad del caso).

La elección de las tecnologías debe satisfacer los requerimientos anteriores. Actualmente existen varias alternativas como por ejemplo CORBA, Servidores de Aplicaciones, DCOM[25], WebServices[47], etc.

La alternativa de uso de servidores de aplicaciones (cuando sea aplicable) es recomendable por las siguientes razones:

- *Tendencia general*. El uso de Servidores de Aplicaciones resulta ser un factor común en las recomendaciones de diferentes actores del medio de Telecomunicaciones (por ejemplo: TMForum y Lucent).
- *Productividad y calidad en el desarrollo.* Los Servidores de Aplicaciones aprovechan, y de alguna forma inducen, a diseños orientados a componentes, que luego puedan ser reutilizados o modificados en su implementación.
- *Escalabilidad y facilidad en la explotación.* Los Servidores de Aplicaciones son sistemas que pueden evolucionar en escala, tanto verticalmente (aumentando la potencia de una plataforma), como horizontalmente (generando múltiples servidores y balanceando carga). Estos cambios resultan fundamentalmente transparentes a los desarrollos realizados, lo cual facilita la puesta en producción de componentes de software, aún en plataformas distribuidas.
- *Reutilización de plataformas*. Los actuales productos de Servidores de Aplicaciones pueden ejecutar en diferentes plataformas (hardware, sistemas operativos), cubriendo desde plataformas

Intel hasta mainframes. Esta situación permite hacer un uso refinado de los recursos de cómputo de la empresa, aprovechando las plataformas existentes incluso en forma relativamente dinámica.

- *Estandarización de modelos*. Dentro de las familias existentes (CORBA, Microsoft DCOM, J2EE de Sun) existen estándares que permiten usar Servidores de Aplicaciones sin quedar atados a un producto de marca particular. Asimismo, existe una tendencia fuerte hacia la interoperación entre servidores de diferentes familias.

Resulta importante destacar que no es posible apoyarse en una única tecnología. Una de las razones principales es la existencia de "Sistemas Legados" preexistentes que no se pueden soslayar. Tampoco es posible reemplazar los mismos por el costo que implicaría, por lo que es necesario integrar estos sistemas (adaptación mediante) al framework integrado.

La elección de la tecnología a aplicar debe tener en consideración los requerimientos del servicio (que pueden variar dependiendo de la naturaleza de los mismos). Se deben tener en cuenta las siguientes propiedades:

Estabilidad Es una propiedad que se le exige a la gestión de los recursos de la red. Esta plataforma debe ser robusta dado que de ella depende la propia operación de la red.

Flexibilidad. Para la construcción de componentes de software que puedan articular nuevos servicios se requiere de tecnologías que posibiliten rápidos desarrollos y fácil mantenimiento. La implementación de servicios vinculados al soporte de procesos de alto nivel, más cercano al cliente, tienen características más dinámicas donde los cambios requeridos son más frecuentes. En estos casos la flexibilidad de la plataforma elegida es esencial.

En este sentido tecnologías como CORBA tienen características de madurez y estabilidad que la hacen apropiada para la gestión de elementos de red y sus sistemas de gestión. Por el contrario parece inadecuada para la construcción de componentes que soporten la gestión de servicios y procesos de alto nivel (sometidos a requerimientos cambiantes y exigencias de obtención de resultados en forma rápida). Para este último caso las tecnologías de servidores de aplicaciones (J2EE, WebServices, etc.) parecen más apropiadas.

En síntesis, la construcción de sistemas basados en componentes, necesariamente deberá realizarse en el marco de una arquitectura que posibilite la interoperabilidad entre varias plataformas simultáneamente.

2.6.4 Construcción de sistemas utilizando componentes comerciales estandarizados

NGOSS de TMForum brinda pautas para definir componentes a los efectos de posibilitar la construcción de OSS en base a integración de componentes. En este marco funcionan los "Proyectos Catalizadores" que son emprendimientos entre Proveedores de Servicios e Integradores de Sistemas para demostrar los principios y pautas NGOSS en aplicaciones reales.

La propuesta OSS/J (Sun Microsystems) basada en Java complementa el enfoque NGOSS en el sentido que se provee de código de referencia para APIs para ser usados en las áreas de "Trouble Ticketing", "Billing" , "Quality of Service" y "Provision de Servicios".

La iniciativa NGOSS intenta crear un nuevo framework para la integración de procesos del negocio de empresas de telecomunicaciones, procurando identificar y definir elementos que puedan ser compartidos entre distintos sistemas.

Este enfoque es valioso para la integración de OSS, especialmente el modelo de datos e información compartida *(Shared Information and Data, SID)* que está aún en desarrollo [39]. Es frecuente que muchas de las ineficiencias de los proveedores de servicios se deban al acarreo de información fragmentaria entre distintos componentes. El resto de la información no se comparte directamente entre aplicaciones en general (a menos que todos los datos sean exportados y reunidos en una base de datos de la empresa). Es frecuente además que proveedores de servicio de gran porte mantengan comportamientos estancos dentro

de su compleja estructura organizacional, existiendo departamentos poco propensos a compartir datos. La falta de un modelo de información en sí es costosa en el sentido de que al no disponer de componentes de software reusable, se requiere de código hecho a medida para cada proyecto [28].

El SID *(Shared Information and Data)* de TMForum intenta resolver este tipo de problemas, sin embargo el problema central y el éxito de este enfoque está asociado al éxito de la adopción de este esquema por un numero importante de Proveedores de Servicios, fundamentalmente los más importantes, de manera que el negocio de la construcción de componentes estandarizados sea beneficioso y pueda prosperar [39].

Según la empresa consultora RHK [28] , no está claro aún que va a pasar con la adopción de estándares NGOSS por parte de los proveedores de servicios ya que se observan distintos comportamientos.
Es claro que la mayoría de ellos participa en los "Proyectos Catalizadores" de TMForum, pero esto no implica a priori una definición sobre la adopción de los mismos. Por ejemplo la empresa "Verizon" tiene una visión positiva respecto al tema, pero en los hechos, la mayoría de sus propios aportes no se usan en aplicaciones reales. En el caso de Telcordia, se presenta difícil para la propia empresa el romper la dependencia con su sistema OSMINE *(operations system modifications for the integration of network elements)*. KPN Telecom sin embargo, aborda la problemática en forma distinta al utilizar el enfoque neutral para la interconexión entre OSS y liderar el camino para el uso de gatways bi-direccionales entre dos OSS pertenecientes a distintos proveedores de servicios.

En cuanto a la plataforma OSS/J, permite bajar los costos de integración al proveer un conjunto de interfaces (API) con código fuente (gratis) para implementaciones de referencia y herramientas para verificar compatibilidad con el producto. Detrás de OSS/J se encuentran empresas como BEA, BT, IBM, MEtasolv, Motorota, NEC, Nortel, Nokia, Sun y Telcordia.

Se espera que se expanda la base de OSS/J de alrededor de 30 miembros en el 2002 a más de 100 en un futuro cercano. J2EE aparece como una tecnología ampliamente aceptada por Proveedores y Service Providers por lo que OSS/J tiene un potencial de aceptación importante.

Para que tengan éxito tanto NGOSS como OSS/J es fundamental que tengan el respaldo de los proveedores de servicios mayores. Si no se da así en el corto plazo, el modelo de datos estandarizado tomará algunos años más en madurar y ser soportado por el software comercial.

Capítulo 3

3 Arquitectura Propuesta para interconexión de Building Blocks

3.1 Objetivo

A lo largo del capítulo anterior se ha presentado la idea de construcción de sistemas utilizando un conjunto de *Building Blocks* débilmente acoplados entre sí, donde cada uno de ellos implementa una parte del comportamiento requerido por la propia lógica del negocio, brindando servicios a otros Building Blocks y utilizando eventualmente servicios disponibles brindados por terceros.

En este sentido cada Building Block está constituido por un agrupamiento de implementaciones de servicios que llevan a cabo procesos específicos. Los servicios brindados están caracterizados por "Contratos" donde se especifican las características de las interfases provistas y la forma en que se deben usar. Desde el punto de vista de implementación, el Building Block constituye una unidad indivisible a los efectos de su instalación y puesta en marcha (*deployment*).

El concepto de *Building Block* no es implementable en forma aislada (ver 2.3.2.3). La idea de construcción de software basado en Building Blocks requiere de la existencia de un entorno apropiado para soportarlo.

El objetivo, en este sentido, es definir una arquitectura que posibilite la interconexión y el funcionamiento de los Building Blocks.

Se pretende abordar el problema en una forma independiente de la tecnología. Se toma como referencia las propuestas de TMForum de NGOSS[40] y GB909[41] reseñada anteriormente. La arquitectura se basa en los siguientes principios fundamentales:

- Separación de los procesos del negocio de los componentes de software.
- Existencia de un Vehículo de Comunicaciones Común.
- Existencia de Modelo de Información Común.

La idea central parte de desacoplar el vínculo existente entre la lógica que gobierna los procesos y su implementación en Building Blocks. La lógica de los distintos procesos debería estar descrita en un lenguaje neutral y servir de entrada a un "Gestor de Procesos" (por ejemplo un motor de Workflow) que oficia como ordenador de las actividades dentro del sistema. Este gestor de procesos se apoya en servicios definidos por componentes a lo largo del sistema.

Los servicios brindados por los Building Blocks son accesibles al resto del sistema a través de un vehículo de comunicaciones común. Es fundamental en este esquema de interoperabilidad la existencia de un modelo de información común que permita la definición de contratos (empleando "tipos de datos" del sistema) de manera que se pueda hablar un lenguaje común entre las distintos actores que participan.

3.2 Motivación y Alcance de la propuesta

Dada la abundancia de soluciones que permiten generar esquemas interoperables y la necesidad de evaluarlas a través de sistemas concretos es que surge la necesidad de disponer de una herramienta de trabajo que achique la distancia entre un diseño lógico neutral y las múltiples implementaciones que se puedan llegar a llevar a cabo.

Es de gran valor el disponer de una plataforma tal que la implementación realizada pueda reubicarse en el contexto de otra tecnología e inclusive variar el patrón de interoperabilidad como por ejemplo interacción sincrónica o asincrónica.

Esto permitirá derivar rápidamente desde un diseño lógico en términos neutrales a implementaciones concretas. Inclusive experimentar con el sistema modificando la configuración simplemente, lo que permite el reuso de la implementación en forma casi directa.

En el presente trabajo se hacen algunas restricciones en lo que tiene que ver con la implementación. En principio, se plantea como plataforma de trabajo la plataforma Java.

En cuanto a tecnologías de interoperabilidad se abordan la arquitectura EJB y CORBA.

No está dentro del alcance del presente trabajo abordar la temática de la gestión de procesos ("Motor de Workflow"). El objetivo central está situado en la definición del vehículo de comunicaciones que permita la interoperabilidad entre las distintas entidades.

En el desarrollo siguiente, se expone inicialmente las características de la arquitectura en términos neutrales. Posteriormente se abordan las particularidades que surgen al aplicar la arquitectura a tecnologías específicas.

3.3 Arquitectura "Neutral".

Para caracterizar una arquitectura, es necesario definir los conceptos manejados por la propia arquitectura y las reglas que siguen dichos conceptos, además de brindar modelos donde se ilustren la aplicación de dichos conceptos y reglas.

En la siguiente sección se establecen definiciones básicas y principios seguidos por la arquitectura. En la sección 3.3.2 se expone una visión general de la arquitectura (a través de la formulación de un modelo conceptual) según distintos puntos de vista. Finalmente en la sección 3.3.3 se describen algunos servicios básicos brindados por la arquitectura.

3.3.1 Conceptos y principios básicos seguidos por la Arquitectura.

3.3.1.1 Interfaces definidas a través de "Contratos"

El sistema se caracteriza por el hecho de que todos los *Building Blocks* que proveen servicios (de manera que queden disponibles para otros Building Blocks) lo hacen a través de "Contratos".

Estos contratos tienen las siguientes características:

- Descripción del servicio provisto.
- Metadata usada para describir el Contrato. Se describe la interfase del mismo donde se indican sus métodos y tipos de datos empleados. Se utilizan diagramas UML de clases y un diccionario de datos que define la sintaxis, semántica y el uso de clases y atributos en general.
- Precondiciones bajo las cuales se brinda el servicio (son condiciones que se deben satisfacer a los efectos de poder invocar el servicio en forma adecuada). Se genera una excepción al intentar invocar un Contrato donde no se verifica la Precondición requerida. Se puede utilizar OCL *(Object Constraint Language)* para la formulación de las mismas.
- Poscondiciones. Definen el estado en que queda el sistema luego de invocar el servicio en forma exitosa (sin generar excepciones).
- Excepciones que puedan ser generadas. Una excepción se genera si se satisfacen las precondiciones, pero no se pueden alcanzar las poscondiciones.

En general los contratos se agrupan en dos categorías. El primer grupo está constituido por contratos que brindan servicios propios del negocio. El segundo grupo está constituido por contratos de servicios brindados por la propia arquitectura, que apuntan a dar soporte de operación e integración con el resto del sistema.

3.3.1.2 Vehículo de Comunicaciones Común

En sistema se caracteriza por la existencia de un servicio de comunicaciones (por ejemplo un "bus" de comunicaciones) o alguna forma que permita a los *Building Blocks* "interoperar" entre sí. Cada servicio de comunicaciones ofrece uno o más mecanismos de transporte. Es importante destacar que puede existir más de un servicio de comunicaciones dentro de una misma implementación y en estos casos cada servicio de comunicación representa distintos mapeos hacia tecnologías específicas.

3.3.1.3 Información compartida

El sistema en general se caracteriza por el uso de un único modelo de información común. Esto posibilita la definición de un conjunto de "Servicios de Información y Datos compartidos" (*Shared Information and Data, SID*) que permiten el acceso a los datos manejados por la empresa. Este modelo está diseñado para que sea más que una simple representación de los datos. Define la semántica, comportamiento e interacción entre las entidades gestionadas. En general toma dos formas: Diagramas UML de clases y un diccionario de datos que define la sintaxis, semántica y el uso de clases y atributos en general.

Los contratos no forman parte de la SID. Por el contrario, sí es posible que las especificaciones usen información del SID.

Los servicios de información se canalizan a través de un repositorio oficial donde se registran los datos de la empresa. Esto no implica a priori el uso de un repositorio centralizado o distribuido.

En todos los casos existe un "Dueño" de los datos claramente identificado, el cual controla quien puede realizar cambios en los datos y de qué manera (si existen varios "Dueños", existe un mecanismo claro que decide cual proceso estará habilitado para proceder a la modificación de los datos).

3.3.1.4 Adaptación de Información

Para lograr la interoperabilidad entre *Building Blocks* (uso de servicios disponibles internamente), la arquitectura necesariamente requiere del uso de objetos y estructuras de información que respondan al modelo de información común. En el mundo real, no siempre se posible establecer una conexión directa con este modelo ideal. En este sentido se incluyen los conceptos de "adaptación" y "mediación". "Adaptación" es un mecanismo para convertir una entidad en otra con diferente representación, de manera de poder lograr la interoperación entre ellas. Puede ocurrir que la interoperación pueda requerir el uso de una capa de "mediación".En este caso la transformación no es simplemente de representación y se requiere de funcionalidad adicional para sintetizar o generar u obtener datos a partir de los existentes (recurriendo eventualmente a servicios complementarios).

3.3.1.5 Roles de Usuarios y Seguridad

El sistema se caracteriza por que el usuario realiza el "logon" al sistema en forma general. Se definen las posibilidades de acceso a los distintos servicios en función del perfil del usuario.

3.3.1.6 Sistemas distribuidos escalables débilmente acoplados

La arquitectura propuesta se caracteriza por el hecho de que los *Building Blocks*, están débilmente acoplados entre sí y además eventualmente distribuidos (alojados en contenedores ubicados en Hosts diferentes).

Esto significa lo siguiente:

- Las instancias de *Building Blocks* corren independientemente unas de otras y la operación de uno no interrumpe necesariamente la operación de otro.
- Los *Building Blocks* son capaces de ejecutar en cualquier conjunto de computadoras que son conectadas entre sí por un vehículo de comunicación común.

- Se pueden agregar o remover *Building Blocks* sin afectar el comportamiento del resto de los *Building Blocks* instalados.
- El comportamiento global del sistema está determinado por la acción individual de cada uno de los Building Blocks que componen el sistema.
- La información utilizada por los *Building Blocks* pertenece al modelo de Información de la empresa.

En general los Sistemas de Soporte y Operaciones (OSS) contemporáneos se pueden describir como altamente acoplados, compuestos de un número de aplicaciones monolíticas, cada una de las cuales ofrece un conjunto de servicios dentro de un dominio particular. Estos sistemas legados se pueden integrar utilizando "gateways" de mediación (usando un acoplamiento débil).

3.3.2 Visión general de la Arquitectura

A continuación se ilustra con un ejemplo como funciona la arquitectura en términos generales. En las secciones siguientes se presenta un modelo conceptual donde se muestran los elementos que intervienen en la Arquitectura y las relaciones entre ellos, según distintos puntos de vista.

A los efectos ilustrativos, en la figura siguiente se pueden visualizar interacciones entre implementaciones de contratos encapsuladas en Building Blocks. En este caso el Building Block BB1 internamente requiere del uso de un Contrato externo del cual desconoce la ubicación exacta.

A estos efectos, la arquitectura provee de un Servicio de Registro General (*Registry Service*) que permite registrar y obtener referencias adecuadas a las Instancias de Implementaciones de Contratos (Implementaciones de Contratos activas) disponibles en el sistema. Una vez obtenida la ubicación del Contrato "Destino" (utilizando el Servicio de Registro) se lleva a cabo la propia invocación (ver Figura 18).

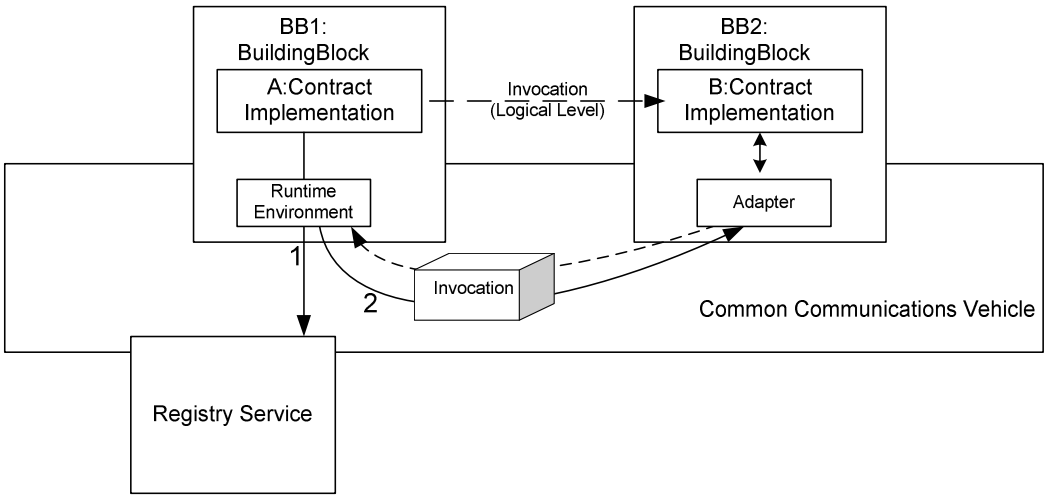


Figura 18 – Visión General

En el contexto de esta arquitectura la invocación del Contrato consiste en armar un "paquete" (Objeto `Invocation` en el Figura 18) que contiene la información requerida, como por ejemplo: referencia al Contrato, método, argumentos, etc. Este paquete se envía al Building Block destinatario de la invocación que contiene la Implementación del Contrato "activa". El Building Block destinatario dispone de un dispositivo Adaptador (`Adapter`), el cual finalmente es el encargado de llevar a cabo la invocación efectiva del Contrato solicitado utilizando la Implementación del Contrato disponible internamente en el Building Block (los detalles del procedimiento de invocación se presentan más adelante).

Para el transporte del "paquete" con los datos de la invocación (`Invocation`) se utiliza un Vehículo de Comunicaciones Común. Este Vehículo de Comunicaciones Común constituye una abstracción de las distintas formas de transporte que se pueden emplear para entregar los "paquetes" de invocación.

A continuación se presentan con mayor detalle los conceptos presentados hasta el momento según distintos puntos de vista.

3.3.2.1 Visión Estática

Las especificaciones de contratos (`ContractSpecification`) disponen de la metadata asociada al Contrato. Se incluye la información de métodos soportados (`ContractMethod`), conjunto de parámetros

utilizado por cada método (`Parammeter`) y condiciones que deben cumplir los métodos. Cada método tiene asociado un conjunto de condiciones que pueden ser Precondiciones (`Precondition`) o Poscondiciones (`Postcondition`) .Cada Condición está vinculada a una Excepción específica (en el funcionamiento del sistema, esta excepción se generará cuando no se verifica la Condición respectiva).

El alcance del concepto "Contrato"(`Contract`) a nivel del framework que modela la presente Arquitectura, se circunscribe al concepto usual de "interfase" a nivel de un lenguaje de programación específico. En este sentido se dispone de Contratos (`Contract`) que responden a sus respectivas especificaciones (`ContractSpecification`) en una relación de dependencia. Luego, los contratos se implementan en plataformas específicas a través de sus Implementaciones de Contratos (`ContractImplementation`).

Finalmente, las Implementaciones de Contratos se agrupan en paquetes de Implementaciones de Contratos o "Building Blocks" que constituyen unidades indivisibles desde el punto de vista de gestión de Instalación (*deployment*) de los mismos (ver Figura 19).

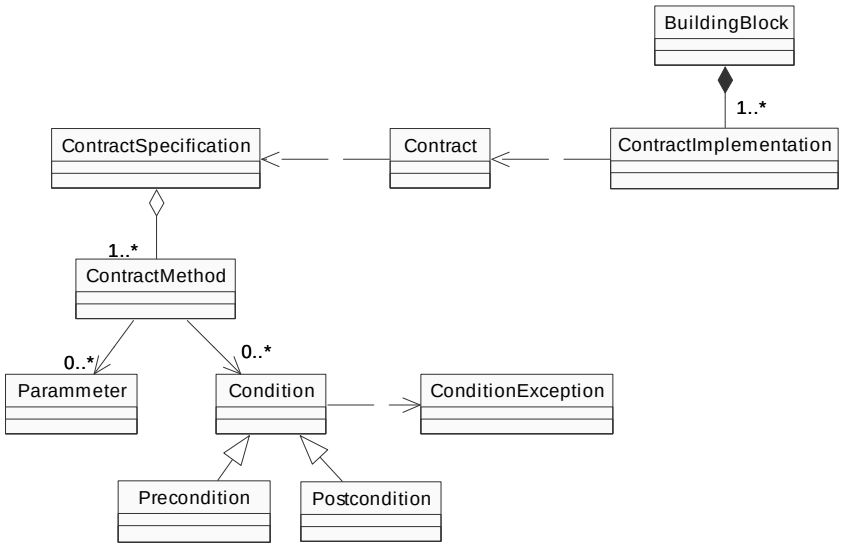


Figura 19 – Visión Estática

Es de destacar que el concepto de `ContractSpecification` está utilizado desde la perspectiva del usuario del contrato. Se especifican requerimientos para el uso de los mismos (no se hacen consideraciones de requerimientos a tener en cuenta por el desarrollador de los mismos).

3.3.2.2 Punto de vista de la ejecución

En la visión anterior, se mostró el concepto de Implementación de Contrato que constituye la implementación del comportamiento a exhibir ante la invocación del Contrato correspondiente.

Para que las implementaciones de contratos sean posibles de ser usadas, deben ser instaladas *(deployed)* e instanciadas. Cada Instancia de Implementación de Contrato (`ContractImpInstance`) está identificada por un nombre que la individualiza, y es registrada a nivel central (utilizando el Servicio de Registro General), a los efectos de que pueda ser ubicada por los usuarios (ver Figura 19).

Es importante distinguir los conceptos de Implementación de Contratos e Instancia de Implementación de Contratos. El concepto de Implementación de Contrato es un concepto estático y se refiere simplemente a la existencia de una implementación específica contenida en un Building Block. La existencia de una Instancia de Implementación de Contrato supone la puesta en marcha de una implementación del Contrato, quedando en un estado "activo" que posibilita el uso del mismo. Es de

destacar que puede haber muchas instancias activas para una misma implementación del contrato (cada una de ellas con identidad propia).

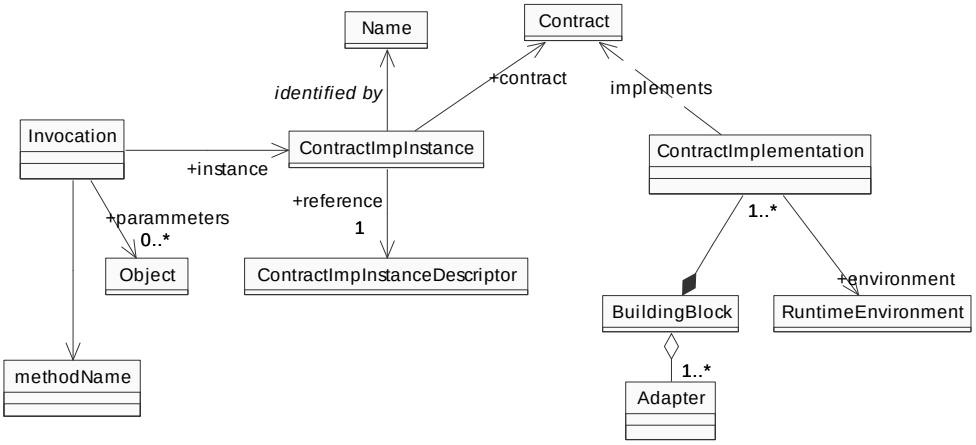


Figura 20 – Punto de vista de la Ejecución

Cada instancia de implementación dispone de un descriptor del Adaptador que funciona como punto de acceso a la misma (`ContractImpInstanceDescriptor`). Este descriptor contiene la información para invocar el servicio.

Desde un punto de vista lógico de alto nivel, es posible interpretar que las instancias de Implementaciones de Contratos están conectadas al Vehículo de Comunicaciones Común y donde cualquier implementación conectada a este "Bus" puede invocar métodos de estas Instancias de Implementaciones de Contratos. El descriptor `ContractImpInstanceDescriptor` se puede interpretar como la dirección de la Instancia mencionada a nivel del Vehículo de Comunicaciones Común.

Es de destacar que el bus de servicios no es homogéneo por lo que distintos protocolos pueden ser utilizados. En este sentido, el descriptor del punto de acceso maneja distinta información de acuerdo a la tecnología subyacente.

Como se puede ver en la Figura 20, todas las realizaciones de Implementaciones de Contratos en el marco de esta Arquitectura, derivan de la clase `ContractImplementation` (ver Figura 20). Esto posibilita la disponibilidad de los servicios básicos brindados por la Arquitectura. De esta manera desde el punto de vista de un cliente de un contrato (por ejemplo la Implementación de Contrato: A en la Figura 18), se dispone de una abstracción de un entorno de ejecución (`RuntimeEnvironment`) que es el que brinda el servicio de invocación de contratos externos, permitiendo acceder a contratos con independencia de la plataforma sobre la cual se apoyan.

El mecanismo de invocación se ilustrará con mayor detalle al derivar la arquitectura a tecnologías específicas, pero se puede resumir fundamentalmente en los siguientes pasos:

- Se obtiene la Instancia de Implementación de Contrato utilizando el servicio de Registro General.
- El Entorno de Ejecución del cliente arma un objeto de tipo Invocación (`Invocation`) con los detalles de la invocación como Instancia de Implementación de Contrato, método, argumentos, etc. (ver Figura 20).
- El cliente envía el objeto Invocación (`Invocation`) al Adaptador (`Adapter`) presente en el Building Block que alberga la Implementación de Contrato destinataria.
- Finalmente el Adaptador es el encargado de hacer efectiva la implementación.

3.3.2.3 Visión de Servicios Distribuidos

Cada implementación de contrato que implementa un contrato registrado, se instancia en el contexto de un nodo de procesamiento o `ServerNode` conectado al bus de servicios (desde el punto de vista de procesamiento distribuido) (ver Figura 21). Cada instancia de implementación de contrato dispone de una identidad propia y tiene asociado una referencia (caracterizada por un descriptor asociado de tipo `ContractImpInstanceDescriptor` como vimos anteriormente). Cada `ServerNode` maneja un conjunto de puntos de acceso alternativos (`gatewayReferences`) que sirven de "pasarela" a los efectos de alcanzar un servicio brindado a nivel de un `ServerNode` específico. Este mecanismo se utiliza a los efectos de "enlazar" distintas plataformas en forma transparente o bien eventualmente cambiar el patrón de interoperabilidad previsto inicialmente por el servicio brindado a través de una implementación de un contrato dado.

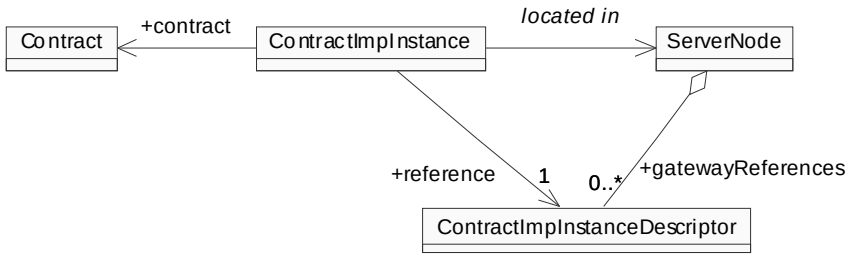


Figura 21 – Visión de Procesamiento Distribuido

Como se puede ver en la Figura 22, pueden existir invocaciones en forma directa entre implementaciones de contratos pertenecientes al mismo nodo. Se ilustran también invocaciones "inter-nodo". En estos casos pueden accederse directamente o eventualmente hacer uso de puntos de acceso alternativo (o gateway) provistos por el propio `ServerNode` aludido por la invocación

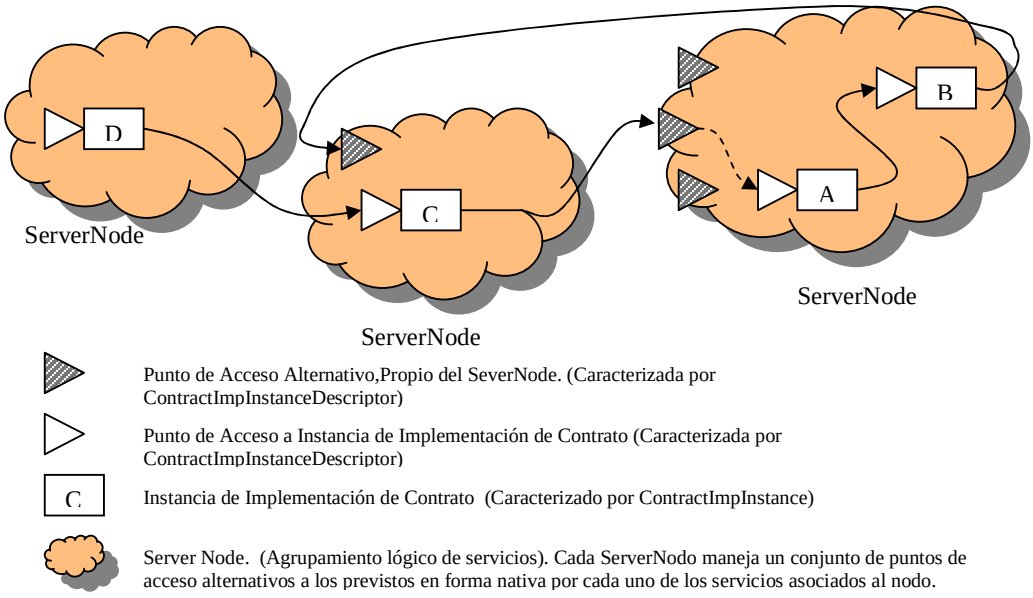


Figura 22 – Nodos desde el punto de vista de la arquitectura

Puede aparecer que los conceptos de `ServerNode` introducidos previamente y la idea de Servidor de Aplicaciones que contiene "componentes" están estrechamente relacionados. En principio el cometido es

agrupar lógicamente servicios que comparten puntos de accesos alternativos o gateways. Podrían existir implementaciones de un `ServerNode` distribuidas a lo largo de un conjunto de Hosts (por ejemplo un punto de acceso que manejara HTTP).

3.3.3 Servicios provistos por la Arquitectura

La arquitectura se caracteriza por la existencia de un conjunto de servicios básicos (de la propia arquitectura) que facilitan la operación de los componentes del sistema. Algunos de estos servicios son:

- *Servicio de Registro General.* Brinda servicios para soportar la transparencia en la ubicación de entidades.
- *Servicio de Invocación.* Permite que las entidades puedan invocar servicios provistos por otras entidades de acuerdo a las reglas del contrato del servicio y las políticas del sistema.
- *Servicios de Seguridad.* Mecanismos que posibilitan la autenticación, autorización, auditoría, etc.
- *Servicios de Transacciones.* Mecanismos que posibiliten el control de la ejecución de contratos en un entorno transaccional.

Notas: Es de destacar que en el presente trabajo no se abordan los aspectos de Servicios de Seguridad .El framework propuesto tampoco maneja explícitamente el control de transacciones. En este sentido, se apoya en el potencial de la tecnología elegida. Por ejemplo, los contratos implementados en el contexto de Servidores de Aplicación J2EE se ejecutan en un entorno transaccional provisto por el propio contenedor en una forma transparente.

3.3.3.1 Registro General

Brinda servicios para soportar la transparencia en la ubicación de distintos elementos propios de la arquitectura. Los elementos registrados que son de interés desde el punto de vista arquitectónico son:

- Registro de Contratos (`Contract`)
- Registro de Building Blocks
- Registro de Implementaciones de contratos (`ContractImplementation`)
- Registro de Instancias de Implementaciones de Contratos Disponibles (`ContractImpInstance`)

El Servicio de Registro General provee de un repositorio (se asume que es un único repositorio lógico, pero que puede estar compuesto de una federación de repositorios físicos separados) con instrucciones para ubicar los elementos mencionados anteriormente y como pueden ser accedidos o invocados.

Principios usados:
- Se puede encontrar información independiente de la tecnología referenciando los contratos apropiados.
- Se puede encontrar información (aplicada a tecnologías específicas) referenciando las implementaciones de contratos apropiadas.

Este servicio tiene importancia solamente durante el descubrimiento de las operaciones.

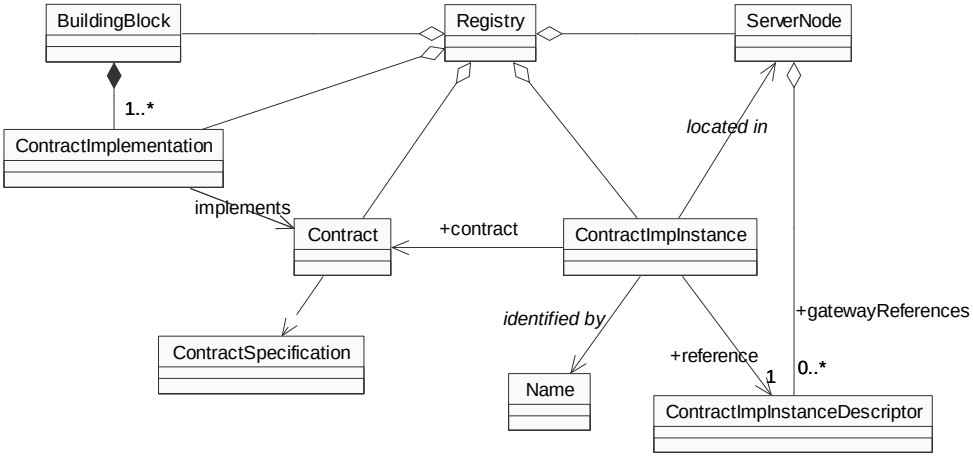


Figura 23 – Servicio de Registro

Como se puede ver en la Figura 23 el servicio de registro mantiene la información de Building Blocks (`BuildingBlock`) y contratos implementados (`ContractImplementation`) disponibles en cada uno de ellos. A su vez para cada contrato existen las implementaciones de contratos instanciadas (`ContractImpInstance`). Cada implementación de contrato instanciado (`ContractImpInstance`) registrada tiene un identificador que lo caracteriza. Con este nombre característico se puede referenciar cualquier contrato instanciado del sistema.

3.3.3.2 Servicio de Invocación

Uno de los objetivos del uso del servicio de invocación es ocultar los detalles del procedimiento subyacente involucrado. La tecnología y el modo de invocación son transparentes al desarrollador de la aplicación que especifica una determinada invocación.

La existencia del servicio de invocación posibilita entre otras cosas:

- Validación de credenciales para el acceso a contratos (se validan roles de acceso).
- Validación de precondiciones. Se validan las precondiciones estipuladas en el contrato. Si no se cumplen se genera la excepción correspondiente.
- Log de uso de la interfaz. Es posible hacer un Log (de las invocaciones realizadas a la interfaz).

Notas:

- El uso de precondiciones puede soslayarse en aquellos subsistemas donde las interacciones son realizadas por el propio sistema. Por el contrario, puede ser de importancia a los efectos de asegurar la integridad del sistema en aquellos casos donde la invocación se realiza desde un motor de Workflow o son operaciones iniciadas por operadores del sistema. Esta misma consideración vale para la validación de credenciales del solicitante del servicio.

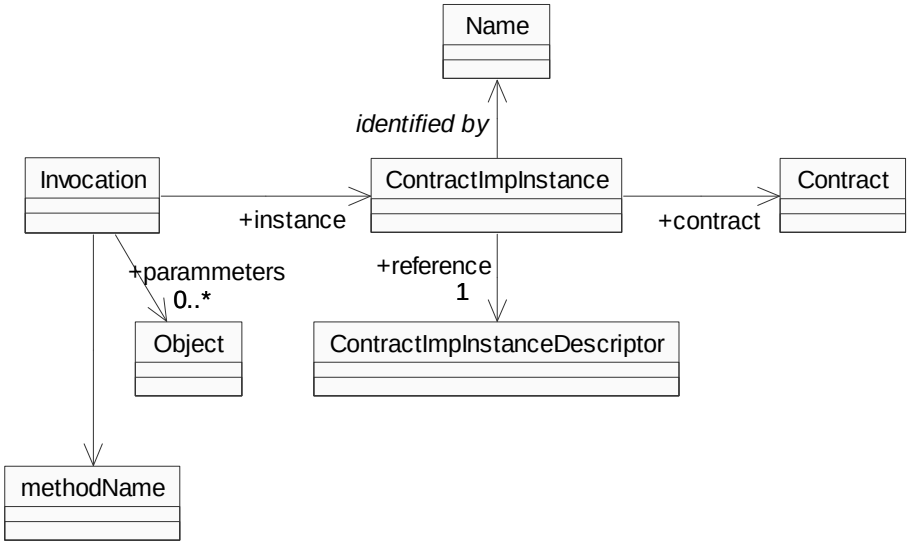


Figura 24 – Invocación

Cada Implementación de Contrato que desea invocar un Contrato disponible (`Contract`) en un Building Block lo hace a través del servicio de invocación.

El servicio de invocación opera en dos etapas:

1. El servicio de invocación ubica el Contrato usando el servicio de Registro General. Se obtiene la Instancia de Implementación de Contrato (`ContractImpInstance`) asociada a la identificación o nombre proporcionado (`Name`) (ver Figura 24).
2. Con la Instancia de Implementación de Contrato obtenida (`ContractImpInstance`), se arma un "paquete" de invocación (`Invocation`) agregando además, método a invocar y argumentos requeridos (ver Figura 24). Finalmente se remite este "paquete" al Building Block que alberga la Implementación del Contrato "destino".

La información de la Instancia de Implementación de Contrato (obtenida a través del servicio de Registro General) especifica las características de la invocación (si es sincrónica o asincrónica), la plataforma que la implementa y la dirección real del dispositivo Adaptador en el Building Block "destino". Esta información es utilizada para "enrutar" correctamente la invocación hacia el destino real utilizando los recursos y tecnologías apropiadas para ello.

A modo de ejemplo, en el caso de invocación sincrónica, el "paquete" se transporta directamente a través de la invocación en forma sincrónica utilizando primitivas disponibles en el Adaptador del Building Block que contiene la implementación del Contrato requerida (por ejemplo en la plataforma J2EE puede implicar la ejecución de un EJB Session Bean).

En el caso asincrónico el "paquete" de invocación se enruta al destinatario a través de algún sistema de mensajería. En la arquitectura J2EE el "paquete" es enviado a un Message Driven Bean receptor, a través de un canal de mensajería JMS.

Es importante destacar que desde el punto de vista del Cliente del Contrato, la implementación del código permanece "neutral" respecto a la forma de comunicación con la Implementación del Contrato "Servidor". Los detalles de la tecnología empleada quedan asociados a la configuración (disponibles en el Servicio de Registro General). El servicio de Invocación utiliza en forma transparente para el cliente del servicio la información tecnológica disponible. Los detalles se presentan en la siguiente sección donde se aborda la Arquitectura desde el punto de vista de su aplicación a Tecnologías específicas.

3.4 Arquitectura aplicada a tecnologías específicas

En la sección anterior se ha presentado la arquitectura en términos neutrales. En particular se ha mostrado cómo implementaciones encapsuladas en Building Blocks pueden interactuar entre sí utilizando un vehículo de comunicaciones común, permitiendo que las implementaciones hagan referencias a Contratos en términos neutrales (sin referencias a los mecanismos o tecnologías empleadas).

En esta sección se pretende mostrar con mayor detalle, el concepto de Vehículo de Comunicaciones Común y como se puede implementar utilizando mecanismos sincrónicos y/o asincrónicos operando sobre distintas tecnologías.

3.4.1 Patrones de Interoperabilidad

A lo largo del presente trabajo se hará referencia con frecuencia al término "Patrón de Interoperabilidad". Dada una interacción real posible entre un Cliente y un Servidor, nos referimos al "patrón de interoperabilidad" empleado, a la forma en que se lleva a cabo la interacción teniendo en cuenta las siguientes dimensiones:

- Modalidad de Interoperabilidad (si es sincrónica o asincrónica)
- Tecnología de interoperabilidad empleadas (Ejemplo CORBA, EJB, etc.).

La arquitectura permite manejar distintos patrones de interoperabilidad a los efectos de procesar las interacciones entre Building Blocks. Las interacciones pueden ser sincrónicas o asincrónicas y apoyarse sobre distintas tecnologías. Por ejemplo, se pueden manejar interacciones asincrónicas apoyadas sobre mensajería JMS (en el caso de J2EE) o el servicio *Notification Service* de la plataforma CORBA. Lo mismo sucede a nivel de interacciones sincrónicas, podría usar RMI o IIOP. Incluso se pueden modelar en forma transparente el uso de interfaces locales o remotas para EJB.

3.4.2 Interacciones entre Implementaciones de Contratos.

En la sección 3.3.2 dentro de la presentación de la visión general de la arquitectura (neutral) se presentó en la Figura 18, un diagrama donde se muestran interacciones entre implementaciones de contratos encapsuladas en Building Blocks. A continuación, se amplia el mismo esquema (Figura 25) de manera de mostrar el hecho de que el Vehículo de Comunicaciones Común está constituido potencialmente por diferentes alternativas de comunicaciones (tanto sincrónicas como asincrónicas) y utilizando eventualmente distintas tecnologías.

Se puede visualizar al Vehículo de Comunicaciones Común como un conjunto de rutas paralelas de transporte (por donde transitan paquetes de invocación y eventuales respuestas), donde cada ruta responde a un patrón de interoperabilidad específico (según la definición de la sección anterior). Se puede ver en la figura, que las rutas A y B implican caminos directos hacia los distintos destinos (correspondientes a patrones de interoperabilidad sincrónicos), mientras que en el caso de la ruta etiquetada como C, figura como intermediario un sistema de mensajería. En este caso el transporte se lleva a cabo en forma asincrónica. Los "paquetes" transportados se entregan en la "Oficina Postal" y esta entrega posteriormente y en forma asincrónica el paquete al destinatario final.

Cada Implementación de Contrato disponible en el sistema es accesible a través de un Adaptador (o más de uno eventualmente). Cada adaptador está conectado a una ruta específica. En el caso de la figura se puede visualizar que la Implementación de Contrato en BB2 es servida a través de un Adaptador conectada a la ruta B (la ruta de transporte que responde al patrón de interoperabilidad B).

Desde la perspectiva del cliente, la implementación de contrato en BB1, hace uso del servicio de invocación provisto por su Entorno de Ejecución. A este nivel, la comunicación entre la implementación y el entorno de ejecución se lleva a cabo en términos neutrales (se desconocen los detalles de la implementación a este nivel).

El Entorno de Ejecución recurre al Servicio de Registro para ubicar la Instancia de Implementación de Contrato (como se ha especificado anteriormente). Es de destacar que la Instancia de Implementación de Contrato devuelta por el Servicio de Registro, entre otras cosas informa de las características de la implementación real. De esta manera el entorno de ejecución conoce la "ruta" por la que se llega al destino requerido.

Como se puede ver en la figura, el entorno de ejecución dispone de un conjunto de puertas de acceso a cada una de las "rutas" o caminos de transporte. El entorno de ejecución , utiliza la puerta de salida apropiada para poder utilizar la ruta de transporte adecuada para llegar al destino requerido.

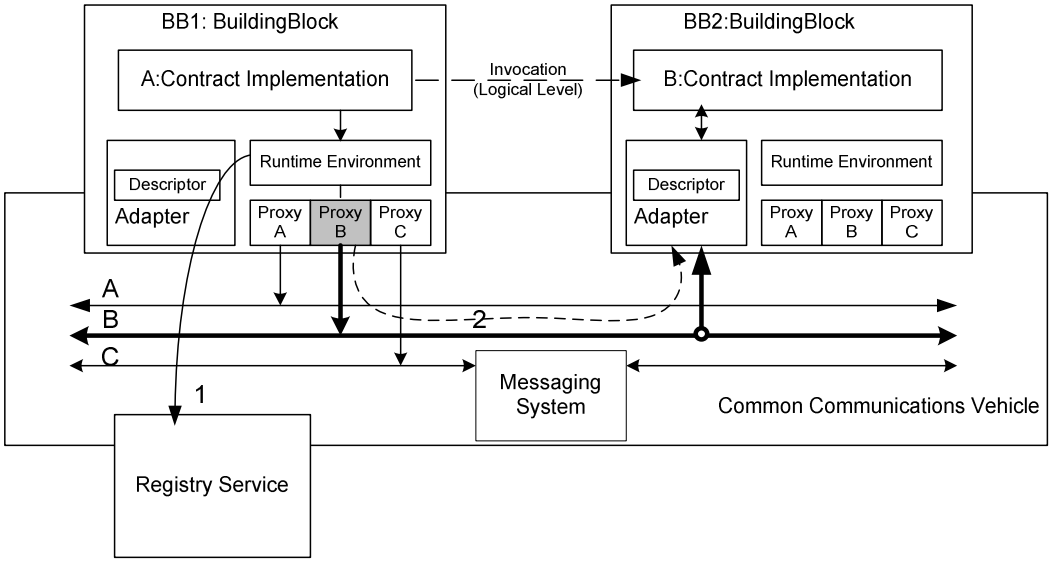


Figura 25 – Invocación de Contratos

3.4.2.1 Perspectiva del Servidor

Las implementaciones de los contratos se realizan en el marco de un framework básico, como se describe a continuación. La implementación del Contrato (`ServiceA` en la Figura 26) se lleva a cabo en una clase que realiza la interfaz del contrato a implementar (`IServiceA`). La clase que implementa el contrato deriva de una clase abstracta `ContractImplementation` de manera que hereda los servicios básicos del framework, posibilitando la comunicación con componentes externos. De esta manera la implementación se realiza en términos "neutrales" en cuanto a la interoperabilidad con otros Building Blocks. La implementación está escrita de manera que la comunicación con otros componentes se realiza a través de la primitiva de invocación de contratos.

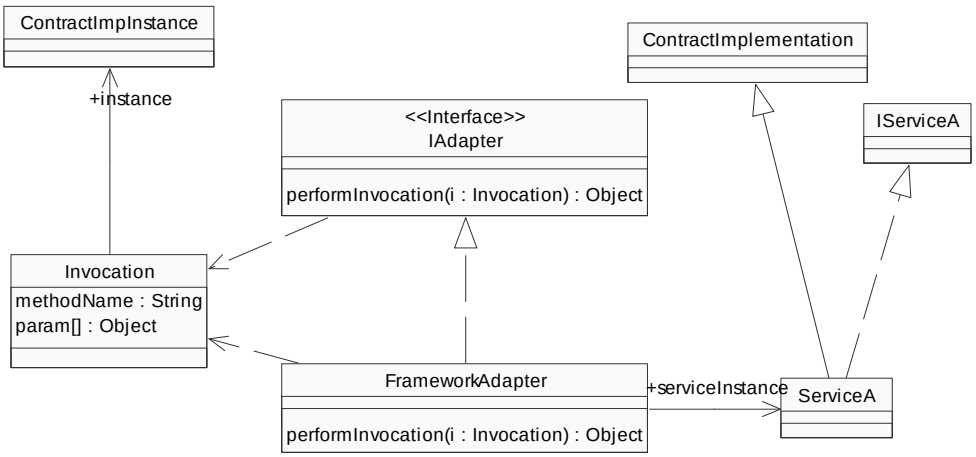


Figura 26 – Adaptadores para Implementaciones de Contratos

La implementación del contrato es accesible a través de un adaptador (`FrameworkAdapter`) asociado a la misma. El adaptador que posibilita el acceso consiste básicamente en un artefacto que implementa la interfaz `IAdapter` (lo cual exige la implementación del método `performInvocation(Invocation:i ): Object ).`

El adaptador sostiene la implementación del contrato. Las implementaciones de contratos que requieren comunicarse con una instancia de implementación de contrato (implementada por `ServiceA`) lo hacen a través del adaptador mencionado, llevando a cabo la operación `performInvocation` sobre el mismo, pasando como parámetro el objeto `Invocation` que contiene los detalles de la invocación a realizar. Finalmente el adaptador llevará a cabo la invocación real del método solicitado con los argumentos correspondientes, utilizando la implementación del contrato (ver Figura 26).

3.4.2.2 Perspectiva del Cliente

Ya se ha explicado que las implementaciones de contratos utilizan el Servicio de Invocación (ofrecido por el Entorno de Ejecución disponible) para invocar Contratos en general. A continuación en la siguiente sección se ilustra el procedimiento por el cual, el Entorno de Ejecución se comunica con el Vehículo de Comunicaciones Común.

3.4.2.2.1 Acceso al Vehículo de Comunicaciones Común desde el Entorno de Ejecución.

El entorno de ejecución está equipado con un conjunto de dispositivos que implementan cada uno de los patrones de interoperabilidad requeridos, denominados `Proxy (ver` Figura 25`)`. Cada `Proxy` es capaz de manejar un patrón de interoperabilidad específico. El perfil del entorno de ejecución (`RuntimeEnvironment`) esta dado por el conjunto de artefactos `Proxy` disponibles (ver Figura 27).

Por otro lado cada instancia de implementación de contrato (`ContractImpInstance`), caracterizada por un descriptor asociado (`ContractImpInstanceDescriptor`) (obtenible desde el servicio central de Registro General), señala el patrón de interoperabilidad requerido como se puede ver en el diagrama (`InteroperabilityPattern`).

En el momento de la invocación se selecciona dinámicamente el `Proxy` adecuado, compatible con la implementación. Este dispositivo es el que finalmente lleva a cabo la interacción requerida.

En principio el uso de los distintos patrones de interoperabilidad se maneja en forma transparente.

Cuando no existe una compatibilidad directa entre la implementación del Contrato y el entorno de ejecución, se utilizan “gateways” en forma transparente. Cada “nodo” conectado al vehículo de comunicaciones común, maneja puntos de accesos alternativos a los ofrecidos por las implementaciones

de contratos. De esta manera la búsqueda de la implementación de un contrato dado tiene en consideración el perfil del entorno de ejecución "invocante", de manera que el descriptor de la implementación de contrato devuelto sea compatible con el perfil del entorno de ejecución.

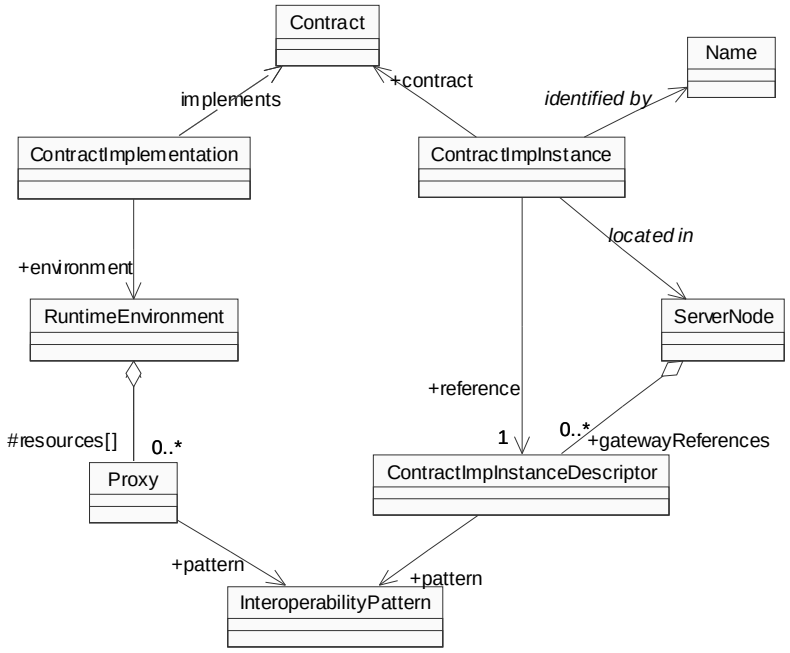


Figura 27 - Patrones de Interoperabilidad

3.4.2.2.2 Invocación desde la perspectiva del Cliente

A continuación se presenta un diagrama de secuencias donde se ilustran los procesos involucrados para llevar a cabo una invocación (según la perspectiva del Cliente) (Figura 29).

Inicialmente el Entorno de Ejecución (`RuntimeEnvironment`) se comunica con el servicio de Registro (`Registry`) para obtener una Instancia de Implementación de Contrato (`instance` en el diagrama) a partir del nombre de la instancia requerida (`instanceName`). A continuación se crea un objeto `Invocation` que contiene todos los detalles de la invocación (`ContractImpInstance` requerido, método y lista de argumentos del método). El servicio de invocación del Entorno de Ejecución localiza en primer lugar el dispositivo `Proxy` compatible con el patrón de interoperabilidad requerido por la Instancia de Implementación de Contrato `instance`. En el diagrama, el `proxy` p1 es el elegido. Se enruta de esta manera el paquete de Invocación `"i"` hacia el `proxy` `"p1"`. Éste sabe como hablar con el adaptador asociado a la implementación del contrato requerido, enrutando la invocación hacia el adaptador (`FrameworkAdapter`). Finalmente es el Adaptador quien invoca el método por mecanismos de reflexión. En este caso se utilizan los mecanismos de reflexión de Java.

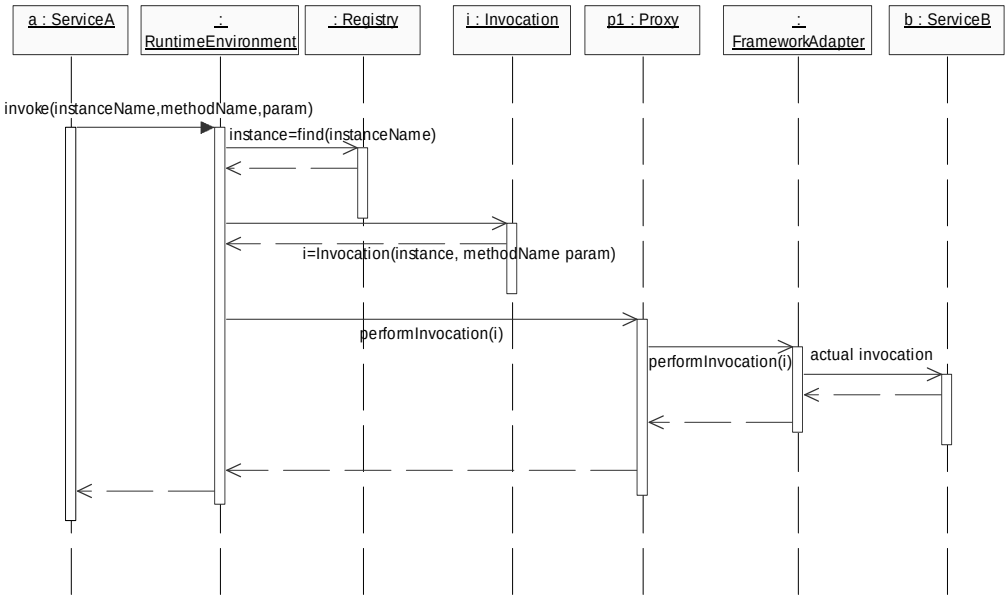


Figura 28- Diagrama de Secuencias de una Invocación

3.4.2.2.3 Ubicación de Instancias de Implementaciones de Contratos

Desde la perspectiva del entorno de ejecución, en lo que tiene que ver con la invocación de un servicio el framework provee al servicio de dos primitivas básicas

- Invocar un servicio directamente indicando el "nombre" de la instancia.
- Resolver a través del servicio de registro el "nombre" de la instancia

En el primer caso, el framework internamente resuelve el nombre, a través del propio servicio de registro.

En el segundo caso, el servicio de registro retorna un objeto de tipo `ContractImpInstance` que contiene la información para llevar a cabo la invocación en un momento posterior.

Es importante destacar que el procedimiento para resolver el "nombre" de un servicio, tiene en cuenta en forma transparente para el usuario, las características del entorno de ejecución disponible. El perfil del entorno de ejecución resulta del OR-lógico de todos los patrones de interoperabilidad previstos (cada patrón de interoperabilidad está caracterizado por una máscara binaria única que no se solapa con los demás patrones de interoperabilidad). Al invocar el servicio de registro se pasa como parámetro este valor, de manera que el servicio de registro devuelva una respuesta compatible con el entorno de ejecución.

Para lograr lo anterior se ubica la implementación nativa. Cada instancia de implementación está caracterizada por su descriptor, que a su vez señala un patrón de interoperabilidad específico. Si el descriptor ubicado es compatible con la máscara del perfil del entorno de ejecución (AND-lógico entre mascaras de patrones de interoperabilidad debe ser no nulo) entonces se devuelve la implementación del contrato encontrado. En el caso de que no se encontrase el servicio de registro, recurre a los puntos de accesos alternativos disponibles en el nodo que alberga el servicio. En este sentido, se recorre la lista de descriptores asociados a cada nodo, verificando si alguno de ellos es compatible con el perfil del entorno de ejecución invocante. Si se encuentra algún descriptor alternativo, entonces el servicio de invocación "arma" la descripción de la instancia de implementación de contrato, utilizando el descriptor alternativo (que oficia de gateway).

3.4.2.2.4 Políticas de selección de patrones de interoperabilidad específicos

El framework maneja en forma transparente la elección del `Proxy` apropiado para procesar la invocación.

En algunos casos puede requerirse la elección de un determinado patrón de interoperabilidad. Por ejemplo, si el nodo que alberga el servicio dispone de varios puertos de acceso alternativos y el entorno de ejecución maneja varias alternativas, en estos casos el framework maneja un esquema de prioridades por defecto para la selección del `Proxy` apropiado. El desarrollador de la implementación puede dirigir la elección dinámica del `Proxy` en base a directivas que pueden ser específicas para la invocación o la sesión. De esta manera la elección dinámica tiene en cuenta solo los patrones de interoperabilidad "aceptables" para el desarrollador, encasillando la elección del `Proxy` a utilizar dentro de un criterio específico. Por ejemplo, el desarrollador podría haber forzado el uso de alternativas sincrónicas o asincrónicas, o incluso forzar el uso de un patrón especifico como el uso de una mensajería específica.

La implementación de esta política es sencilla y se reduce a especificar una máscara apropiada de los patrones "aceptables" a utilizar. En el momento de la invocación se aplica la máscara, al perfil del entorno de ejecución, de manera de limitar las posibilidades del entorno de ejecución al deseo del desarrollador.

3.4.3 Extensión del Framework "Básico" hacia tecnologías específicas

Para cada tecnología específica se extiende el modelo del "framework básico neutral" de manera de poder soportar la metadata requerida en cada escenario de interoperabilidad.

Los patrones de interoperabilidad previstos por el framework propuesto hasta el momento son:

- *Ejb Modo Sincrónico*. La implementación del servicio se realiza en la arquitectura EJB *(Enterprise Java Beans)* a través de un EJB de tipo *Session Bean* [30].

- *Ejb Modo Asincrónico*. La implementación del servicio se realiza en la arquitectura EJB *(Enterprise Java Beans)* y se accede a través de un canal de mensajería de la arquitectura J2EE [30].

- *Corba Modo Sincrónico.* La implementación se realiza utilizando un Servicio CORBA.

- *Corba Modo Asincrónico.* La implementación se realiza utilizando un Servicio CORBA y se accede a la misma a través de l servicio de eventos de CORBA *Notification Service*.

Como se observa en la Figura 29, los conceptos: `RuntimeEnvironment`, `Proxy` y `ContractImpInstanceDescriptor` se extienden para modelar las particularidades de cada patrón de interoperabilidad previsto.

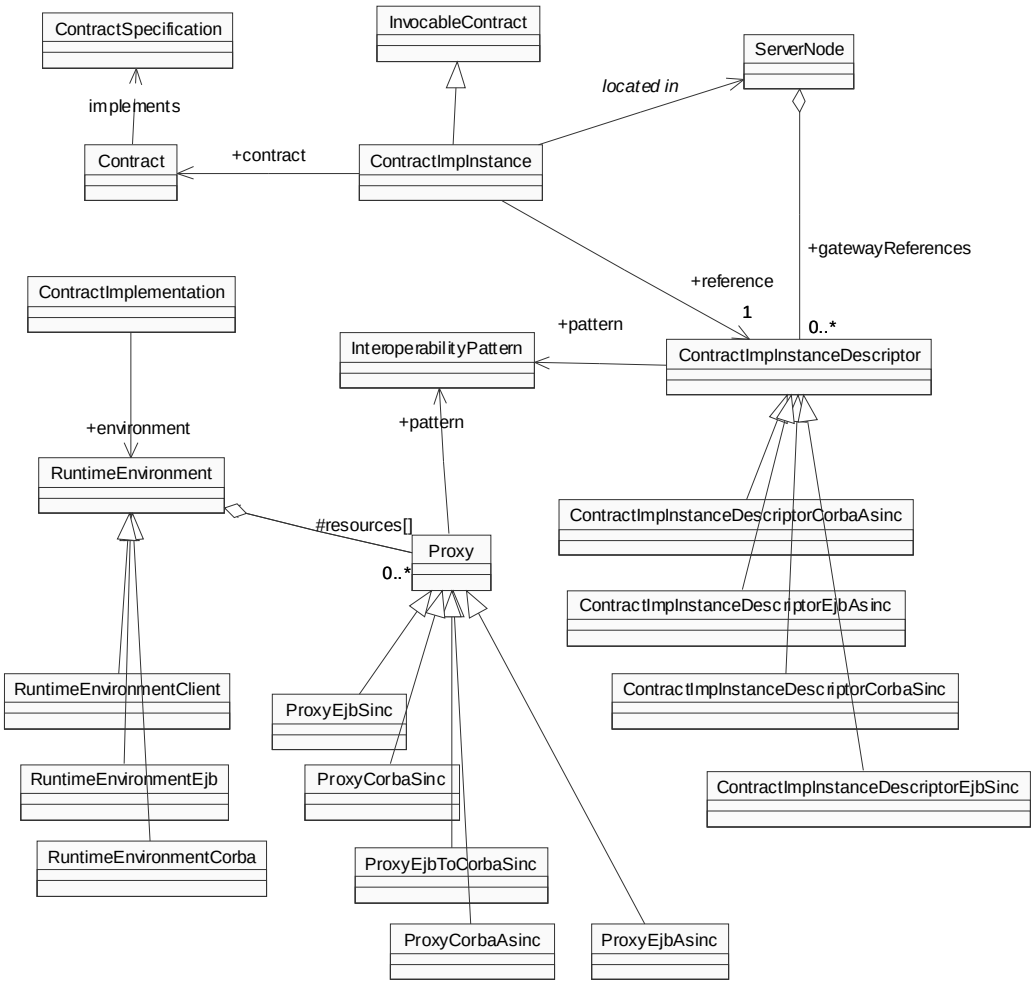


Figura 29 - Extensión del Framework a Tecnologías específicas

3.4.4 Adaptadores para la Implementación de Servicios

3.4.4.1 Adaptadores de Implementaciones de contratos en arquitectura EJB.

Desde el punto de vista de la implementación vamos a distinguir dos tipos de casos.

- Contratos Instanciados que mantienen el estado interno
- Contratos Instanciados que no mantienen el estado interno.

Los contratos instanciados que no mantienen el estado interno presentan ventajas desde el punto de vista de la implementación ya que funcionan mejor en un escenario donde se requiere “alta disponibilidad”.

Es importante destacar que la implementación brinda el servicio en dos modalidades:
- Modo sincrónico. La invocación del servicio espera por el resultado de la operación.
- Modo asincrónico. La invocación del servicio no espera por el resultado de la operación.

3.4.4.2 Contratos Instanciados de servicios que no mantienen el estado interno

3.4.4.2.1 Interfaz sincrónica

El Adaptador consiste en un *Session Bean Stateless* que implementa la interfaz `IAdapter` . El *Session Bean Stateless* implementa el método `performInvocation` que recibe como parámetro un objeto `Invocation` y devuelve un objeto `Object` (como se puede ver en la Figura 30) .

Dado que no se mantiene el estado interno, se crea una instancia del proveedor de servicios internamente en cada invocación.

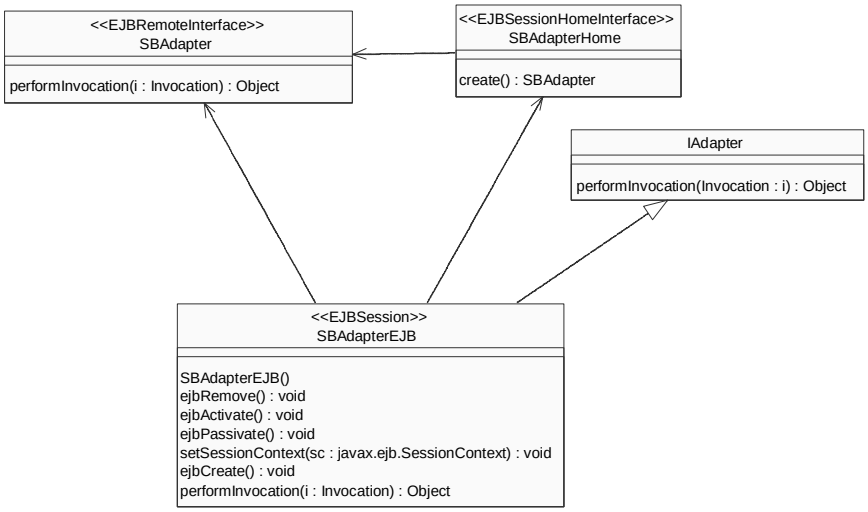


Figura 30 - Adaptador para tecnología EJB sincrónico (stateless)

La invocación se efectiviza por la ejecución del método `performInvocation` del Adaptador.

El objeto `Invocation` (pasado como parámetro) contiene los detalles de la invocación requerida, contiene en particular la información de la clase que implementa el servicio. La clase `BuildMethod` provista por el framework provee de la lógica para construir la invocación real. De hecho se crea dinámicamente una instancia de la clase que implementa el servicio, asignándole inicialmente un entorno de ejecución EJB (creado dinámicamente para esta ocasión).Una vez creada la clase que implementa el servicio, se efectiviza la invocación del método.

3.4.4.2.2 Interfaz asincrónica

En este caso la invocación se lleva a cabo a través de la mensajería del Servidor de Aplicaciones *(Application Server)*. El componente que realiza una invocación asincrónica a este servicio, crea el objeto `Invocation` con los detalles del servicio a ejecutar y se encapsula en un mensaje JMS remitido al canal de mensajería asociado al servicio destinatario (ver Figura 31). Los canales de mensajería son tópicos de JMS disponibles en el contenedor.

El Adaptador consiste en un *Message Driven Bean (MDB)* que está asociado al canal de mensajería que le pertenece (ver Figura 32). De esta manera cuando el tópico asociado al MDB recibe un mensaje con una invocación, se activa el MDB (se ejecuta el método `onMessage`). Posteriormente, se levanta una sesión con el Adaptador sincrónico tal como se mostró anteriormente, de manera de invocar el método `performInvocation` por el Adaptador sincrónico.

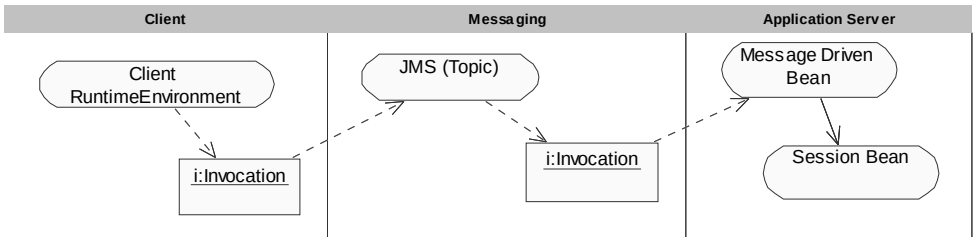


Figura 31 - Invocación asincrónica vía JMS

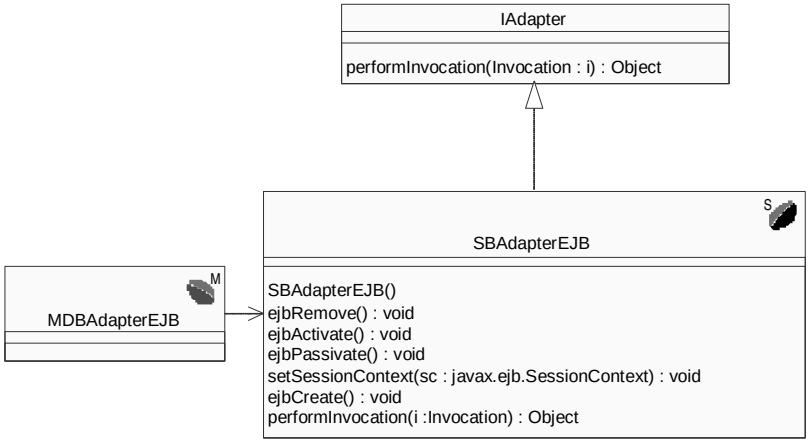


Figura 32 - Adaptador para tecnología EJB asincrónico

3.4.4.3 Contratos Instanciados de servicios que mantienen el estado interno

Interfaz Sincrónica

A diferencia del caso anterior, cuando se crea el servicio, en este momento se crea una sesión que mantiene el estado interno utilizando un *EJB Session Bean Statefull*. La sesión debe albergar el proveedor de servicios de manera que la primitiva `performInvocation` se enruta al proveedor de servicios creado inicialmente.

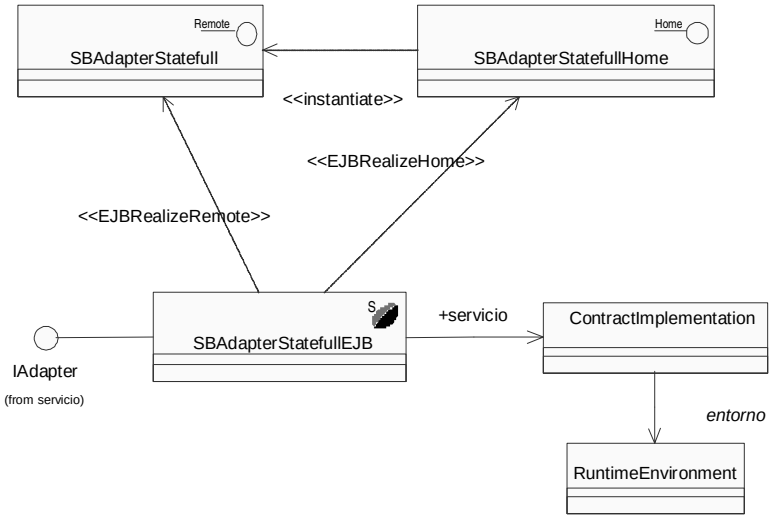


Figura 33 - Adaptador para tecnología EJB sincrónico (statefull)

El Descriptor del contrato contiene la referencia a la sesión que hace que se posible su ubicación.

Interfaz Asincrónica
Es similar al caso anterior. La diferencia es que en este caso la invocación se realiza directamente sobre la sesión referenciada en el objeto `Invocation`.

3.4.4.4 Adaptadores de Implementaciones de contratos en arquitectura CORBA.

3.4.4.4.1 Interfaz Sincrónica

El cliente que realiza una invocación sincrónica lo hace dirigiéndose a una instancia de un objeto CORBA que implementa la interfaz `IAdapter`.

```
Interface IAdapter
{
Object performInvocation(in Invocation i);
}
```

Este objeto CORBA dispone internamente de la instancia propia del servicio (`ContractImplementation`), de manera que al activarse el servicio a través de la interfaz genérica, se lleva a cabo la invocación posteriormente utilizando la información del objeto `Invocation`. Se utiliza la lógica de reflexión propia del Framework, para construir y llevar a cabo efectivamente la invocación sobre la implementación real (instanciada) del servicio.

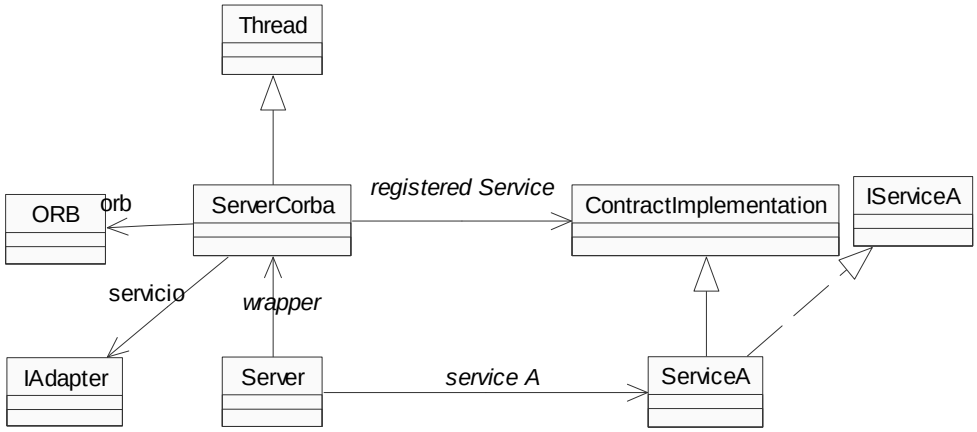


Figura 34 - Adaptador para tecnología CORBA sincrónico

El framework dispone de la clase `ServerCorba` que encapsula los artefactos para implementar un servicio, ejecutándose como un `Thread` aparte. Basta que el Servidor arranque una instancia del Adaptador `ServerCorba`, registrando previamente la instancia que implementa el servicio (`ContractImplementation`).

3.4.4.4.2 Interfaz Asincrónica

El cliente que procesa la invocación asincrónica, construye el objeto `Invocation` (con los detalles de la invocación), lo encapsula en un mensaje y se lo envía al canal de mensajería asociado al Servicio destinatario de la invocación. Se utilizan *Event Channels* del Servicio *Notification Service* de CORBA.

El Adaptador de la implementación del servicio se apoya en un servicio CORBA de tipo "Listener" que consume mensajes del canal de Eventos asociado en modalidad "Push".

El framework provee de una implementación del Adaptador asincrónico que puede ser utilizada por el usuario (ver Figura 35). `ServerCorbaAsinc` es una implementación del Adaptador asincrónico que se ejecuta como un `Thread` separado, a los efectos de poder consumir mensajes en modalidad "Push". Para utilizarlo basta con instanciar la clase que implementa el servicio (`ServiceA`) y registrarla en el Adaptador. Al arrancar el Adaptador automáticamente comenzará a consumir mensajes conteniendo invocaciones. Cada invocación es procesada utilizando el servicio registrado (que implementa `IAdapter`).

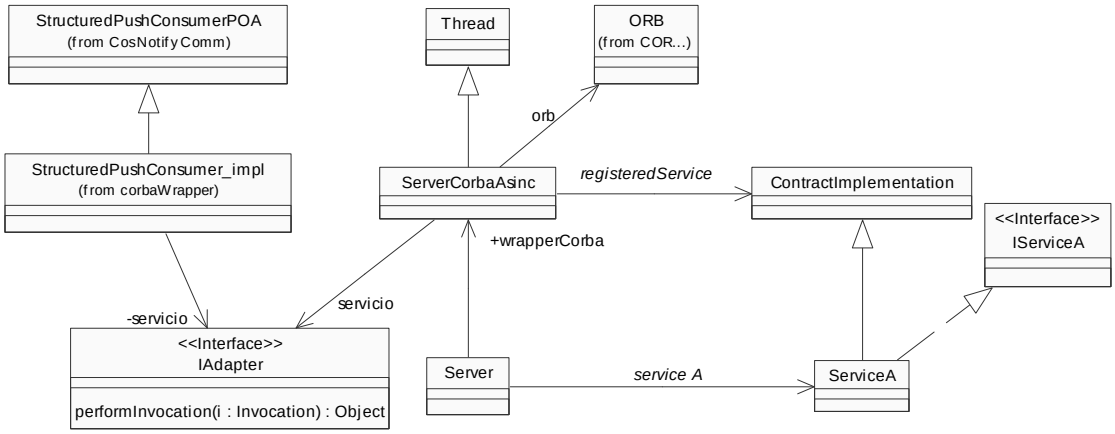


Figura 35 - Adaptador para tecnología CORBA asincrónico

3.4.5 Matriz de interoperabilidad entre Clientes y Servidores

A continuación se ilustran las distintas formas de interacción entre usuarios de Contratos (clientes) y las implementaciones de Contratos disponibles (servidores).

Desde la perspectiva del "Cliente", la arquitectura propuesta dispone de los siguientes Entornos de Ejecución.

- Entorno EJB en Application Server (`RunTimeEnvironmentEjb`)
- Servicio CORBA "standalone" (`RunTimeEnvironmentCorba`)
- Cliente Java Remoto (`RunTimeEnvironmentClient`).

.

Las implementaciones de contratos corren sobre alguna de las plataformas mencionadas anteriormente.
La plataforma "Cliente java remoto" en principio no tiene el cometido de poder albergar implementaciones de contratos (para ser invocados en forma sincrónica). Sin embargo, sí soporta una interfaz de Notificación asincrónica.

A continuación se muestra una matriz que muestra las posibles interacciones que se pueden llevar a cabo entre las distintas plataformas. Como se puede observar, cada fila de la matriz se corresponde con un Entorno de Ejecución. En este sentido, para cada Entorno de Ejecución se muestran las posibilidades de interacción con Contratos implementados en distintas plataformas. En la matriz figuran tres grandes columnas que se corresponden con la tecnología utilizada en la implementación de contratos (EJB, CORBA, CLIENTE). Para cada tecnología utilizada en el servidor, se divide el análisis en los casos Sincrónico y Asincrónico.

A modo de ejemplo, en el caso de una implementación de contrato ubicada en el contexto de un Servidor de Aplicaciones EJB para comunicarse sincrónicamente con una implementación de contrato dentro de la misma plataforma, se utilizan los dispositivos Adaptadores *Session Bean* propios de la arquitectura EJB. En el caso asincrónico (como muestra la figura), se utiliza un dispositivo Adaptador que consiste en un *Message Driven Bean* de la arquitectura EJB, conectado a un Tópico de mensajería JMS.

Para cada caso, la matriz muestra los recursos utilizados en la implementación correspondiente.

Entorno de ejecución	**Patrón de Interoperabilidad usado en el Destino**					
	EJB		**CORBA**		**CLIENTE**	
	SINC	**ASINC**	**SINC**	**ASINC**	**SINC**	**ASINC**
EJB	SBL/SBR	MDB	JCA/CORBA	JCA/CORBA/NS	-	clienteJMS
CORBA	SC/RMIIIOP	ClienteJMS o BridgeNS/JMS	IIOP	NS	-	NS
CLIENTE	SBR	clienteJMS	IIOP	NS	-	clienteJMS/NS

Notación utilizada:

SBL	Acceso vía EJB usando Session Beans con interfaces Locales
SBR	Acceso vía EJB usando Session Beans con interfaces Remotas
MDB	Acceso vía EJB usando Message Driven Beans (conectados a tópicos JMS)
clienteJMS	Acceso vía Mensajería JMS en forma directa. (El cliente receptor se suscribe al tópico asociado)
JCA/CORBA	Se utiliza un "conector" JCA hacia un recurso externo que oficia de cliente en la plataforma CORBA.
NS	Acceso a través del Servicio Notification Service de CORBA.
SC/RMIIOP	El acceso se realiza a través de un Servicio brindado a nivel del application server accesible vía IIOP.
BridgeNS/JMS	Acceso es a través de Notification Service de CORBA. Un dispositivo "Bridge" entre mensajería NS y JMS trasiega los mensajes a la plataforma JMS
IIOP	Acceso directamente utilizando IIOP

3.4.6 Acoplamiento entre componentes

Una de las premisas de la presente arquitectura consiste en lograr un desacoplamiento débil entre los Building Blocks.

En el contexto de la presente Arquitectura el grado de desacoplamiento está determinado por:

- Características de Objetos intercambiados a través de los Contratos.
- La forma en que se codifican los paquetes de Invocación y las respuestas requeridas eventualmente.
- La forma en que se transportan los paquetes de invocación.

En cuanto a los mecanismos de transporte previstos por la arquitectura, se posibilitan interacciones sincrónicas y asincrónicas entre implementaciones de contratos contenidas en Building Blocks. La interacción asincrónica (utilizando un sistema de Mensajería) implica por definición un mayor grado de desacoplamiento que las interacciones sincrónicas.

En cuanto a la codificación de paquetes de Invocación se consideraron las alternativas siguientes:

- Paquetes de Invocación codificados en XML. Es la opción que implica un mayor grado de desacoplamiento pues posibilita la independencia total respecto a tecnologías especificas.
- La otra alternativa es utilizar *JavaValueTypes* (tipos de Java Serializables) para armar los "paquetes" de Invocación. En este caso la alternativa es dependiente en cierto grado de la tecnología (implica el uso de Java) pero es flexible respecto a la posibilidad de interconexión de distintas plataformas de interoperabilidad que utilicen Java como por ejemplo CORBA o EJB.

En la implementación del presente trabajo se aborda la interoperabilidad utilizando exclusivamente JavaValueTypes. No es objeto del presente estudio abordar el mapeo de tipos a XML. Existe la especificación de OSS/J de Java [35] que define el lenguaje XML a emplear. TMForum tiene el tema en estudio. Debería considerarse SOAP como una alternativa

Se aplica el patrón "ValueType" en la formulación de los contratos. Esto implica que los contratos necesariamente intercambian datos por valor. Esto eventualmente implica una restricción en cuanto al uso de la potencialidad de la plataforma distribuida.

Por ejemplo un contrato no debería devolver o recibir referencias a objetos reales distribuidos. En su lugar se aplica el patrón ValueType y las entidades afectadas se intercambian "por valor". Por ejemplo para las entidades del sistema, se pueden intercambiar objetos por valor que representan la clave de la misma. En

este sentido la entidad es accesible a través de un manejador de la misma (donde las operaciones a realizar involucran necesariamente especificar explícitamente la identidad o "clave" de la misma).

3.4.7 Mecanismos de Expansión del Framework

Para considerar nuevas plataformas basta disponer del nuevo `Proxy` y el descriptor del punto de acceso del servicio que contiene la información para poder procesar la invocación, además de dar de alta un nuevo patrón de interoperabilidad a considerar.

Es posible definir un nuevo entorno de ejecución o agregarle la nueva funcionalidad a algún entorno de ejecución existente (`RuntimeEnvironment`). Basta modificar el constructor del entorno de ejecución para que incluya el nuevo `Proxy`.

Por ejemplo, es posible definir entornos de ejecución mixtos que soporten varias tecnologías simultáneamente. Se puede definir un entorno de ejecución Cliente (RMI) y CORBA, agrupando los recursos de ambas plataformas. En este caso se define un orden de prioridad para los dispositivos `Proxy` en función del orden asignado en el constructor del entorno de ejecución. La invocación se llevará a cabo con el primer `Proxy` compatible con el descriptor de la implementación del contrato que se desea ejecutar.

3.5 Conclusiones

En este capítulo se ha presentado inicialmente la arquitectura propuesta en términos neutrales. Posteriormente se hicieron consideraciones respecto a su aplicación en plataformas específicas como lo son la arquitectura J2EE y CORBA considerando tanto interacciones sincrónicas como asincrónicas.

Particularmente se ha puesto el acento en resolver los problemas de interoperabilidad asincrónicos entre diferentes tecnologías, lo cual es un requerimiento importante para los escenarios de aplicación de la presente Arquitectura (ver el Caso de Estudio en el siguiente capítulo donde se aborda la problemática de un sistema de gestión de fallos de una red).

Es de destacar que el énfasis del trabajo está situado en la formulación adecuada de un servicio de invocación que permita esconder los detalles de la implementación “destino” de la invocación. De esta manera la implementación, si bien queda escrita en un lenguaje específico (Java en el caso del presente trabajo), es independiente de las características de los contratos utilizados. Esto posibilita entre otras cosas el reuso del código de la implementación en distintos contextos. Por ejemplo, una implementación de un contrato podría cambiarse de plataforma sin afectar los clientes del mismo.

Hay aspectos de la arquitectura como Seguridad y Control de Transacciones que no son abordados y debe extenderse el estudio en dichas áreas. Tampoco está dentro del alcance del presente trabajo la gestión de metadata asociada a precondiciones y poscondiciones y la implementación de la verificación de las mismas.

Una de las alternativas para manejar la especificación de precondiciones es utilizar el lenguaje de scripting ECMAScript (de manera de poder usar la expresión directamente en el motor intérprete).

En este sentido, se estima como factible extender el framework para manejar un “entorno de ejecución” de scripting ECMAscript y en este caso sería posible formular expresiones para precondiciones y poscondiciones que permitieran ser verificadas en tiempo real a través de un motor de scripting que corra bajo el entorno del framework.

Capítulo 4

4 Caso de estudio

4.1 Introducción

Se pretende abordar las problemáticas de Gestión de Fallos. El objetivo del caso de prueba no es dar una solución a estas problemáticas sino por el contrario, relevar requerimientos generales y sobre esta base diseñar un sistema lógico que brinde un marco arquitectónico apropiado sobre el cual se pueda construir un sistema que responda a las necesidades reales.

En el desarrollo del caso de prueba se pretende aplicar la arquitectura propuesta anteriormente.

4.2 Los Procesos de Gestión de Fallos

La tarea principal es brindar la información de alarmas recibidas a los operadores del sistema y eventualmente otros procesos de la organización que requieran de esta información.

4.2.1 Complejidades de la temática

El problema de gestión de fallos de red requiere la interacción con diferentes tecnologías (SDH, ATM, ADSL, etc.), con diferentes tipos de equipamiento y la integración de varios sistemas legados.

La gestión de fallos de red debe ser capaz de discernir a partir del alto flujo de alarmas que se genera cuando se produce un fallo, cual fue realmente la causa del problema. Este alto flujo de alarmas se genera por la recursividad que existe entre recursos y servicios en las redes de telecomunicaciones. La superposición de sistemas da lugar a esta recursividad (frame relay sobre la capa de transporte SDH y a su vez sobre la red óptica). Por ejemplo, un servicio de datos constituido por dos módems HDSL conectados mediante recursos de transmisión, puede incluir conjunción de tramos de fibra, pares de cobre, etc.

Es claro que frente a un fallo de alguno de los equipos involucrados se generarán una serie de eventos (o alarmas) de varios de los equipos involucrados generando así el torrente de alarmas. Es necesario determinar la causa principal de fallo del servicio mediante la correlación de esos eventos y poder establecer mecanismos de filtrado que permitan distinguir entre el gran flujo de eventos aquellos importantes o determinantes de fallos en el servicio.

El análisis resulta más complejo si se trata de considerar la realidad donde los servicios pueden ser brindados por estructuras más o menos autónomas dentro de la empresa o en empresas diferentes. Es posible que una empresa se encargue de brindar el servicio de datos usando módems HDSL y para interconectarlos contrate servicios de transmisión de otra empresa. Esta segunda empresa brindará el servicio de transmisión y podrá hacerlo con equipamiento propio o a su vez sub-contratar servicios de una tercera empresa.

En este contexto, aparece más claramente el concepto de "servicios que usan servicios" y teniendo en cuenta que esos servicios pueden potencialmente ser brindados por diferentes empresas surge claramente la diferenciación entre alarmas de servicio y alarmas de elementos de red.

4.2.2 Características generales de los sistemas de gestión de fallos

Se pretende que el sistema permita dar una visión unificada de todas las alarmas integrando todas las tecnologías existentes.

La tarea principal de este sistema es brindar la información de alarmas recibidas a los operadores del sistema involucrados y eventualmente otros procesos de la organización que requieran de esta información.

Se pretende que el sistema sea capaz de correlacionar alarmas a medida que son recibidas. El objetivo de la correlación de alarmas es reducir la cantidad de información que es desplegada a un operador encargado de supervisar la red, tratando de identificar los fallos más probables que dieron origen a las alarmas con el fin de tomar rápidamente medidas correctivas.

Para la correlación de alarmas, además del flujo de alarmas recibido de la red, otras fuentes de información son esenciales, como por ejemplo información sobre la topología de la red de los elementos generadores de alarmas.

Los usuarios que interactúan con este sistema tienen que ser capaces de poder extraer información de alarmas de red y de servicio. Dado que existen distintos perfiles de usuarios, el sistema debe proveer diferentes vistas de la información y estas tienen que poder ser configurables por el usuario.

Además el sistema debe ser capaz de guardar un histórico de las alarmas que se producen y permitir a los usuarios realizar consultas sobre el mismo.

4.2.3 Definiciones

4.2.3.1 Notificaciones de Alarmas

Se distinguen los términos: evento, alarma y fallo.

Un **evento** es la ocurrencia de un cambio en determinados atributos de un objeto gestionado. El objeto gestionado puede ser un elemento de red y también puede generalizarse definiendo eventos en servicios o en otros objetos gestionables de diversa índole.

Las **alarmas** son las notificaciones de información de ocurrencia de eventos. Estas notificaciones son normalmente generadas en forma espontánea por los objetos gestionados. Las alarmas pueden o no representar la ocurrencia de un error o mal funcionamiento del equipo.

Se dice que hay un **fallo** en un objeto gestionado cuando se producen desvíos en el comportamiento esperado del sistema.

4.3 Análisis del Caso de Estudio para el Sistema de Supervisión

4.3.1 Particularidades del proceso de Gestión de Fallos considerado

Es de destacar que el escenario planteado para el presente estudio es tal que plantea algunas limitaciones. En particular, solo se considerará la comunicación desde los Elementos de Red, fundamentalmente a través del flujo de reporte de eventos y alarmas emitidos por los propios elementos de red, canalizados por sus respectivos sistemas de gestión.

Por esta razón se excluye del alcance del presente trabajo, las actividades que requieren "actuación" sobre los elementos de red (a través de sus respectivos sistemas de gestión). En este sentido no se consideran las actividades vinculadas al "testing" y acciones correctivas (como reconfiguración dinámica de elementos de red), actividades incluidas en el proceso "mantenimiento y restauración de red" de TOM. En un escenario ampliado, donde se disponga de facilidades de actuación sobre elementos de red, las actividades de reconfiguración y "testing" deberían ser incluidas.

Tampoco se considera el análisis de correlación de alarmas (posibilidad de analizar más de una alarma en forma grupal, teniendo en cuenta el escenario topológico al que pertenecen los objetos generadores de dichas alarmas).

4.3.2 Descripción General

Los equipos remotos gestionados envían las notificaciones de alarma al módulo "Gestor de Equipos" (ver Figura 36). Este módulo realiza la adaptación requerida para manejar un formato común de notificaciones de alarma a lo largo del sistema. Esta alarma se almacena en el registro histórico general ("Gestión Log Histórico")

Posteriormente la alarma es enrutada al módulo "Gestión de Red". La función llevada a cabo internamente es la de filtrado y recategorización.

Esta funcionalidad se implementa sobre la base de un motor de reglas interno. La evolución de la alarma en el sistema está definida en la base de datos de reglas. Cada regla especifica una condición que debe satisfacer la alarma para que la misma se aplique. Las reglas indican complementariamente la transformación a realizar en la alarma (por ejemplo recategorización) y el destino de la misma.

Finalmente las alarmas llegan a los módulos de "Supervisión Especializada" que se encargan del mantenimiento del estado de los nodos supervisados. Cada mensaje de alarma implícitamente o explícitamente está asociado a un nodo de supervisión (conceptualmente un nodo de supervisión puede ser el objeto que emite el mensaje o una entidad del sistema afectada). La llegada del mensaje implica un cambio de estado del nodo supervisado por lo que se requiere la actualización de la lista de alarmas que están activas para el nodo.

La idea es que puede existir más de una forma de considerar nodos de supervisión (en función de diferentes perfiles de supervisión) y esto determina eventualmente la existencia de varios módulos de Supervisión Especializada. El módulo de Gestión de Red debería hacer llegar los mensajes de alarmas en forma apropiada a cada uno de los módulos de Supervisión Especializada.

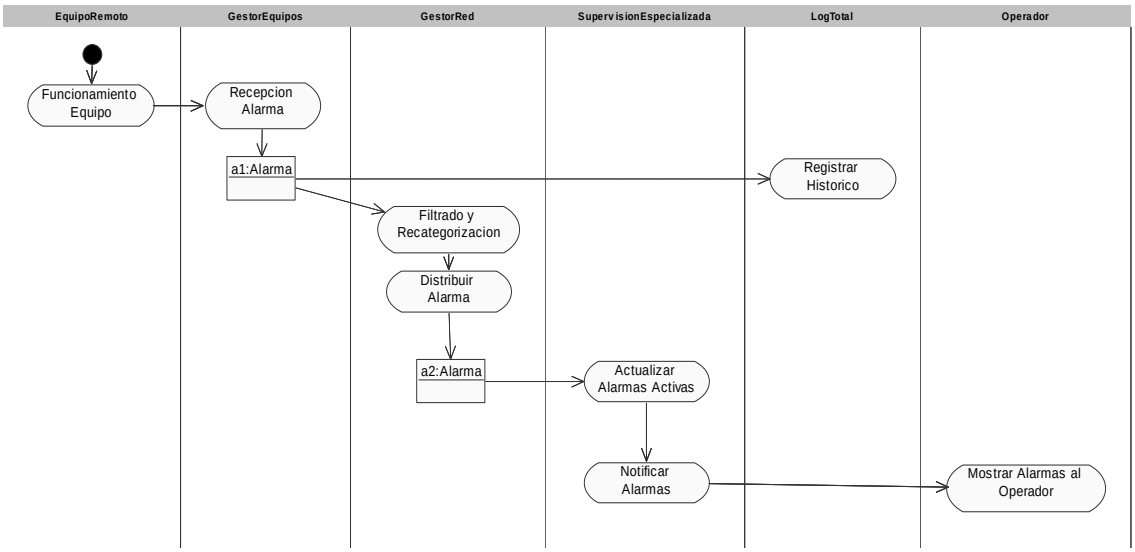


Figura 36 – Diagrama de Actividades vinculadas al procesamiento de alarmas.

4.3.3 Diseño del sistema lógico neutral

A continuación se ilustra sintéticamente la descomposición del sistema general en subsistemas. (para ver detalles sobre el análisis de casos de uso del sistema ver el Anexo correspondiente).

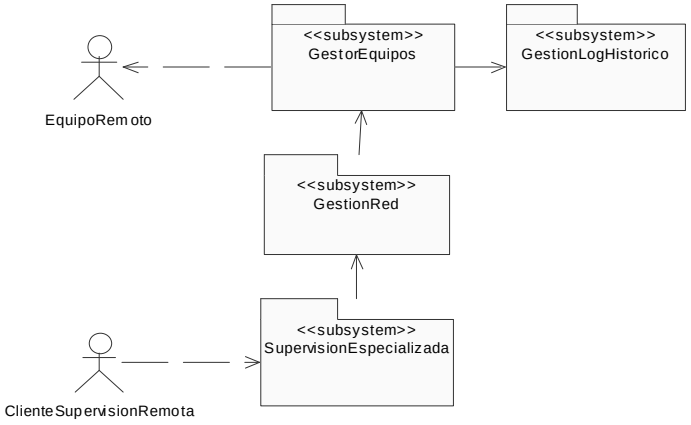


Figura 37 - Subsistemas para el caso de estudio

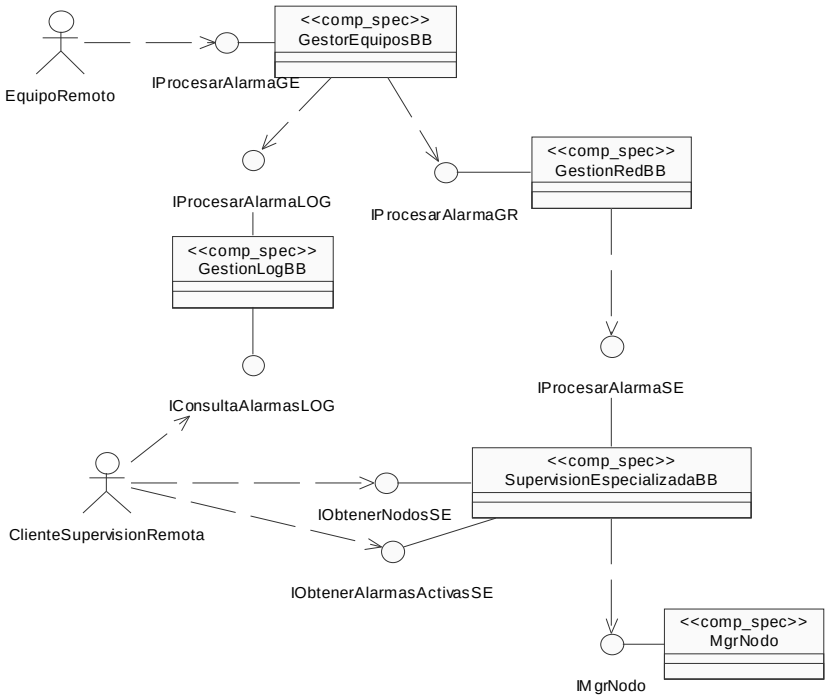


Figura 38 - Diagrama de Componentes para el Caso de Estudio

Se plantea resolver el problema en base a un conjunto de Building Blocks que implementan los servicios requeridos. La formulación inicial es en términos neutrales definiendo los contratos asociados a cada uno de los servicios. En la figura siguiente se muestra un diagrama de componentes UML donde se observan los "Building Blocks" requeridos y sus interfaces y las dependencias entre ellos y actores externos (ver Anexo).

4.3.4 Implementación

Como se puede ver en la figura siguiente, el proceso de implementar servicios a partir del diseño neutral formulado anteriormente es directo. Cada interfaz prevista para cada Building Block se mapea a una implementación derivada de `ContractImplementation` provista por el propio framework.

El código dentro de la implementación permanece casi neutral, en el sentido de que las dependencias con otros módulos expresadas a través de invocaciones están formuladas en términos de artefactos propios del framework que esconden las particularidades de cada tecnología.

Para que la implementación pueda ser activada se requiere de un Adaptador compatible con el framework que pueda ser capaz de invocar efectivamente la implementación. El propio framework provee de un conjunto de implementaciones por defecto de adaptadores apropiados que pueden ser utilizados a los efectos de facilitar la creación de servicios.

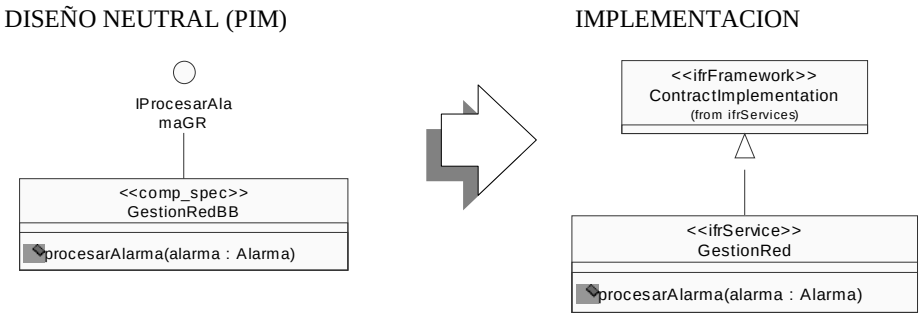


Figura 39 - Pasaje de Diseño neutral a Implementación de contratos

A continuación se resumen algunos aspectos de las implementaciones llevadas a cabo.

A los efectos ilustrativos, el Building Block"Gestor Equipos" aparece instanciado dos veces a los efectos de mostrar la ocurrencia de distintos sistemas de supervisión integrados al sistema. Cada gestor de equipos puede manejar muchos equipos. Por simplicidad en el caso de estudio se ha asociado un equipo remoto con cada Gestor de Equipos.

Como se puede ver en el diagrama (Figura 40), los Building Blocks de Gestión de Red y Supervisión Especializada se instalaron (deploy) en un servidor de aplicaciones EJB. Lo mismo ocurre con los servicios de EIT requeridos. Los Building Blocks "Gestor de Red" y "Equipo Remoto" se implementaron bajo servicios CORBA, dada la conveniencia o la necesidad de su ubicación remota. El cliente se implementa como un "Bean" de apoyo a servicios JSP.

En el diagrama (Figura 40) las flechas indican el sentido de las interacciones. Las flechas gruesas están indicando interacciones asincrónicas mientras que las flechas delgadas muestran interacciones sincrónicas. En particular, en el caso asincrónico (flechas gruesas) se ha dispuesto que el trazo sólido indique uso de mensajería apoyada en el Servicio "Notification Service" de CORBA, mientras que el trazo punteado señala el uso de mensajería JMS disponible en el servidor de aplicaciones. Los círculos dibujados, agrupan componentes que pertenecen al mismo nodo (desde el punto de vista de la arquitectura). En particular todos los componentes del Servicio de Aplicaciones EJB pertenecen al mismo nodo lógico.

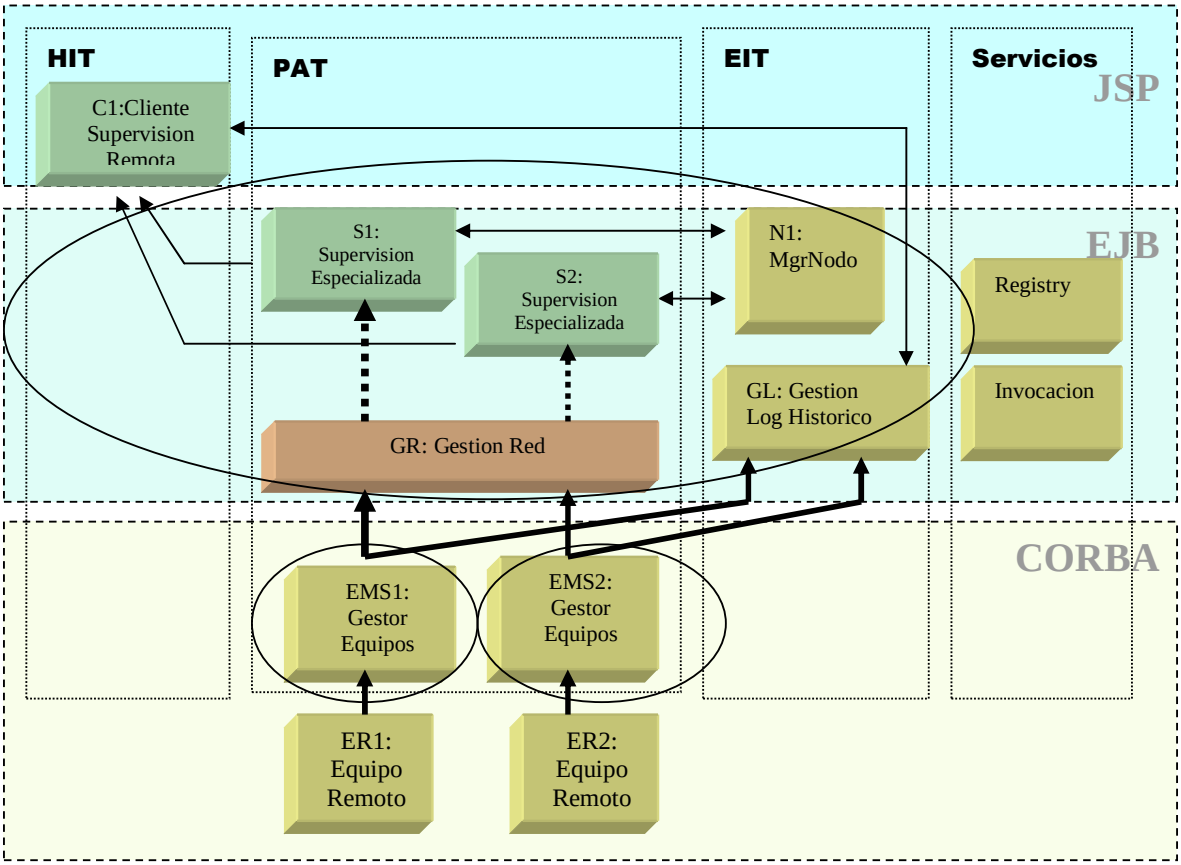


Figura 40 - Implementación del Caso de Estudio

4.3.5 Gestores de equipos y agentes remotos

En el caso de estudio se utilizaron distintos esquemas para modelar equipos remotos. Se utilizaron tanto un generador virtual de mensajes de alarma, como "gateways" apropiados hacia equipos reales, de manera de obtener tráfico de "alarmas" en condiciones más realistas.

Dada la disponibilidad de sistemas integrados a Dominios JIDM *(Joint Interdomain Management Network)*, se modificó un Gestor CORBA JIDM con cambios menores de manera de encaminar alarmas usando la arquitectura propuesta.

4.3.6 Implementación de Building Blocks

Se ilustra a continuación la implementación de algunos Building Blocks utilizados.

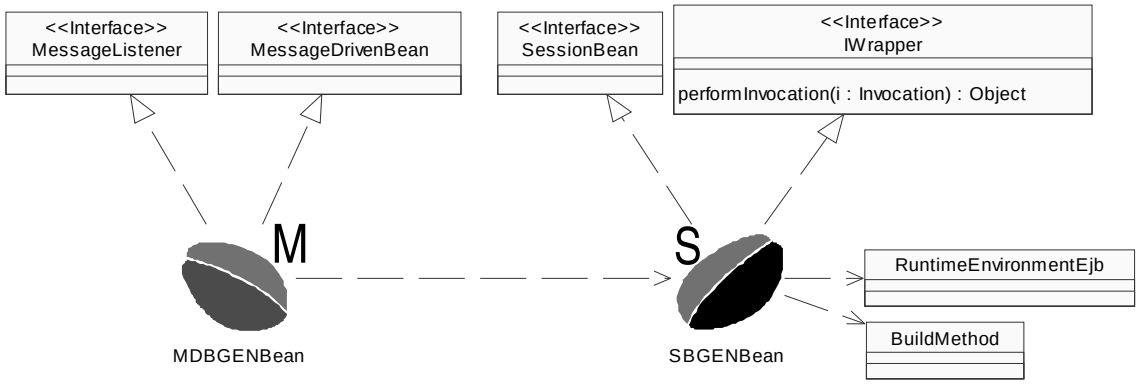
4.3.6.1 Implementación de Building Blocks en la plataforma EJB.

Se toma como ejemplo el Building Block "Gestión Red" que implementa la interfaz `procesarAlarma`. Para la implementación completa del servicio es necesario:

- implementar el servicio.
- implementar el Adaptador (o utilizar algún adaptador ofrecido por defecto a nivel del framework).

Para implementar el servicio a partir del diseño neutral basta con crear una clase que implemente la interfaz del servicio requerido y que derive de `ContractImplementation` (como se ha expresado anteriormente).

Para construir el punto de acceso en la plataforma EJB, se utiliza un *Session Bean* (Stateless en este caso) que implementa la interfaz `IAdapter`.



La invocación se efectiviza por la ejecución del método `performInvocation` del Adaptador.
El objeto `Invocation` (pasado como parámetro) contiene los detalles de la invocación requerida, contiene en particular la información de la clase que implementa el servicio. La clase `BuildMethod` provista por el framework provee de la lógica para construir la invocación real apoyado en el entorno de ejecución `RuntimeEnvironmentEjb`.

Para implementar un punto de acceso asincrónico se crea un *Message Driven Bean* asociado a un canal de mensajería JMS *(Topic)*. Con el objeto `Invocation` recibido (en el mensaje) se lleva a cabo una invocación sincrónica utilizando el *Stateless Session Bean* asociado.

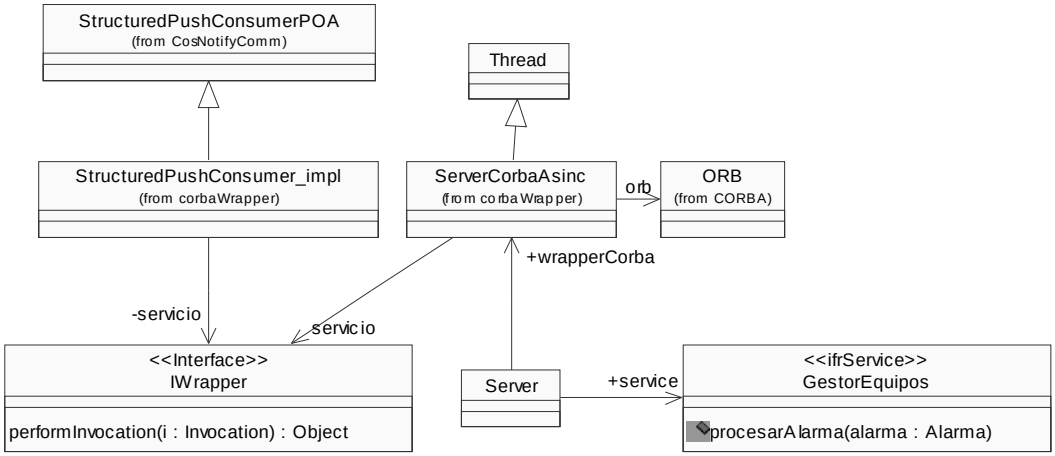
4.3.6.2 Puntos de Acceso alternativos en el Nodo.

Desde el punto de vista de procesamiento distribuido, el componente "Gestión de Red" está instalado en el nodo "Servidor" (que coincide con el propio Servidor de Aplicaciones EJB). Este nodo está equipado con puntos de acceso alternativos a los efectos de dar servicio a usuarios de la plataforma CORBA. En particular se aceptan llamadas sincrónicas a través de un servicio CORBA destinado a este cometido (implementado como un *MBean* en el servidor de Aplicaciones). También se dispone del servicio que consume mensajes de mensajería manejada por el Servicio *Notification Service* de CORBA. De esta manera los entornos de ejecución CORBA (en el sentido estricto) que no manejan la arquitectura EJB tienen puntos de acceso alternativos para los servicios instalados en la arquitectura EJB (tanto en forma sincrónica como asincrónica).

4.3.6.3 Implementación de Building Blocks en plataforma CORBA

El framework proporciona los Adaptadores requeridos. Basta con implementar el servicio tal como se indicó anteriormente. Por ejemplo el componente `GestorEquipos` se construye implementando la interfaz

`IProcesarAlarmaGE` y extendiendo `ContractImplementation`. Al igual que la arquitectura EJB, se pueden implementar servicios que mantienen el estado o no. En este caso se utilizan servicios que mantienen el estado. Para ello es suficiente instanciar la clase que implementa el servicio, registrarlo en el Adaptador provisto por el framework (`ServerCorbaAsinc`) y arrancarlo (corre como un Thread que consume mensajes de un canal de *Notification Service* en modo "Push").



4.4 Resultados obtenidos con el Caso de Estudio

Al presente, el caso de estudio está operativo, manejando la funcionalidad mencionada anteriormente (utilizando tráfico de alarmas proveniente de agentes reales). Para la implementación se utilizó JBOSS como plataforma J2EE, mientras que se utilizó ORBACUS como plataforma CORBA.

Es de destacar que ha sido posible la integración de nuevos Building Blocks en forma gradual y efectuar cambios sobre sus implementaciones sin alterar el sistema. Esto muestra de alguna manera el alto grado de desacoplamiento que existe entre Building Blocks, además del potencial de escalabilidad del sistema.

Con pocos cambios ha sido posible integrar gestores de equipos remotos, sin que los mismos tuvieran la necesidad de conocer los detalles tecnológicos involucrados como por ejemplo, la dirección real del servicio al cual conectarse, la tecnología empleada y el patrón de interoperabilidad (estos detalles son manejados en forma transparente por el propio framework y la configuración establecida).

El bloque de Gestión de Red, maneja un motor de reglas interno para enrutar los mensajes de alarma en función de las condiciones expresadas en cada regla a distintos destinatarios. En este sentido la implementación resultó limpia y sencilla ya que existe una relación directa entre destinos lógicos (expresados en reglas) y el nombre de las implementaciones de contratos que los procesan.

Es posible recrear diversos escenarios de interoperabilidad, simplemente variando la configuración. Llamadas asincrónicas pueden hacerse asincrónicamente y viceversa cuando sea adecuado.

También es posible con bajísimo costo reubicar las implementaciones en otra plataforma con alto nivel de reuso. Recordemos que las dependencias con otros módulos expresadas en la implementación, están formuladas en términos de artefactos propios del framework que esconden las particularidades de cada tecnología, lo cual hace posible un alto nivel de reuso.

Capítulo 5

5 Conclusiones

5.1 Resumen

Las empresas de telecomunicaciones están atravesando enormes desafíos caracterizados por una alta competencia y la necesidad de poder brindar respuestas rápidas a pedidos del mercado, implementando nuevos servicios de valor agregado en forma ágil.

Son valiosos los aportes de organizaciones como ITU, TINA-C y TMForum que hacen un análisis de los procesos que se dan en el interior de la empresa, identificando cadenas de valor (comenzando el análisis desde la perspectiva del cliente y continuando con los procesos requeridos para llevar a cabo los distintos servicios ofrecidos).

Actualmente el área de *e-business* está en auge y se intenta aplicar estas tecnologías al área de OSS *(Operation and Support Systems)*. Se busca automatizar los procesos de negocios incluyendo la interacción con los procesos del negocio de los clientes y proveedores. Se usan tecnologías para manejar flujos de procesos (Ej. Workflows) y sistemas ensamblados utilizando componentes COTS *(Comercial off-the-shelf)*, con interfaces bien definidas, de manera que sea posible la reconfiguración del sistema ante cambios de requerimientos.

La posibilidad de construir sistemas basados en COTS disponibles requiere el uso de modelos estandarizados.

En general no es posible reemplazar un conjunto de OSS pues existen grandes inversiones en sistemas que están operativos, por lo que el escenario marca la necesidad de adaptación de modelos a los efectos de utilizar componentes disponibles. La aplicabilidad de este esquema depende fundamentalmente de los costos de adaptación al modelo estándar.

Si se requiere implementar nuevas funcionalidades (no disponible comercialmente o no aplicable) es deseable que los desarrollos se hagan teniendo en cuenta algún modelo estandarizado (en la medida que sea posible). En cualquier caso es fundamental el diseño del sistema en términos neutrales y respondiendo a modelos estándares (o cercanos a los mismos). Existen varias metodologías para el diseño basado en componentes siendo una de las más interesantes MDA. El resultado del diseño del sistema es un conjunto de especificaciones de servicios a encapsular en *Building Blocks*.

La propuesta arquitectónica que se presenta pretende dar un marco para generar sistemas basados en *Building Blocks*, resolviendo el problema de interacción entre ellos a través de un vehículo de comunicaciones común pudiendo esconder distintos patrones de interoperabilidad (sincrónico, asincrónico, etc.).

Esta propuesta facilita la implementación de los servicios encapsulados en *Building Blocks*. Dado el diseño lógico neutral, por ejemplo utilizando modelos independiente de la plataforma (PIM en el enfoque MDA) es muy sencillo derivar dichos modelos a una implementación concreta en el contexto de la arquitectura propuesta. Basta que las implementaciones deriven de la clase base para implementaciones de contratos (`ContractImplementation`) (heredando de esta manera los servicios provistos por el framework) además de implementar las interfaces requeridas, propias del servicio brindado.

Los *Building Blocks* incluyen uno o más puntos de acceso al servicio. Estos puntos de acceso son artefactos propios de la plataforma en que finalmente se implementan (adaptadores). Estos dispositivos son los responsables de la activación real de cada uno de los servicios disponibles.

Es de destacar que el desarrollador puede incluir sus propios dispositivos adaptadores o eventualmente utilizar dispositivos genéricos provistos por el propio framework (en este sentido, la implementación no arranca de cero).

La clara separación que la arquitectura establece entre la implementación de servicios y las formas de interoperabilidad utilizadas permite esconder con facilidad la aplicación de “Patterns” utilizados frecuentemente en el ámbito de Servidores de Aplicaciones (algunos de ellos requeridos para mejorar el

rendimiento de los mismos), de manera de obtener implementaciones performantes despojadas del ruido que significaría la inclusión (en el contexto de la propia implementación) de artefactos auxiliares requeridos para mejorar performance.

Este encare no busca proponer un nuevo framework que resuelva por sí sólo una multiplicidad de situaciones diferentes, sino que por el contrario, se presenta este enfoque como una estrategia donde el propio constructor del sistema transforma el framework básico de acuerdo a sus necesidades (especialización) permitiendo encapsular problemas de interoperabilidad tecnológicos, mejoramiento de performance y la propia configuración requerida por el usuario.

En síntesis, la propuesta arquitectónica de interoperabilidad entre *Building Blocks* que se presenta permite achicar la distancia entre el diseño lógico neutral y su implementación y por otro lado facilitar el reuso a nivel de aplicación además de constituir un refugio ante cambios tecnológicos (al mantener desacoplada la implementación de las particularidades de la plataforma).

Con la finalidad de verificar la aplicabilidad del framework, se desarrolló un caso de estudio que aborda el problema de la Gestión de Fallos en la Red. El cometido fundamental es el procesamiento de mensajes de alarma que provienen de equipos remotos a los efectos de que dicha información quede disponible para los operadores del sistema.

Actualmente, el caso de estudio está operativo, con tráfico de alarmas proveniente de agentes CORBA reales (permitiendo obtener alarmas originadas en equipos remotos).

Es de destacar que ha sido posible la integración de nuevos *Building Blocks* en forma gradual y efectuar cambios sobre sus implementaciones sin alterar el sistema, mostrando el grado de desacoplamiento ofrecido por la arquitectura. Esto ha sido posible porque cada implementación que requiere de servicios externos no necesita conocer los detalles de la implementación de los mismos.

Otra experiencia realizada fue la de reusar implementaciones en el contexto de otra plataforma. Solo se requirieron ajustes menores para el reensamblado del *Building Block* en la nueva plataforma. Cambia el Adaptador que activa el servicio, pero la implementación en sí del servicio se reusa directamente. Entre las experiencias realizadas se llevaron implementaciones sobre plataforma EJB a CORBA y viceversa, (además de variar el patrón de interoperabilidad utilizado, sincrónico o asincrónico).

5.2 Trabajo Futuro

La oferta tecnológica de esquemas de interoperabilidad es abundante y cambiante. Actualmente están en auge las tecnologías Web Services y SOAP. Es muy probable que puedan surgir numerosos servicios aplicados al dominio de Gestión de Redes y Servicios disponibles en esta plataforma. En este sentido es importante extender el framework a estas tecnologías para no quedar fuera de un escenario que presente numerosos beneficios.

Hay dos líneas de trabajo a considerar que no son excluyentes en este sentido:

- Posibilitar que los servicios brindados por los distintos Building Blocks sean accesibles en forma nativa para usuarios de Web Services. En este sentido los WSDL deberían generarse a partir de la información de los contratos registrados (este enfoque posibilitaría el uso de servicios desde plataformas usuarias de Web Services).
- El otro encare complementario pasa por definir un nuevo entorno de ejecución del framework, que se maneja fundamentalmente usando el protocolo SOAP. Este encare sería simétrico respecto a la posibilidad de usar cualquier servicio desde cualquier plataforma. Presenta como desafío, la necesidad de “personalizar” el framework de manera de posibilitar el mapeo entre tipos de datos y XML.

Un aspecto no implementado hasta ahora es el de verificación de precondiciones de Contratos (previo a la invocación efectiva de los mismos). Una implementación posible puede apoyarse sobre la formulación de precondiciones utilizando expresiones ECMAScript (plataforma de scripting).

Se estima como factible la definición e implementación de un entorno de ejecución ECMAScript que posibilite verificar en tiempo real las precondiciones requeridas.

De la misma manera se requiere trabajo adicional para cubrir aspectos vinculados al control de transacciones, aspecto que en la presente propuesta está apoyado en las posibilidades de la plataforma elegida. Fundamentalmente se requiere resolver el problema de control de transacciones interplataformas. Se plantea como muy probable el caso de transacciones de configuración que arrancan en el entorno de ejecución EJB y que involucran la configuración de servicios CORBA remotos.

Finalmente, es necesario flexibilizar la gestión de los propios Building Blocks a lo largo del ciclo de vida de los mismos. Se requiere mejorar la gestión de configuración, instalación y supervisión de los Building Blocks operativos. Eventualmente puede ser necesaria una nueva formulación del Servicio de Registro, atendiendo a mejorar los aspectos de seguridad (validación de credenciales de usuarios), soporte de las nuevas características de la configuración y federación de Servicios de Registros (para atender escenarios distribuidos complejos).

De cualquier manera la evolución futura de la arquitectura no debería perder la característica de su simplicidad actual. Esta condición es esencial para posibilitar el mantenimiento de la misma y que el propio desarrollador pueda “adaptar” el framework propuesto a sus propias necesidades.

6 ANEXO: Análisis de casos de uso del “Caso de Estudio”

6.1 Metodología.

A los efectos de definir el conjunto de Building Blocks (componentes) requeridos para la construcción del Caso de Estudio y definir las interfaces que implementan, se siguen las principales guías de la metodología para el diseño de sistemas basados en componentes, propuesta por Andrés Vignaga y Daniel Perovich [48].

El objetivo central del caso de estudio, es validar la aplicabilidad del framework de interconexión. En este sentido, el diseño a nivel lógico de componentes requeridos presentado en esta sección no pretende definir los contratos en toda su extensión, limitando el alcance a definir interfases asociadas a cada componente.

6.2 Separación en Subsistemas Lógicos

La primer aproximación es estructurar subsistemas para cada uno de los actores internos en la cadena de procesamiento de las alarmas (ver Figura 41).

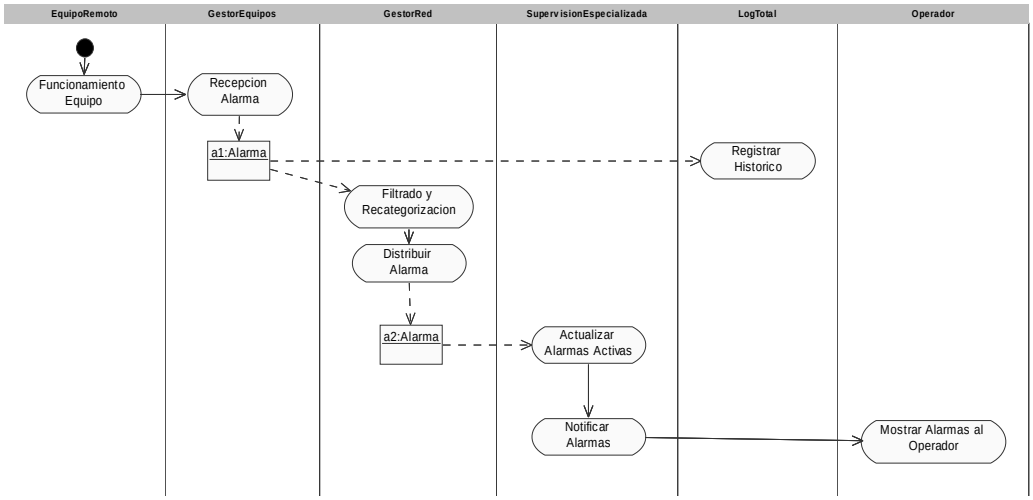


Figura 41 – Procesamiento de alarmas.

Una motivación para el encare a nivel de subsistemas es el hecho de que cada uno de los actores involucrados en el procesamiento de las alarmas puede encontrarse físicamente en lugares distintos. En este sentido el encare es abordar cada actor de la cadena del procesamiento de las alarmas como un subsistema en sí mismo.

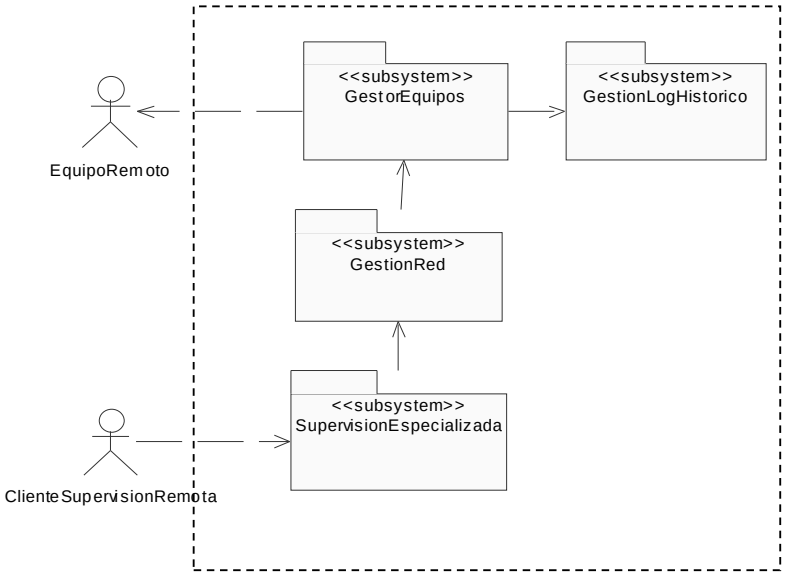


Figura 42 – Subsistemas Lógicos del Caso de Estudio.

6.3 Modelo Conceptual

Cada equipo esta subordinado a un "Gestor" de los mismos ("GestorElementos"). Cada equipo está ubicado en una Estación. El concepto de estación está asociado a la idea de lugar geográfico. Cada "AlarmaEquipo" está asociada a un equipo que la generó.

Los servicios del sistema están definidos en términos de los equipos utilizados. Para cada servicio también se especifican Alarmas de Servicio.

Cada “Supervisión Especializada” maneja un conjunto de "Nodos" (que eventualmente puede estar asociado al concepto de estación). Para cada nodo se maneja un conjunto de Alarmas Activas asociadas.

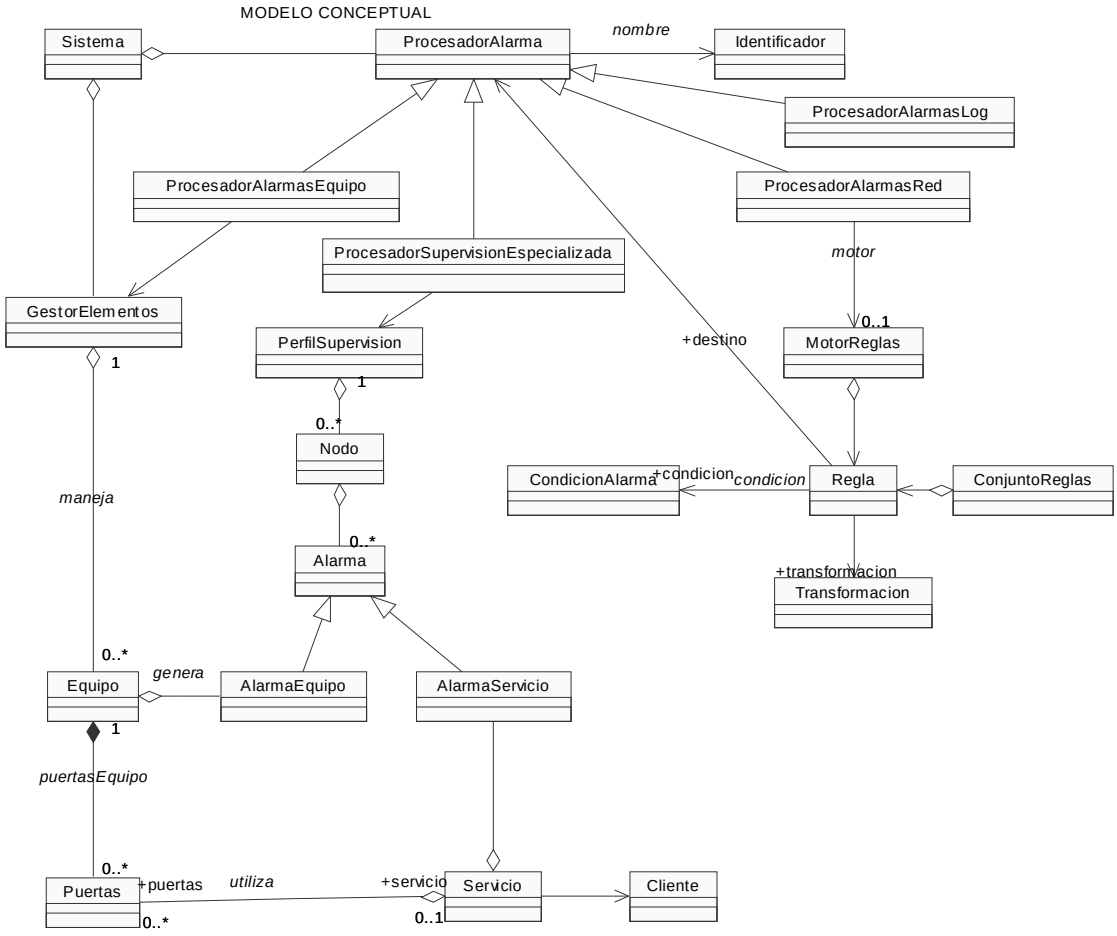


Figura 43 –Modelo Conceptual.

6.4 Requerimientos generales

A continuación se presentan algunas definiciones y se enumeran los requerimientos necesarios para el caso de prueba. Se considera que la frontera del sistema en estudio excluye la capa de interfaz humana, no se consideran requerimientos de “presentación” de la información.

6.4.1 Requerimientos

6.4.1.1 Recepción de alarmas

El sistema tiene que recibir todas las alarmas generadas por los elementos de red, integrando todos los elementos de red. Adaptando formato eventualmente

6.4.1.2 Aviso de alarmas

El sistema tiene que avisar inmediatamente ante alarmas importantes.

6.4.1.3 Llevar un histórico

El sistema tiene que guardar un histórico de las alarmas que se producen

6.4.1.4 Consulta de histórico

El sistema tiene que permitir consultar la información del histórico

6.4.1.5 Consulta de información

El sistema tiene proveer de un mecanismo para permitir extraer información del mismo por parte de otro sistema o componente que lo requiera (Ej.: configuración o servicios)

6.4.1.6 Cada perfil de supervisión gestiona un conjunto de "Nodos de supervisión"

Cada perfil maneja un conjunto de nodos bajo los cuales se mantienen la lista de alarmas activas asociadas a cada uno de ellos. Cada perfil conoce la forma en que se vinculan las alarmas (en función del origen de la misma) y el nodo de supervisión asociado.

6.5 Descripción de Actores:

Equipo Remoto
Representa al sistema externo generador de alarmas

Administrador
Representa a los usuarios expertos que configuran el sistema, fundamentalmente las reglas que gobiernan el flujo de alarmas a través del sistema.

Gestor Equipos
Representa el componente del sistema que dialoga con el equipo remoto. Recibe las alarmas de estos y a su vez es la puerta de entrada para la configuración de los mismos. Pueden existir muchas instancias de este componente. En general están en correspondencia con equipos "Gestores de Elementos de Red" externos asociados a la supervisión de equipos específicos.

Gestión de Red
Representa el componente que tiene visión total de las alarmas recibidas a lo largo de todo el sistema. La misión final es la tarea de correlacionar alarmas de distintos orígenes, generar alarmas nuevas y enrutarlas a los interesados. En el presente caso de estudio, so se aborda la posibilidad de correlacionar alarmas por lo que el análisis de alarmas se hace en forma individual. Este componente es capaz de enrutar selectivamente a distintos interesados en función de algunos atributos internos del mensaje de alarma.

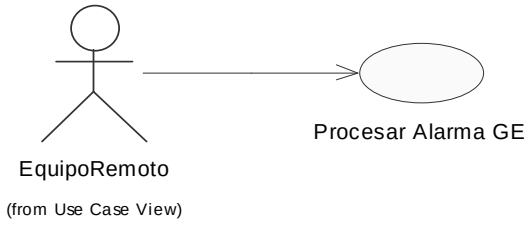
Supervisión Especializada
Representa el componente del sistema encargado de mantener la "visión" del estado de situación de los recursos gestionados por un Área de supervisión específica

Gestión Log Histórico
Componente del sistema encargado de registrar cada alarma. Brinda servicios de consulta para visualizar información histórica.

Cliente de supervisión Remota
Componente remoto del sistema que interactúa con la supervisión especializada. El cliente se conecta con el sistema de supervisión especializada para consultar alarmas de nodos específicos. Este es el componente de software que interactúa con el operador

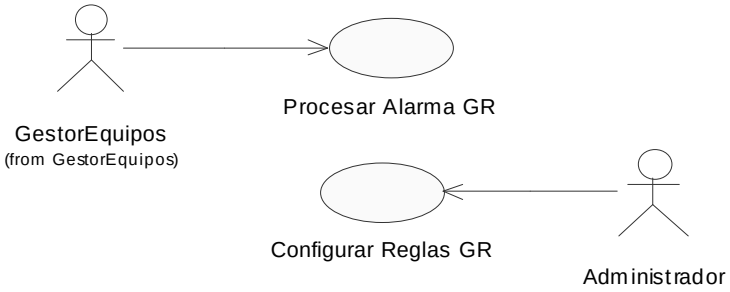
6.6 Casos de Uso

6.6.1 Subsistema "GestorEquipos"



Caso de Uso	***Procesar Alarma GE***
Actores	**Equipo Remoto**
Tipo	**Primario**
Descripción	Se reciben alarmas desde el "Equipo Remoto" y se realizan actividades como: - Filtrado Preliminar y Adaptación de Formato del Mensaje de Alarma - Almacenamiento Histórico - Enrutamiento hacia su procesamiento a nivel de gestión de Red

6.6.2 Subsistema "gestión Red"

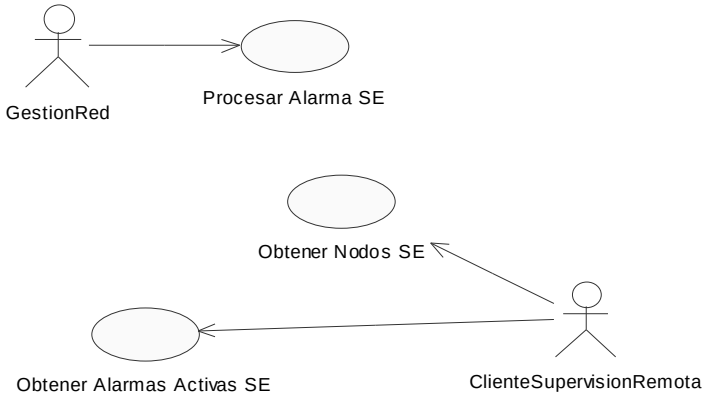


Caso de Uso	***Procesar Alarma GR***
Actores	**Gestor Equipos**
Tipo	Primario
Descripción	Se reciben alarmas desde el "Gestor Equipos" y eventualmente desde otras instancias de "gestión de Red" Cada alarma recibida es evaluada por un motor de reglas interno. Cada regla especifica: - Condición a verificar por el mensaje de alarma - Transformación a aplicar al mensaje de alarma (en el caso de verificarse la condición anterior) - Consumidor destinatario (en el caso de verificarse la regla el mensaje es enrutado al "Consumidor" indicado).

Caso de Uso	***Configurar Reglas GR***
Actores	**Administrador**
Tipo	***6.6.2.1.1.1 Secundario***

Descripción	Se configuran las reglas aplicadas en el Caso de uso “Procesar Alarma GR”

6.6.3 Subsistema "supervisión Especializada”

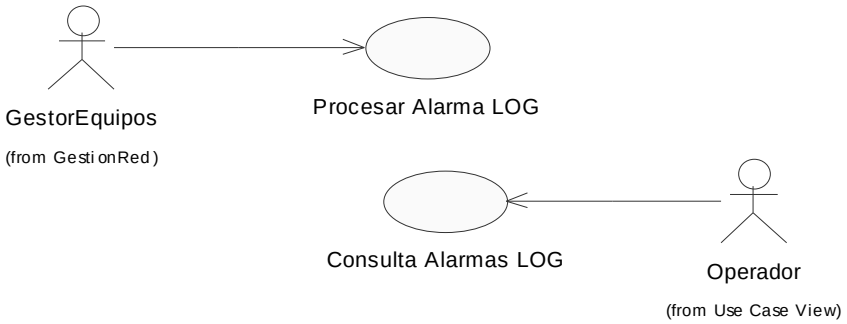


Caso de Uso	***Procesar Alarma SE***
Actores	**Gestión Red**
Tipo	Primario
Descripción	Se procesan las Alarmas recibidas desde la “gestión de Red”. La configuración realizada sobre la Gestión de Red determina la “selección” de un subconjunto de alarmas que llegan al modulo de supervisión Especializada (lo cual implícitamente define el perfil de supervisión). En este contexto el procesamiento de la Alarma implica mantener actualizada la visión de “Alarmas” que están activas. Cada alarma tiene un “estado” que indica si el fallo del equipo o servicio asociado se está activando o desactivando. Con esta información se mantiene la lista de alarmas activas. La lista se estructura en forma jerárquica. Existen nodos de supervisión que mantienen las “alarmas activas” vinculadas a dichos nodos.

Caso de Uso	***Obtener Nodos SE***
Actores	**Cliente supervisión Remota**
Tipo	***6.6.3.1.1.1 Primario***
Descripción	Brinda la lista de “nodos” gestionados a los efectos de que el usuario de esta operación pueda seleccionar un nodo específico.

Caso de Uso	***Obtener Alarmas Activas SE***
Actores	**Cliente supervisión Remota**
Tipo	***6.6.3.1.1.2 Primario***
Descripción	Se devuelve la lista de Alarmas “activas” para el nodo de supervisión solicitado.

6.6.4 Subsistema "gestión Log Histórico”



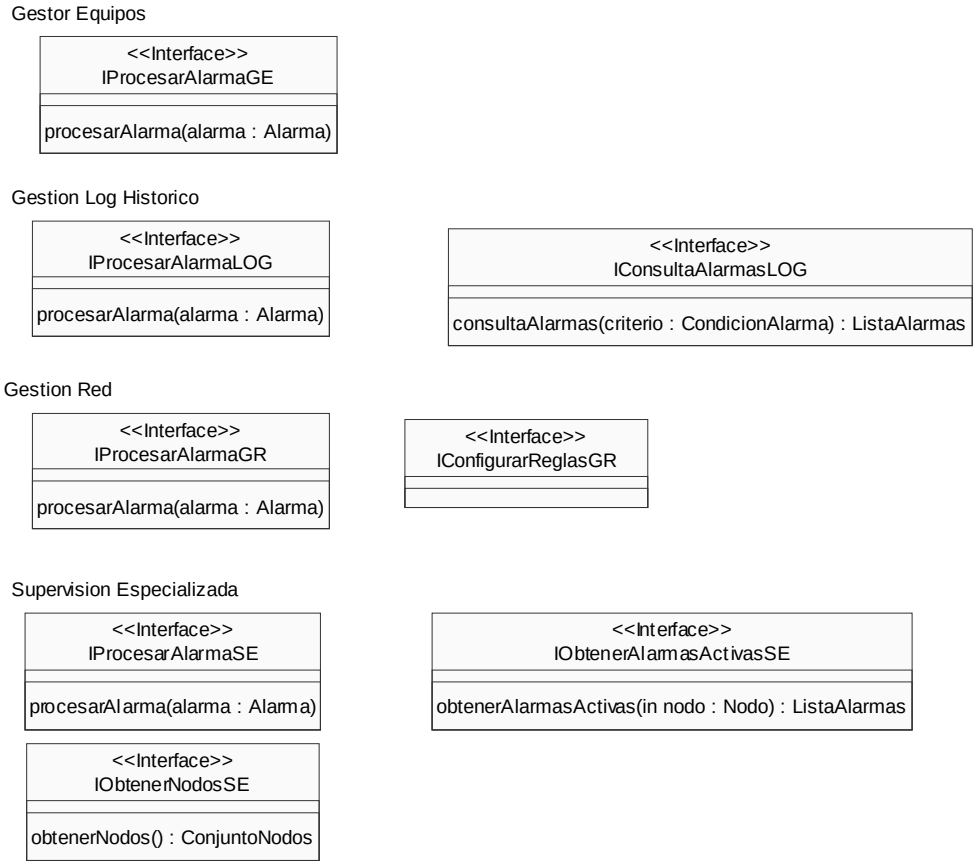
Caso de Uso	***Procesar Alarma LOG***
Actores	**Gestor Equipos**
Tipo	Primario
Descripción	Las alarmas recibidas se registran en un medio persistente de manera que queden disponibles para consultas históricas.

Caso de Uso	***Consulta Alarmas LOG***
Actores	**Operador**
Tipo	Secundario
Descripción	El Operador puede especificar algún criterio de selección de Alarmas Históricas de modo de recuperar la sucesión histórica de eventos en forma selectiva. Por ejemplo el operador podría especificar un rango de fecha/hora y algún tipo de alarma a los efectos de visualizar la ocurrencia de las mismas en dicho lapso de tiempo.

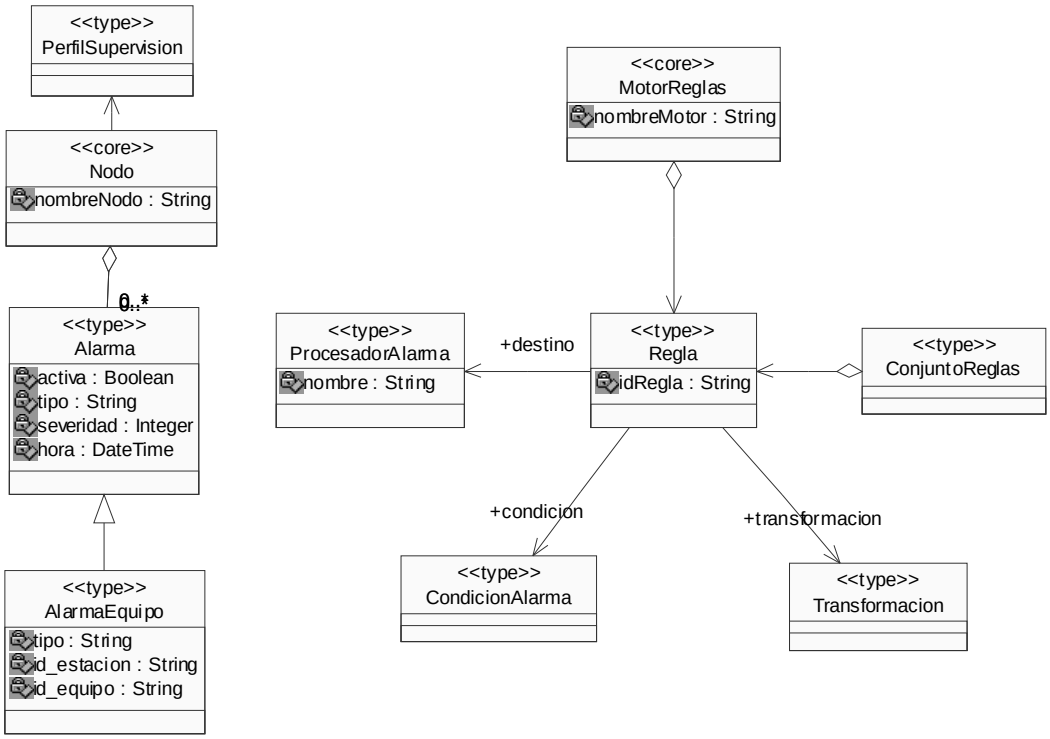
6.7 Identificacion de componentes

6.7.1 Operaciones del sistema

En primer lugar se establecen Interfaces para cada caso de uso (Operaciones del sistema)



6.7.2 Desarrollo del modelo de tipos del negocio

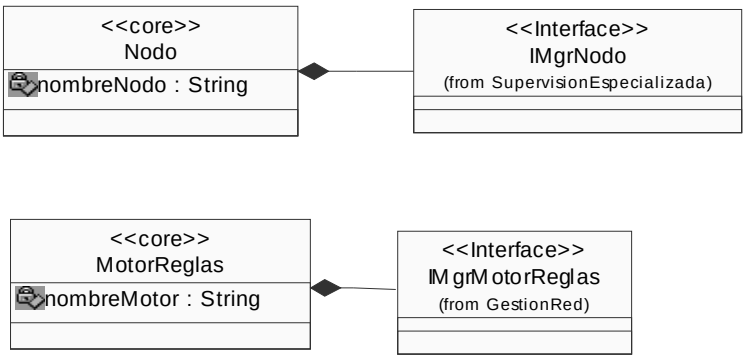


El acceso al modelo de información se llevará a cabo a través de entidades que consideramos como “tipos principales”, el resto de la infamación se canaliza utilizando tipos del presente modelo a través de las interfases de acceso a estas entidades.

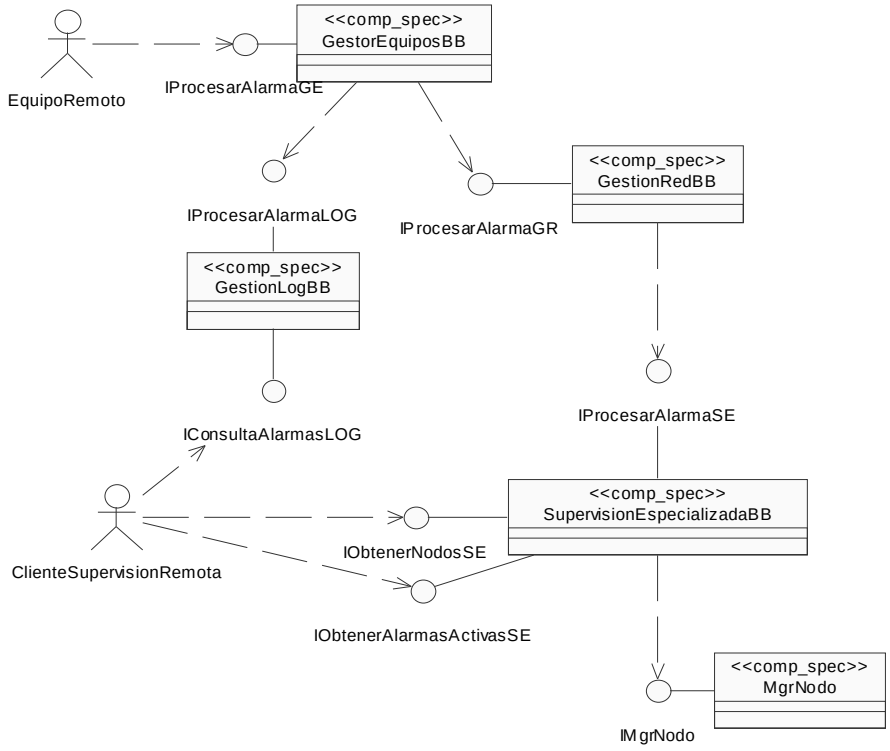
Las entidades `Nodo` y `MotorReglas` se manejarán como entidades principales.

6.7.3 Acceso al modelo de Información

El acceso a las entidades principales se canalizará por interfaces asociadas a cada uno de los tipos principales



6.7.4 Especificación de componentes y arquitectura inicial

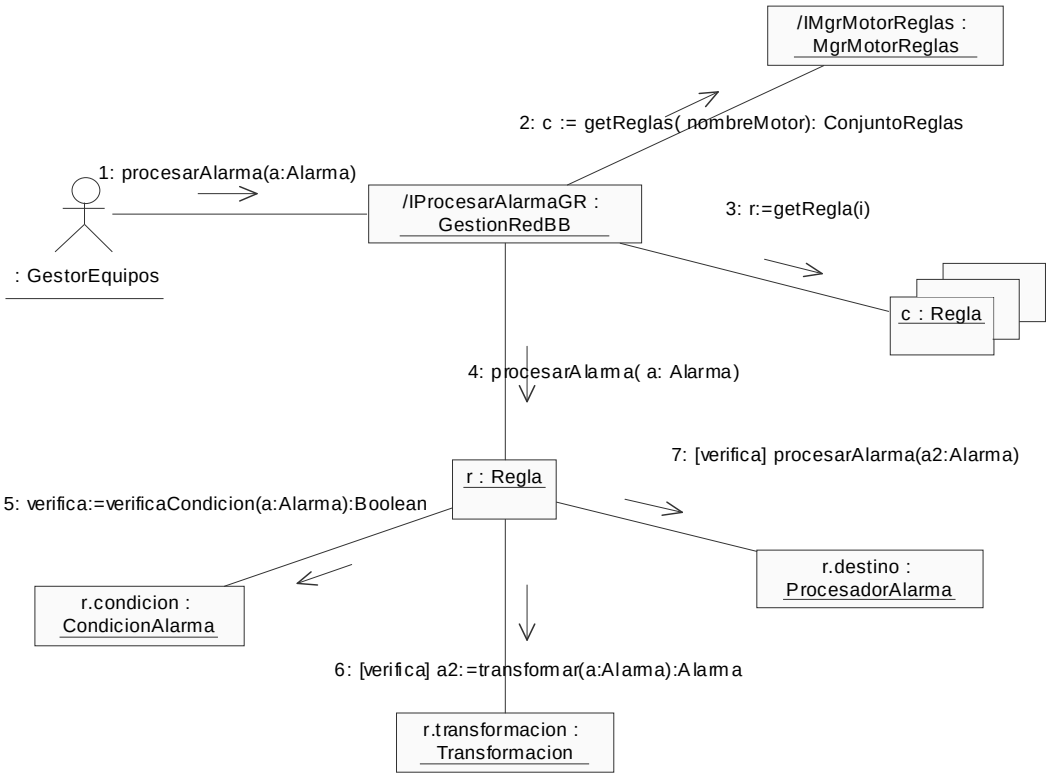


6.7.5 Interacción de componentes

Se realizan las interacciones para cada operación del sistema

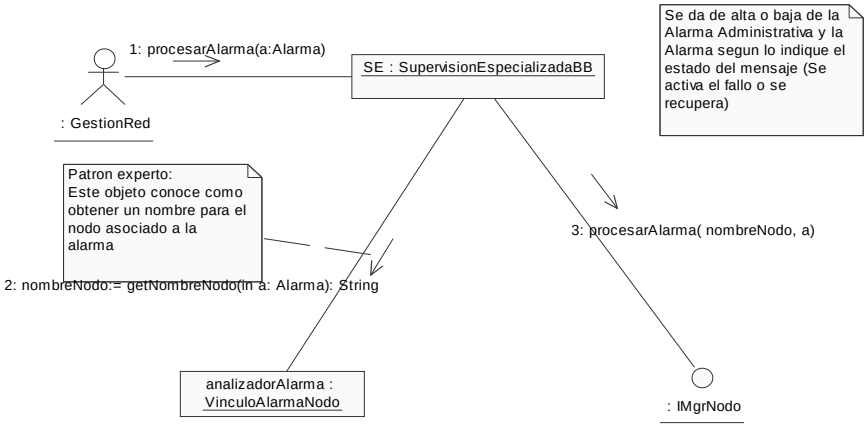
6.7.5.1 Gestión de Red

6.7.5.1.1 Procesar AlarmaGR

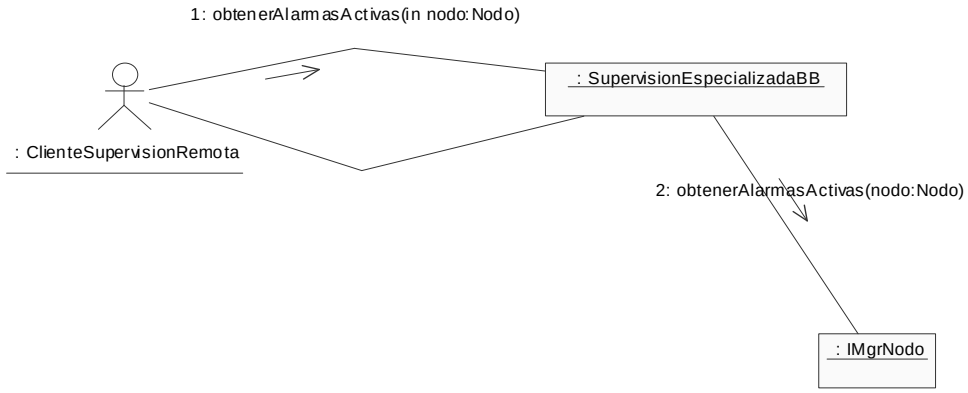


6.7.5.2 Supervisión especializada

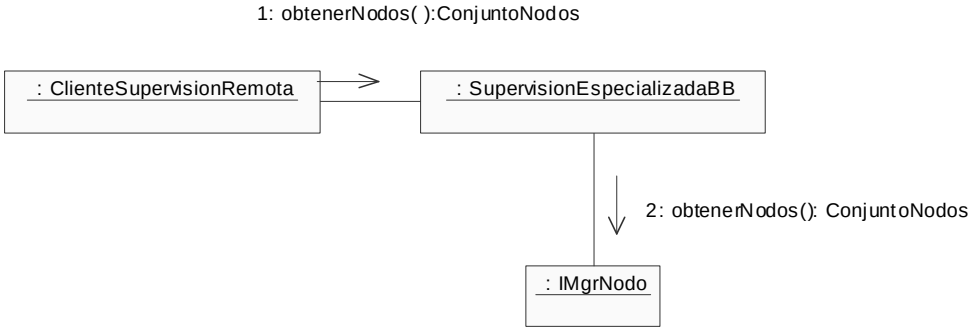
6.7.5.2.1 Procesar AlarmaSE



6.7.5.2.2 Obtener Alarmas Activas SE

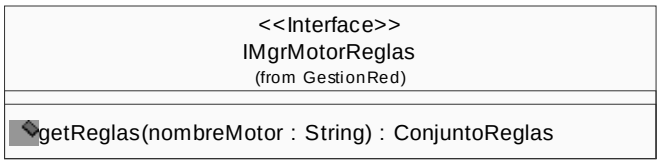


6.7.5.2.3 Obtención Nodos SE



6.7.6 Descubrimiento de operaciones del negocio

De las interacciones anteriores con los tipos principales surge el detalle de las operaciones para cada tipo del negocio



Bibliografía y Referencias

[1] Ayers D., *Professional Java Server Programming*, Wrox, 2000.

[2] Booch,G., *The Unified Modeling Language User Guide,* Addison Wesley, 1999.

[3] DMTF-Distributed Management Task Force. *Common Information Model (CIM) Specification.* Versión 2.2. Junio 1999.

[4] Erfani, Lawrence, Malek, Sugla; *Network Management: Emerging Trends and Challenges,* Bell Labs Tecnical Journal, 1999.

[5] Gallo, Jorge. Anotaciones Personales. 2002.

[6] Glitho, Hayes; *TMN: Vision vs Reality,* Communications Magazine March, Vol. 33 Nº3, 1995.

[7] Goers,Willam; Brenner,Michael; *Implementing a Management System Architecture Framework,* Bell Labs Technical Journal , Oct-Dec 2000.

[8] Hebrawi, B., *GDMO Object Modelling & Definition for Network Management* , Technology Appraisals, Twickenham, UK, 1995.

[9] IEC-International Engineneering Consortium. Vertel. *Tutorial: Telecommunications Management Networks* http://www.iec.org/online/tutorials/tmn (fecha de acceso: 01/03/2003).

[10] Inoue, Lapierre, Mossotto Editors, *The TINA Book. A cooperative solution for a competitive world*. Prentice Hall Europe 1999.

[11] ITU-T, *Principios para una red de gestión de las telecomunicaciones, Recomendación M3010,* 1996.

[12] ITU-T, *Red de gestión de las telecomunicaciones , Recomendación M3400,* 1997.

[13] Larman,C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process* (2nd Edition).Prentice Hall 2001.

[14] Lewis,David; Wade, Vincent. *A Logical Architecture for an Open Development Framework for Component-based Management Systems.* White Paper. FORMS Consortium. Octubre 2001.

[15] Lewis,David; Wade, Vincent; Cullen, Brian. *Towards the Technology Neutral Modelling of Management Components.* Journal of Network and Systems Management archive.Volume 11 , Issue 1 ISSN:1064-7570 .March 2003.

[16] Marinescu, Floyd. *EJB Design Patterns.* John Wiley & Sons, Inc. ISBN: 0-471-20831-0. 2002.

[17] OMG. *Notification Service Specification.*-Version 1.0.1 (especificación formal/02-08-04) Agosto 2002.

[18] OMG. *Overview and Guide to OMG's architecture,MDA Guide* Versión 1.0. Object Management Group 2001.

[19] Pattison T., *Programming Distributed Applications with Microsoft COM+ and Visual Basic 6.0,* Second Edition, Microsoft, 2000.

[20] Pauthner, Power; *Telecommunications network management,* IEEE series, Cap.3, pag. 122, 1998.

[21] Perovich, D; Rodríguez Viacava,L.;Vignaga, A. . *Arquitectura de sistemas de información basados en componentes sobre la plataforma J2EE.* Reporte Técnico 03-16. (ISSN: 0797-6410) Biblioteca de PEDECIBA. Facultad de Ingeniería. Universidad de la República (Uruguay) 2003.

[22] Poole, John. *Model Driven Architecture: Vision, Standards and EmergingTechnologies* Hyperion Solutions Corporation, Remitido a ECOOP2001. Abril 2001.
[23] Pras, Aiko. – *Network management architectures*. Ph.D. thesis. Centre for Telemathics and Information Technology . Netherlands. ISBN 90-365-0728-6. 1995.
[24] Pras, Aiko; van Beijnum, B.J.; Sprenkels,R. *Introduction to TMN* . CTIT Technical Report (ISSN 1381-3625), 99-9, April 1999 Univerity of Twente. Netherlands. 1999.
[25] Pritchard, Jason. COM and CORBA Side by Side.Architectures,Strategies, and Implementations,Adison-Wesley ISBN 0201379457. Feb. 2000.
[26] Raman,L., *Fundamentals of Telecommunications Network Management*, IEEE Press, Cap. 1.3, 1999.
[27] Ramémont, Bosse, Henry, Sexton, Schiavoni; *Gestión de los elementos de red SDH: una aplicación del modelado de información*, Revista Comunicaciones Eléctricas Alcatel, 1993.
[28] RHK, *Fanning the NGOSS Flame at TMW* , News Alert. Noviembre 2002
[29] Ritter S., *Enterprise Java Beans: The answers to the three questions every developer will ask*, Sun Microsystems (www.sun.com), 1999.
[30] Roman, Ed;Ambler, Scott;Jewel, Tyler. *Mastering Enterprise JavaBeans™ Second Edition.* John Wiley & Sons, Inc. ISBN: 0-471-41711-4. 2002
[31] Rozenblit,Moshe; *Security for Telecommunications Network Management*, IEEE Press, pag. 8, 2000.
[32] Rumbaugh, J. *The Unified Modeling Langage Reference Manual*, Addison Wesley, 1999.
[33] Sexton, Reid; *Broadband Networking ATM, SDH and SONET*, Artech House, Cap7,pag. 223, 1997.
[34] Sigel. *OMG's Model Driven Architecture*. Publicación electrónica EURESCOM mess@ge 2/2002. 2002.
[35] Sun Microsystems. *OSS through Java J2EE.Design Guidelines. OSS-J* Architecture Board. Version 1.1.October 31 2001.Sun Microsystems (http://java.sun.com/products/oss accedido en enero/2003).
[36] TINA Tutorial, Report of ACTS SIA Chain 2º Meeting CNET, Paris, 24 Sept 1996.
[37] TMForum *GB914 System Integration Map.*Versión 2.5. 2002
[38] TMForum *GB921 Enhanced Telecom Operations (eTOM)The Business Process Framework.* Versión 3.0, 2002.
[39] TMForum *GB922 Shared Information/Data (SID) Model-Phase I: Concepts, Principles, and Business Entities - GB922* Versión 1.5. 2002.
[40] TMForum , *NGOSS Architecture Technology Neutral Specification - TMF053 v3.0. Abril 2003.*
[41] TMForum, *GB909 Part 1-Generic Requirements for Telecommunications Management Building Blocks: Part I of the Technology Integration Map - GB909 v3.0*, 2001.
[42] TMForum, *GB909 Part 2 - Smart Telecommunications Management Network Technology Integration Map Technology Direction Statement - GB909 v1.1.* Octubre 1998.
[43] TMForum, *Millennium Strategic Plan, GB912* release 1999-1, Febrero 1999.
[44] TMForum, *Telecom Operations Map, GB910*, Versión 2.1, 2000.

[45] TMForum, *TOM Application Note, GB910A*, Versión 1.1, 2000.
[46] TMForum, *TOM Application Note, GB910B*, Versión 1.1, 2000.
[47] Vawter C. y E. Roman, *J2EE vs. Microsoft.NET: A comparison of building XML-based web services*, 2001.
[48] Vignaga,A; Perovich, D. *Enfoque metodológico para el desarrollo basado en componentes.* Reporte Técnico 03-14. (ISSN: 0797-6410) Biblioteca de PEDECIBA. Facultad de Ingenieria. Universidad de la República (Uruguay) 2003.
[49] Zahavi R., *Enterprice Application Integration with CORBA*, OMG Press, 2000.