(1·OSt 18) XX

CINVESTAV
Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

ANALISIS, DISEÑO E IMPLEMENTACIÓN DE UN HABILITADOR CTI BASADO EN TAPI

TESIS QUE PRESENTA
ROSA GABRIELA PADILLA GUTIÉRREZ

PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE
INGENIERÍA ELÉCTRICA

Guadalajara, Jal., Junio de 2000.

Bibliográficos

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN HABILITADOR CTI BASADO EN TAPI

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

por:

Rosa Gabriela Padilla Gutiérrez

Licenciado en Informática
Universidad de Guadalajara, 1993-1997

Becaria del CONACYT, expediente no. 121145

Director de Tesis:

Dr. Deni Librado Torres Román

CINVESTAV del IPN Unidad Guadalajara, Junio de 2000.

Agradecimientos

A Dios por estar siempre a mi lado para alumbrar mi camino.

A mis padres por su paciencia y cariño, y por darme la oportunidad de continuar con mis estudios.

A mi asesor Dr. Deni Torres Román por su trabajo y tiempo invertidos en el desarrollo de este trabajo.

A mi prometido Luis Prabú Ximénez Cárdenas, quien dedicó el tiempo del mundo para resolver mis dudas y para motivarme a seguir adelante.

Al Dr. Félix Ramos Corchado por su tiempo, dedicación y acertados consejos, los cuales me permitieron llevar a buen término esta tesis.

A CONACYT por su apoyo económico, el cual me permitió realizar esta maestría.

Rosa Gabriela Padilla Gutiérrez.

Indice General

Indice de Figuras

Lista de Acrónimos

ACD	Automatic Call Distributor
API	Application Programming Interface
ATSP	Proveedor de Servicios del Asistente Telefónico
PBX	Private Branch Exchange
CALT	Control Administrativo de Llamadas Telefónicas
CRC	Class Responsibility Collaborator
COM	Component Object Model
CTI	Computer Telephone Integration
DID	Direct Inward Dialing
DLL	Dynamic Link Library
FST	Formal Specification Technique
HTT	Habilitador Telefónico basado en TAPI
ISDN	Integrated Services Digital Network
MSP	Media Service Provider
MSPI	Media Service Provider Interface
NDIS	Network Driver Interface Specification
OMT	Object Modeling Technique
OOT	Object Oriented Technology
POTS	Plain Old Telephone Services
SPI	Service Provider Interface
TAPI	Telephony Application Programming Interface
TPLH	Teleprocessing Line Handling
TSAPI	Telephony Services Application Programming Interface
TSP	Telephony Service Provider
TSPI	Telephony Service Provider Interface
VATD	Controlador Virtual del Asistente Telefónico
WOSA	Windows Open Services Architecture

1 Introducción

1.1 Antecedentes

Los orígenes de CTI (Computer Telephony Integration) se remontan a finales de los sesentas, cuando IBM desarrolló el TPLH, siglas de Teleprocessing Line Handling. Este dispositivo actuaba de enlace especial entre una computadora (mainframe) y un conmutador (PBX[1]). El enlace permitía a la computadora enviar comandos al conmutador y al conmutador reportar eventos relacionados con el estado de las llamadas telefónicas a la computadora. El TPLH no tuvo gran impacto en aquel entonces, dado que su costo era alto y pocas aplicaciones justificaban la inversión.

Posteriormente surgió un concepto que se convertiría en uno de los desarrollos más ampliamente utilizados de CTI: el Centro de Llamadas[EDG97] (Call Center). El objetivo básico del Centro de Llamadas es la realización de negocios por teléfono de manera eficiente y personalizada. El Centro de Llamadas ha evolucionado para adaptarse a distintos medios de comunicación: voz tradicional, voz sobre IP, correo electrónico, chat, entre otros. A este nuevo concepto se le conoce como Centro de Contactos[MAY99] (Contact Center).

Actualmente, el alcance de CTI se ha extendido a todos los tipos de integración de computadoras y sistemas telefónicos y a todas las formas de integración de voz y datos. Un área de trabajo relevante en este campo es la que se refiere a la integración de servicios de telefonía basados en IP con los servicios proporcionados por la red tradicional de circuitos conmutados[RIZ99] a través del concepto de Red Inteligente (IN-Intelligent Network)[ITUQ.1200].

1.2 Habilitadores CTI

Un elemento clave para el desarrollo de soluciones de software CTI es el aislamiento de las aplicaciones de usuario de los detalles específicos de los sistemas telefónicos y de la red subyacente. La razón es que un software CTI puede trabajar ya sea sobre la red de telefonía pública o sobre la Internet, por ejemplo. Además puede hacerlo a través de un modem, de un PBX o de un hardware propietario, por mencionar algunos dispositivos o sistemas telefónicos.

A principios de los 90's dos empresas líderes en software, Microsoft y Novell, incursionaron en el campo CTI haciendo énfasis en el desarrollo de software (APIs[2]) que aislara al usuario y al desarrollador de los problemas de compatibilidad con el hardware. De

---

[1] PBX: Private Branch Exchange.

[2] API : Application Programming Interface.

ahí surgieron TAPI (Telephony API)[MIC97A] y TSAPI (Telephony Services API)[NOV99], respectivamente. Una API es un conjunto de llamadas a funciones de bibliotecas de alto nivel que han sido construidas a partir de llamadas a funciones de sistemas de más bajo nivel para ocultar complejidad[MCL98]. Tanto TAPI como TSAPI pertenecen a un tipo de software conocido como habilitadores CTI. Un *habilitador CTI* es un módulo de software que proporciona a otras aplicaciones acceso a servicios de telefonía.

Walters [WAL97] define cuatro tipos de habilitadores CTI:

1. *API básica*

   Es una interfaz de software que ofrece las funciones de telefonía básica (hacer una llamada, transferir una llamada o realizar conferencias) a las aplicaciones. La interfaz con la aplicación es usualmente una biblioteca de funciones.

2. *API de alto nivel*

   Es una capa de software que se coloca entre la API básica y el software de aplicación. Su función es reducir la API básica a unos pocos comandos clave, motivo por el cual se le conoce como capa "embudo"[3]

3. *"Middleware"*

   El "middleware" está conformado por una API de alto nivel más una capa denominada capa de enlace. La capa de enlace proporciona el medio para ligar las funciones telefónicas a un programa de aplicación existente sin modificarlo.

4. *Servidor de Telefonía*

   El Servidor de Telefonía incluye, además de las características de los habilitadores anteriores, la capacidad de actuar como interfaz con software que no es CTI y proporciona funciones de sistemas interactivos de voz (IVR[4]) y de acceso a bases de datos.

1.3 Motivación

Uno de los habilitadores CTI más ampliamente utilizado es TAPI[5] TAPI está constituida por un grupo de funciones que aíslan a las aplicaciones de la comunicación con los protocolos específicos de los dispositivos y de la red telefónica. No obstante, TAPI posee ciertos inconvenientes:

- No provee el control necesario para permitir la concurrencia de llamadas telefónicas
- No maneja el asincronismo de algunas de sus funciones
- La realización de operaciones telefónicas sencillas, tal como hacer una llamada, requiere de la ejecución de varias funciones

---

[3] Funnel layer.
[4] IVR: Interactive Voice Response.
[5] TAPI se describe en el Capítulo 3.

- El enfoque orientado a funciones dificulta ˙ ˙diseño de las aplicaciones que hacen uso de esta interfaz

El manejo de estos aspectos vuelve a las aplicaciones basadas en TAPI innecesariamente complejas. Por lo tanto, resulta evidente la necesidad de una API de alto nivel que encapsule la funcionalidad de TAPI y proporcione a las aplicaciones una interfaz de telefonía más sencilla.

1.4 Aplicación de una API de alto nivel basada en TAPI: Asistente Telefónico

El proyecto Asistente Telefónico (Communication Helper) es una solución CTI para la administración de los recursos telefónicos en ambiente Windows®[6]. El Asistente Telefónico provee al usuario con las siguientes capacidades, todas ellas administradas a través de perfiles de usuario:

- Control de llamadas entrantes y salientes en varias líneas
- Máquina contestadora
- Correo de voz
- Envío automático de mensajes y programación de llamadas
- Directorio personal
- Teléfono en pantalla
- Tarificación

El Asistente Telefónico está conformado tanto por software como por hardware. Esto es, consta básicamente de una aplicación de usuario desde la cual se accede a los servicios descritos anteriormente y de un hardware propietario que provee el soporte físico necesario. Además la aplicación de usuario debe ser capaz de interactuar con hardware distinto de la tarjeta del Asistente Telefónico (por ejemplo, con un modem). Para lograr esta portabilidad la aplicación de usuario debe acceder a los servicios telefónicos a través de alguna interfaz de telefonía, como TAPI. Por lo tanto, una API de alto nivel basada en TAPI es útil en este contexto.

1.5 Objetivos

Los objetivos de la tesis son:

- El estudio de las APIs de telefonía sobre Windows para aplicaciones CTI

---

[6] Microsoft Windows® 95 y Windows® 98.

- La especificación, análisis y diseño de un habilitador CTI basado en TAPI que proporcione a las aplicaciones basadas en Windows® una interfaz para el acceso a los servicios telefónicos
- La implementación del habilitador CTI basado en TAPI para la aplicación denominada Asistente Telefónico

1.6 Organización de la tesis

En el Capítulo 2 se presenta una visión general de los servicios de telefonía sobre Windows® En el Capítulo 3 se introducen los conceptos principales del enfoque orientado a funciones de TAPI. En el Capítulo 4 se presenta la especificación de requerimientos del habilitador CTI a desarrollar. En el Capítulo 5 se resume el proceso de desarrollo en Fusion del habilitador. Finalmente, el Capítulo 6 presenta las conclusiones y el trabajo futuro.

2 Telefonía sobre Windows®

2.1 Introducción

En el capítulo anterior se menciona que uno de los habilitadores CTI más ampliamente utilizados ha sido TAPI. TAPI es sólo una parte de un grupo de elementos que trabajan en conjunto para proporcionar una interfaz "estándar" de telefonía sobre Windows® El objetivo de este capítulo es presentar, de manera general, el esquema y la evolución de la interfaz de telefonía sobre Windows® desarrollada por Microsoft®

2.2 El Modelo WOSA

Los servicios de telefonía sobre Windows® se basan en el Modelo WOSA[MIR96] (Windows Open Services Architecture). Este modelo proporciona un marco de trabajo en el cual las aplicaciones pueden acceder de manera transparente a los recursos y servicios en ambientes locales o distribuidos. Para ello, WOSA define una arquitectura genérica compuesta por tres capas de software:

- *Una Interfaz para la Programación de Aplicaciones o API* (Application Programming Interface), mediante la cual las aplicaciones pueden acceder a un determinado hardware o servicio
- *Una Interfaz API/SPI*, constituida por una o más Librerías de Enlace Dinámico o DLLs que se encargan de enlazar de manera dinámica las aplicaciones con los servicios o controladores de hardware
- *Una Interfaz para Proveedores de Servicios o SPI* (Service Provider Interface), a través de la cual se solicitan los servicios a los controladores específicos de un dispositivo o hardware

Estas capas evitan a los desarrolladores de aplicaciones ocuparse de la comunicación directa con los diversos servicios que requieran y de los detalles subyacentes de la red. Además, los fabricantes de hardware disponen de un medio para que sus productos puedan ser utilizados por todas las aplicaciones que cumplan con la especificación de la API correspondiente, según sea el caso.

2.3 Los servicios de telefonía sobre Windows®

El acceso a los servicios de telefonía sobre Windows® es proporcionado, de acuerdo con el modelo WOSA, por los siguientes elementos:

- *Una interfaz para la programación de aplicaciones de telefonía denominada TAPI* (Telephony Application Programming Interface), mediante la cual los programadores de aplicaciones acceden a los servicios de telefonía
- *Una interfaz para proveedores de servicios de telefonía conocida como TSPI* (Telephony Service Provider Interface) [MIC97B], la cual implementan los fabricantes de hardware de telefonía y una interfaz para proveedores de servicios de medios llamada MSPI (Media Service Provider Interface)[MIC99A]. MSPI sólo está disponible en la versión más actual (TAPI 3.0) de la telefonía sobre Windows®
- *Una o más librerías de enlace dinámico de telefonía* (DLLs[7] de Telefonía), las cuales forman parte del sistema operativo

La Figura 2-1 muestra cómo interactúan estos elementos con las aplicaciones y los proveedores de servicios de telefonía y de medios en el marco de WOSA. Las aplicaciones solicitan el acceso a los recursos y/o servicios de telefonía mediante invocaciones a las funciones definidas por TAPI. Las invocaciones a funciones de TAPI son transformadas en invocaciones a funciones de TSPI o MSPI por las DLLs de telefonía de Windows®. Los proveedores de servicios de telefonía y de medios son los encargados de implementar las funciones de TSPI y MSPI, respectivamente.

Las flechas en ambos sentidos indican que la comunicación entre las distintas partes es bidireccional. La comunicación hacia abajo consiste de las solicitudes de servicios de telefonía. La comunicación hacia arriba constituye las respuestas a las solicitudes realizadas y notificaciones de eventos ocurridos en los dispositivos físicos.

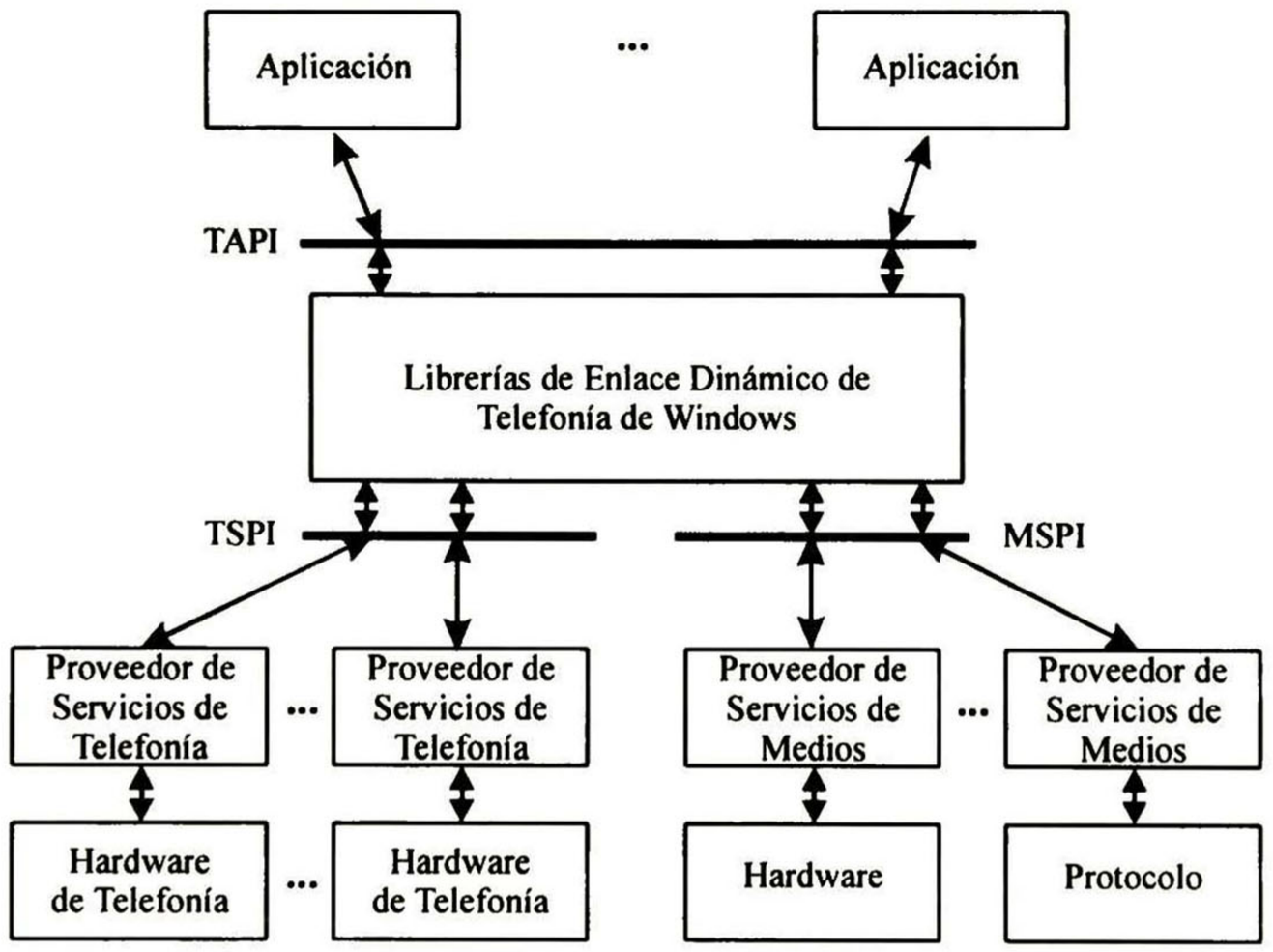


Figura 2-1. Arquitectura básica de los servicios de telefonía sobre Windows® de acuerdo con el modelo WOSA

[7] DLLs: Dynamic Link Libraries.

2.3.1 TAPI

TAPI es una biblioteca de funciones mediante la cual se lleva a cabo la comunicación entre las aplicaciones de telefonía y el sistema operativo Windows®. TAPI permite el desarrollo de software independiente de dispositivo que proporcione servicios de telefonía tales como:

- Consulta de las capacidades de los dispositivos TAPI (de línea y telefónicos[8])
- Marcación de números telefónicos desde la computadora
- Llamada en espera
- Creación y manejo de conferencias entre llamadas telefónicas
- Detección de cambios en el estado de las llamadas telefónicas
- Manejo de llamadas telefónicas entrantes basado en el servicio de callerID

Esta independencia del hardware se logra porque la aplicación siempre accede a los diferentes servicios a través de funciones TAPI. Las DLLs de TAPI se encargan de traducir las peticiones de las aplicaciones a los protocolos e interfaces de comunicación requeridos.

2.3.2 TSPI

TSPI es la interfaz para la programación de proveedores de servicios de telefonía o TSPs (Telephony Service Providers). Los TSPs son librerías de enlace dinámico que responden a las solicitudes de las DLLs de telefonía de Windows® y realizan las tareas de bajo nivel necesarias para comunicarse con los dispositivos de telefonía. Algunos ejemplos de estas tareas son las que controlan los estados de las llamadas y la marcación, generación y detección de señales.

Por lo tanto, los TSPs son los elementos responsables de convertir el modelo de llamada independiente de protocolo de TAPI en el mecanismo de control de llamadas con protocolo específico. Por ejemplo, el Unimodem[MIR95] puede traducir entre funciones TAPI y comandos estándar de un modem AT. El conocimiento de los comandos del modem reside en el TSP y no es requerido por los desarrolladores de aplicaciones basadas en TAPI.

Los TSPs no se comunican de manera directa con los dispositivos físicos, sino a través de controladores[9] Un controlador de dispositivo es un elemento de software que actúa como interfaz entre un sistema operativo y un dispositivo. En Windows® 95 este controlador puede ser implementado como una DLL o como un controlador virtual denominado VxD [10] [FIG99]. Los TSPs suelen considerarse como parte de los controladores de dispositivos.

---

[8] Descritos en el Capítulo 3.
[9] Drivers.
[10] VxD: Virtual Device Driver.

2.3.3 MSPI

La versión más actual del conjunto de interfaces de telefonía sobre Windows® incorpora una Interfaz para Proveedores de Servicios de Medios o MSPI. Esta interfaz permite a las aplicaciones controlar el transporte de información durante una llamada. Un proveedor de servicios de medios o MSP (Media Service Provider) proporciona a una aplicación la capacidad de controlar el medio para un mecanismo de transporte particular. Normalmente existirá un par MSP-TSP que proporcione a las aplicaciones tanto control de llamadas como control de medios.

Al igual que un TSP representa la capa de abstracción para el control de llamadas, el MSP controla el medio sin necesidad de que la aplicación TAPI contenga codificación específica a un dispositivo. Por ejemplo, Microsoft Windows® 2000 incluye un MSP que puede manejar el protocolo de telefonía sobre IP H.323[ITUH.323] y otro que puede manejar conferencias multicast sobre una red. Además, pueden desarrollarse proveedores de medios para otros protocolos o dispositivos físicos particulares.

2.4 Evolución de la telefonía sobre Windows®

Actualmente, TAPI proporciona la posibilidad de desarrollar aplicaciones seleccionando alguno de los siguientes enfoques:

- Programación orientada a funciones (en lenguaje C), soportada por la versión 2.1
- Programación orientada a componentes (utilizando lenguajes como C++, Java y Visual Basic), soportada por la versión 3.0

El enfoque actual orientado a funciones tiene tras de sí una serie de versiones mediante las cuales se ha ido refinando y ampliando el concepto de telefonía sobre Windows® En las siguientes secciones se describen cada una de las fases por las que ha atravesado el desarrollo de la telefonía sobre Windows®

2.4.1 TAPI 1.3

La versión 1.3 de TAPI permite el desarrollo de aplicaciones de telefonía de 16 bits bajo el ambiente Windows® 3.1x. En esta versión el núcleo del servicio de telefonía radica en un archivo llamado TAPI.DLL así como en un proceso oculto denominado TAPIEXE.EXE (ver Figura 2-2). TAPI.DLL se encarga de validar y dar un formato apropiado a los parámetros de la función TAPI invocada y enviar la solicitud al proceso de servicio de telefonía TAPIEXE.EXE.

Los archivos que implementan TAPI 1.3 se incluyen en el Kit para Desarrolladores de Software TAPI 1.0 (TAPI 1.0 Software Development Kit) y sólo soportan aplicaciones y proveedores de servicios TAPI 1.3 de 16 bits.

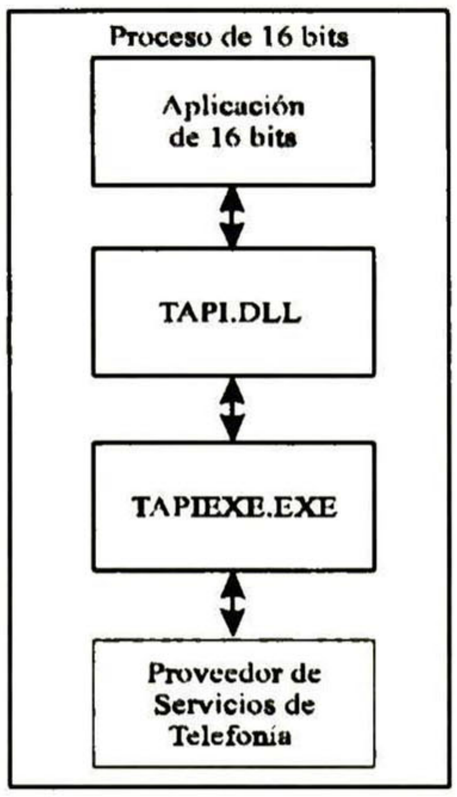


Figura 2-2. Arquitectura de TAPI 1.3

2.4.2 TAPI 1.4

Al igual que TAPI 1.3, la versión 1.4 es una interfaz de 16 bits que se basa en TAPI.DLL y TAPIEXE.EXE para llevar a cabo el procesamiento de las invocaciones a funciones TAPI. Esta versión viene incluida con el sistema operativo Windows® 95 e incorpora nuevas funciones, estructuras, mensajes y constantes a las interfaces TAPI y TSPI. Además permite el uso de TAPI en aplicaciones de 32 bits, eliminando con esto las limitaciones propias de las aplicaciones de 16 bits.

La Figura 2-3 muestra que las aplicaciones de 16 bits se enlazan directamente con TAPI.DLL. Las aplicaciones de 32 bits lo hacen a través de TAPI32.DLL. TAPI32.DLL transforma las funciones TAPI de 32 bits a funciones TAPI de 16 bits, las cuales se enlazan con TAPI.DLL. TAPI 1.4 soporta aplicaciones y proveedores de servicios de TAPI 1.3.

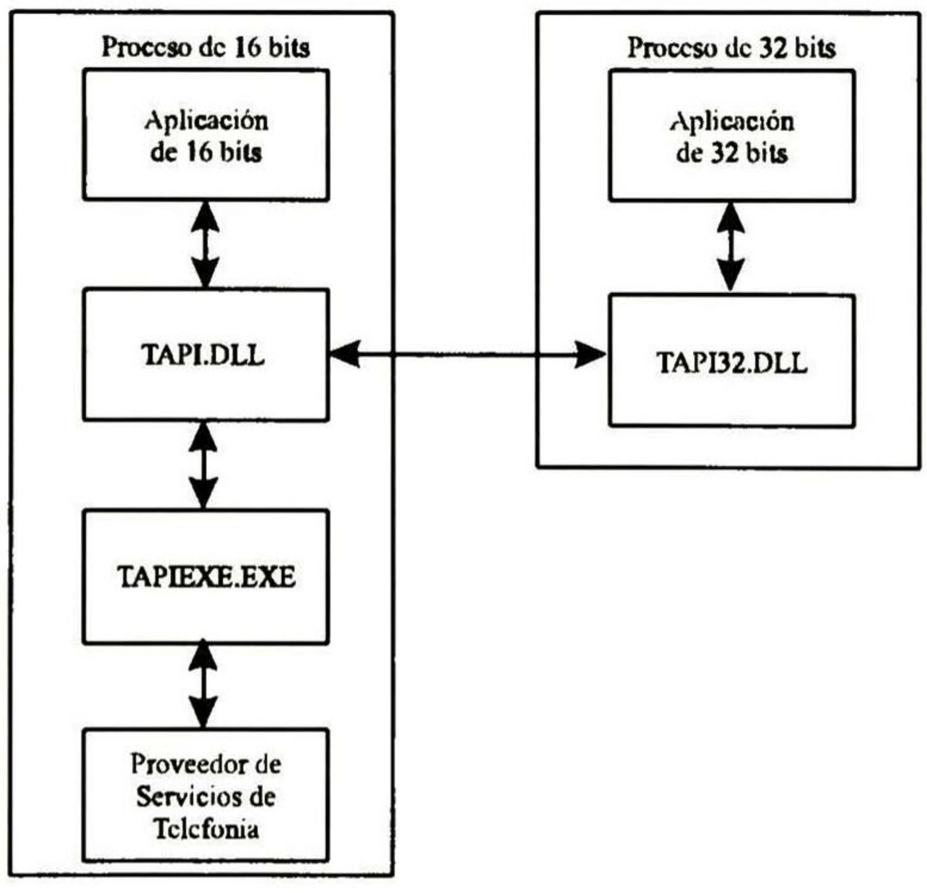


Figura 2-3. Arquitectura de TAPI 1.4

2.4.3 TAPI 1.5

La versión TAPI 1.5 fue diseñada específicamente para la versión 1.0 de Windows® CE y constituye un subconjunto de las funciones proporcionadas por TAPI 1.4.

2.4.4 TAPI 2.0

TAPI 2.0 aporta pocas mejoras con respecto a la funcionalidad básica proporcionada por la versión 1.4, pero presenta cambios arquitecturales con el propósito de llevar TAPI a la plataforma Windows® NT[NIX96]. En Windows® NT el núcleo de TAPI está constituido por un nuevo proceso de servicio llamado TAPISRV.EXE.

TAPI 2.0 es una interfaz de 32 bits (ver Figura 2-4). Por lo tanto, sucede lo contrario que en TAPI 1.4. Las aplicaciones de 16 bits se enlazan con TAPI32.DLL mediante TAPI.DLL. Las aplicaciones de 32 bits se comunican directamente con TAPI32.DLL. TAPI32.DLL se encarga de validar y dar un formato apropiado a los parámetros de la función y enviar la solicitud al proceso de servicio de telefonía TAPISRV.EXE. Otras características de esta versión son las siguientes:

- *Portabilidad para aplicaciones de 32 bits*

  Las aplicaciones de 32 bits para Windows® 95 (basadas en TAPI 1.4) pueden ser ejecutadas en Windows® NT sin modificación ni recompilación.

- *Portabilidad para aplicaciones de 16 bits*

  Las aplicaciones de 16 bits para Windows® 95 y Windows® 3.1 (que utilicen TAPI 1.3) también pueden ejecutarse bajo Windows® NT sin modificación ni recompilación.

- *Soporte a Unicode*

  Las aplicaciones de 32 bits pueden hacer uso de las funciones TAPI ANSI o de las funciones Unicode[UNI00] correspondientes que pasan o retornan cadenas[11] Las funciones TAPI versión Unicode son aquéllas que llevan el sufijo W, por ejemplo *lineMakeCallW*.

- *Compatibilidad con TAPI NDIS*[12]

  Se preserva el soporte existente en Windows® NT 3.5 para minipuertos ISDN[13] WAN bajo el Servicio de Acceso Remoto. Los controladores de minipuertos WAN NDIS son soportados bajo un proveedor de servicios en modo kernel sin modificación.

- *Soporte para el Registro*

---

[11] Strings.
[12] NDIS: Network Driver Interface Specification
[13] ISDN: Integrated Services Digital Network.

Todos los parámetros de telefonía son almacenados en el registro de Windows®. Los parámetros de telefonía representan información relacionada con: las propiedades de marcación desde la computadora, los TSPs y los códigos de países.

- *Soporte para Centro de Llamadas*

  TAPI provee soporte para implementar Centros de Llamadas[14]. El soporte incluye el modelado de colas y puertos de marcación predictiva (se marcan números telefónicos de una lista y cuando alguno es contestado por una persona, la llamada se enruta a un agente libre), control de agentes ACD[TEC99A] (Automatic Call Distributor) y control de estado de la estación.

- *Soporte de Calidad de Servicio*

  Las aplicaciones pueden solicitar, negociar y renegociar parámetros de calidad de servicio o QoS (Quality of Service) con la red. También pueden recibir indicación de QoS en llamadas entrantes y de modificación de QoS por la red. Las estructuras de QoS son compatibles con aquéllas usadas en la especificación 2.0 de sockets en Windows®

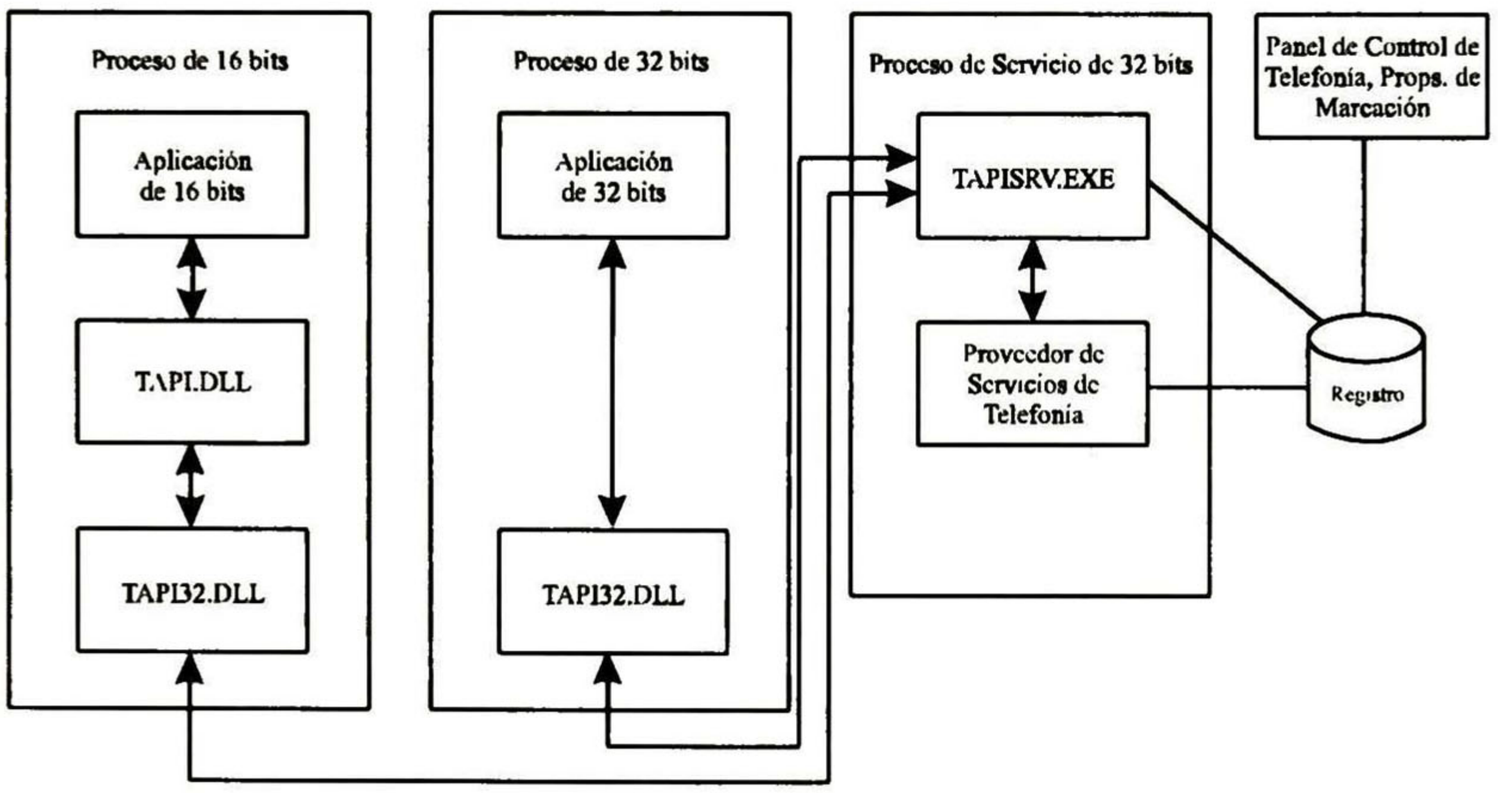


Figura 2-4. Arquitectura de TAPI 2.0

2.4.5 TAPI 2.1

Las versiones anteriores a TAPI 2.1 proporcionan a las aplicaciones una interfaz para comunicarse con el hardware de telefonía local. TAPI 2.1 incorpora la funcionalidad cliente-servidor[JAC97]. Esto permite a las aplicaciones para comunicaciones sobre clientes de red hacer uso de dispositivos conectados a otra computadora. Los componentes de TAPI 2.1 son:

- Las librerías de enlace dinámico TAPI.DLL y TAPI32.DLL, las cuales transmiten las solicitudes de la aplicación al servicio de telefonía para su procesamiento

[14] Call Centers.

- El servicio de telefonía TAPISRV.EXE, el cual implementa y administra las funciones de la interfaz TAPI
- Uno o más proveedores de servicios de telefonía

La versión 2.1 de TAPI soporta:

- El mismo conjunto de aplicaciones y proveedores de servicio que TAPI 2.0 cuando está instalada en Windows® NT, además de la funcionalidad adicional proporcionada por esta versión
- Las aplicaciones TAPI 1.3 y TAPI 1.4 de 16 bits, las aplicaciones TAPI 1.4 y TAPI 2.0 de 32 bits, los proveedores de servicio TAPI 1.3 y TAPI 1.4 de 16 bits y los proveedores de servicio TAPI 2.0 de 32 bits, cuando está instalado en Windows® 95

El extremo izquierdo de la Figura 2-5 muestra la arquitectura de TAPI 2.1. Las funciones TAPI invocadas por las aplicaciones son enviadas al servidor de telefonía TAPISRV.EXE a través de librerías de enlace dinámico TAPI.DLL y TAPI32.DLL, las cuales se omiten en la figura. El servidor de telefonía a su vez envía la solicitud realizada al proveedor de servicios correspondiente. La figura presenta cuatro proveedores de servicios de telefonía desarrollados por Microsoft®, que corresponden a los bloques etiquetados como Unimodem TSP, NDIS Proxy, H.323 e IP Multicast TSP. Además existen otros proveedores de servicios de telefonía desarrollados por terceros. Finalmente, podemos observar que el acceso a los servicios de red se lleva a cabo mediante la interfaz para sockets de Windows®

2.4.6 TAPI 3.0

TAPI 3.0 constituye la evolución de la TAPI 2.1 al modelo COM (Component Object Model)[MIC99B]. Los principales componentes de TAPI 3.0 son:

- *TAPI 3.0 COM API*

  Esta versión se implementa como un conjunto de objetos COM. La evolución de TAPI al modelo COM permite llevar a cabo actualizaciones de los componentes de facilidades TAPI.

- *Servidor de Telefonía*

  El proceso del servidor de telefonía o servidor TAPI (TAPISRV.EXE) abstrae la interfaz de los proveedores de servicios 2.1 y 3.0.

- *Proveedores de Servicio de Telefonía*

  TAPI 3.0 incorpora dos proveedores de telefonía por defecto: el TSP H.323 y el TSP de Conferencia Multipunto IP con sus respectivos MSPs.

- *Proveedores de Flujo de Medios*

  TAPI 3.0 provee una forma uniforme para acceder los flujos en una llamada, utilizando la API Direct Show como el controlador principal de flujo.

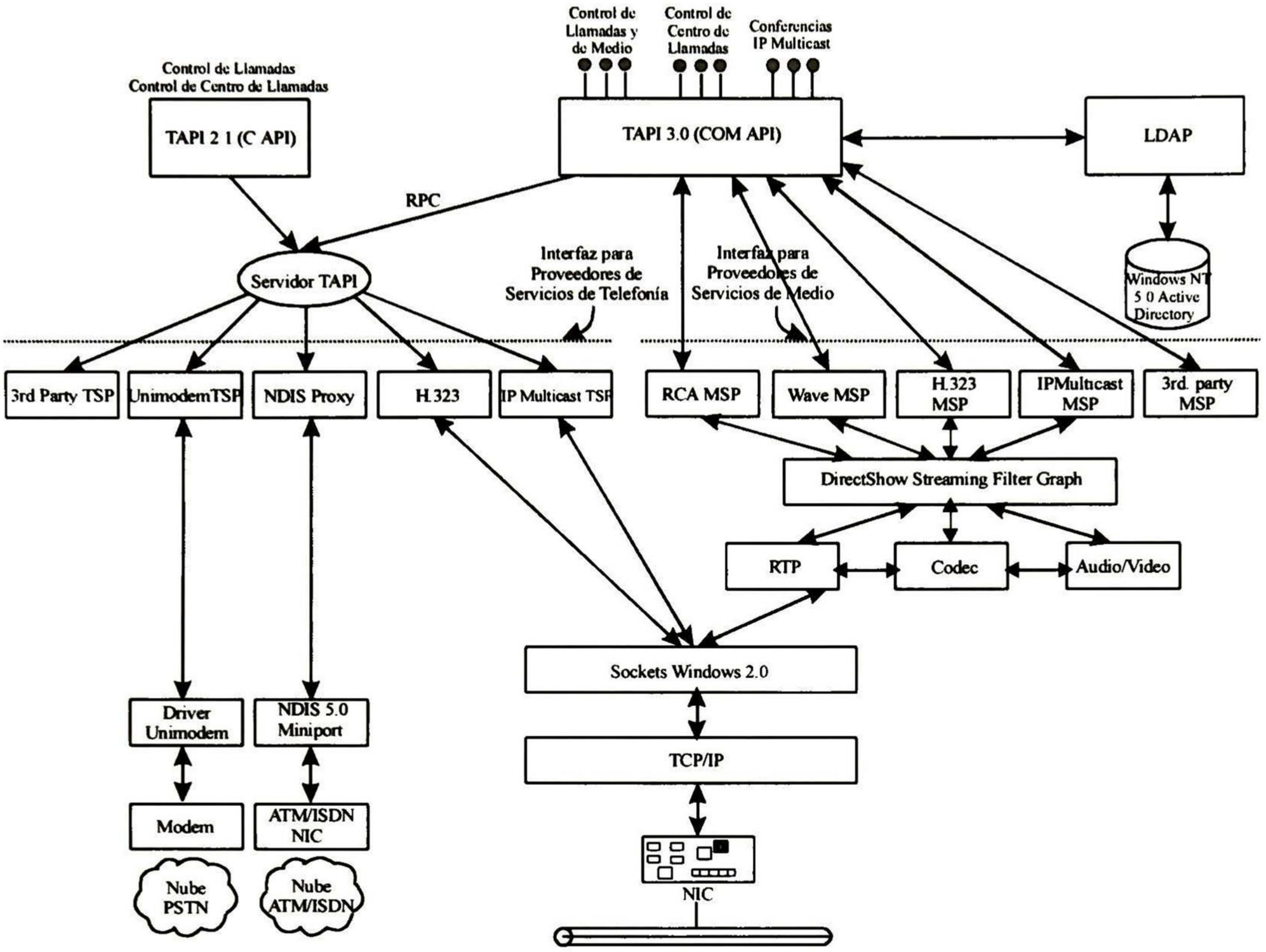


Figura 2-5. Arquitectura de TAPI 2.1 y TAPI 3.0

TAPI 3.0 tiene además las siguientes características:

- Incorpora control de flujos multimedios con la capacidad telefónica actual.
- Soporta el estándar H.323 y conferencias multipunto sobre IP, además de los proveedores de telefonía clásicos.
- Utiliza el servicio Active Directory de Windows® NT 5.0 para simplificar el "desplazamiento" de usuarios dentro de una organización. La dirección de red de un usuario (en este caso su dirección IP) es altamente volátil y no puede asumirse que permanecerá fija entre sesiones H.323. Por ello, el TSP H.323 de TAPI utiliza los servicios de Active Directory para realizar la resolución de direcciones usuario–IP.
- Soporta características de QoS para mejorar la calidad de conferencia y la administración de la red.

La Figura 2-5 presenta la arquitectura de TAPI 3.0. En ella puede apreciarse que las aplicaciones basadas en esta versión tienen acceso al control tanto de las llamadas telefónicas como del flujo de medios. Este control lo proporcionan los proveedores de

servicios de telefonía (TSPs) y de medios (MSPs) mediante las interfaces TSPI y MSPI, respectivamente. Algunos de los proveedores de servicios de telefonía disponibles en Windows® son el Unimodem y el de H.323. Para el control de flujo de medios se tienen los MSPs de H.323, el de IP Multicast, el Wave MSP para audio, así como otros desarrollados por terceros. Los MSPs trabajan en colaboración con otro grupo de elementos para realizar el transporte del flujo de medios sobre la red.

2.4.6.1 H.323

La Recomendación H.323 de la ITU-T describe las terminales y otras entidades que proporcionan servicios de comunicación multimedios sobre redes basadas en paquetes (packet based networks). Las redes basadas en paquetes pueden no proveer una calidad de servicio garantizada. Las terminales H.323 proporcionan capacidades para comunicaciones de audio, vídeo y/o datos en tiempo real. El soporte para audio es obligatorio, mientras que el de vídeo y datos es opcional. Si se soporta audio y vídeo se requiere la capacidad de usar un modo de operación común especificado para que todas las terminales que soporten estos medios puedan interactuar.

Las entidades de un sistema H.323 incluyen:

- *Terminales*

  Una terminal es un extremo de la red que proporciona comunicación en dos direcciones en tiempo real con otra terminal H.323, gateway o MCU [15] (Unidad de Control Multipunto). Esta comunicación consiste de control, indicaciones, audio, vídeo a color y/o datos entre dos terminales. Una terminal puede proveer solamente voz, voz y datos, voz y vídeo, o voz, datos y vídeo. Se le denomina extremo a una terminal H.323, a un gateway o a una MCU. Un extremo puede llamar y ser llamado y genera y/o termina flujos de información.

- *Gateways*

  Un gateway es un extremo de la red que proporciona comunicación en dos direcciones entre terminales H.323 en la red basada en paquetes y otras terminales ITU en una red de circuitos conmutados o con otro gateway H.323.

- *Gatekeepers*

  Un gatekeeper provee traducción de direcciones y controla el acceso a la red de las terminales H.323, gateways y MCUs. También les puede proporcionar otros servicios tales como administración de ancho de banda y localización de gateways.

- *Controladores Multipunto (MCs)*

  Un controlador multipunto provee el control de tres o más terminales participantes en una conferencia multipunto. Puede además conectar dos terminales en una conferencia punto a punto la cual puede convertirse posteriormente en una conferencia multipunto.

---

[15] MCU: Multipoint Control Unit.

Un MC da la capacidad de negociación con todas las terminales para lograr niveles comunes de comunicación. Un MC no realiza la combinación o conmutación de audio, vídeo y datos.

- *Procesadores Multipunto (MPs)*

  Un procesador multipunto suministra el procesamiento centralizado de flujos de audio, vídeo y/o datos en una conferencia multipunto. Un MP tiene la capacidad de realizar la combinación, conmutación u otro procesamiento de flujos de medios bajo el control del MC. Un MP puede procesar flujos de medios sencillos o múltiples dependiendo del tipo de conferencia soportado.

- *Unidades de Control Multipunto (MCUs)*

  Una unidad de control multipunto es un extremo de la red que proporciona a tres o más terminales y gateways la capacidad de participar en una conferencia multipunto. Puede además conectar dos terminales en una conferencia punto a punto la cual puede convertirse posteriormente en una conferencia multipunto. Una MCU generalmente opera en la forma de una MCU H.231[ITUH.231], sin embargo no es obligatorio un procesador de audio. Una MCU consiste de dos partes: una MC (requerido) y MPs (opcionalmente).

2.4.6.2 Modelo de Control de Llamada

El control de llamadas y del medio bajo TAPI 3.0 se lleva a cabo mediante un conjunto de objetos, interfaces y métodos (ver Figura 2-6). Específicamente en el control de llamadas de TAPI 3.0 intervienen cinco objetos, los cuales son:

- *TAPI*

  El objeto *TAPI* es el punto de entrada de la aplicación a TAPI 3.0. Este objeto representa todos los recursos de telefonía a los cuales la computadora local tiene acceso, permitiendo a la aplicación enumerar todas las direcciones locales y remotas.

- *Address*

  Un objeto *Address* representa el punto de origen o destino para una llamada. Las capacidades de cada dirección, tales como medios y terminal soportadas pueden obtenerse de este objeto. Una aplicación puede esperar por una llamada o puede crear un objeto de llamada saliente desde un objeto *Address*.

- *Terminal*

  Un objeto *Terminal* representa el recipiente en el punto inicial o final de una conexión. El objeto *Terminal* puede mapearse al hardware utilizado para interacción humana, por ejemplo un teléfono o un micrófono. También puede ser un archivo o cualquier otro dispositivo capaz de recibir alguna entrada o crear alguna salida.

- *Call*

El objeto *Call* representa la conexión entre la dirección local y una o más direcciones distintas (esta conexión puede ser hecha directamente o a través de un CallHub). Todo el control es hecho a través del objeto *Call*. El objeto *CallHub* contiene objetos *Call*.

- *CallHub*

  El objeto *CallHub* representa un conjunto de llamadas relacionadas. Un objeto *CallHub* no puede ser creado directamente por una aplicación, sino que se crea de manera indirecta cuando se reciben llamadas a través de TAPI 3.0. Usando un objeto *CallHub* un usuario puede conocer a los participantes en una llamada o conferencia. Incluso puede realizar control de llamada en los objetos *Call* remotos asociados con él (debido a la naturaleza independiente de localización del modelo COM). La capacidad de realizar control de llamadas depende de las restricciones que tenga el usuario que desea manipular la información.

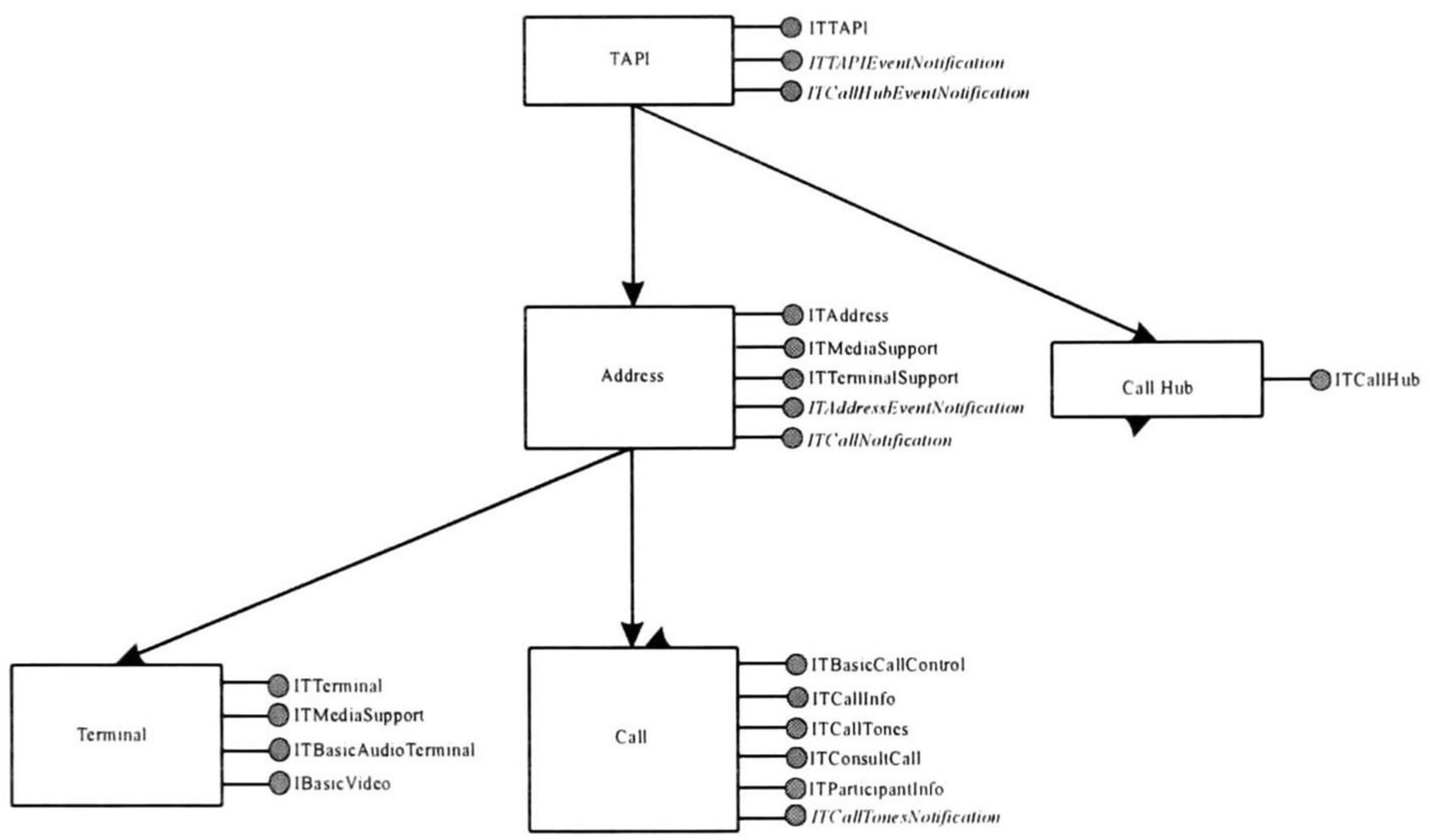


Figura 2-6. Objetos para el control de llamadas de TAPI 3.0

2.5 Crítica a TAPI

En la sección anterior se presentó una visión general de las distintas etapas por las que ha evolucionado el esquema de servicios de telefonía sobre Windows®. Tanto el conjunto de versiones orientadas a funciones como la versión orientada a componentes proporcionan una gama extensa de servicios para la gente que desarrolla software CTI, para los fabricantes de hardware y para los usuarios finales.

Las ventajas principales de utilizar las interfaces de telefonía de Microsoft® son:

- El empleo de TAPI para el desarrollo de aplicaciones CTI oculta al programador los detalles del hardware subyacente y de la red telefónica y al mismo tiempo proporciona compatibilidad a sus aplicaciones
- El uso de TSPI garantiza al fabricante de dispositivos y/o sistemas telefónicos que sus productos trabajarán con cualquier software de telefonía basado en TAPI
- Los usuarios finales pueden obtener el máximo provecho de un amplio rango de servicios y facilidades para comunicaciones

Sin embargo, luego de realizar un estudio detallado de TAPI se ha encontrado que el desarrollo de aplicaciones basadas en cualquiera de sus versiones orientadas a funciones crece en complejidad debido a ciertas desventajas de esta interfaz. Aunque estas desventajas serán revisadas en los capítulos siguientes, se mencionan a continuación:

- TAPI no proporciona el soporte para el proceso de múltiples llamadas telefónicas de manera simultánea
- TAPI no maneja el asincronismo de algunas de sus funciones
- La realización de operaciones telefónicas sencillas, tal como hacer una llamada, requiere de la ejecución de varias funciones
- El propio enfoque orientado a funciones dificulta el diseño de las aplicaciones que hacen uso de TAPI

Se supone que la versión orientada a componentes TAPI 3.0 resolverá estos problemas. Sin embargo esta versión aún se encuentra en proceso de desarrollo, no existe suficiente documentación técnica al respecto y por lo tanto no ha sido aplicada para la creación de software de telefonía.

Es por ello que surge la motivación para desarrollar un habilitador CTI basado en TAPI 2.1 que se encargue de los problemas puntualizados anteriormente. En el capítulo siguiente se describen con mayor detalle los conceptos y procesos principales involucrados en TAPI 2.1.

3 TAPI

3.1 Introducción

En el capítulo anterior se describió el esquema general de los servicios de telefonía sobre Windows® desarrollado por Microsoft®. Este esquema involucra un conjunto de interfaces con un propósito definido: TAPI, TSPI y MSPI. El habilitador CTI a desarrollar se basa en TAPI, específicamente en la versión 2.1. El objetivo de este capítulo es presentar los conceptos y el esquema de operación de TAPI 2.1.

3.2 Esquema de operación de TAPI

TAPI pertenece a la categoría de API básica dentro de la clasificación de habilitadores CTI presentada en el Capítulo 1 de este trabajo. Como tal, expone a las aplicaciones un conjunto de funciones para disponer de facilidades telefónicas o para ejecutar tareas de la misma naturaleza. Cuando alguna aplicación invoca una función TAPI, las librerías de enlace dinámico de telefonía procesan la llamada y la envían al proveedor de servicios de telefonía correspondiente. El proceso (ilustrado en la Figura 3-1) se detalla a continuación:

1. Una aplicación invoca alguna función de la interfaz TAPI
2. La DLL de telefonía correspondiente valida los parámetros de la llamada a la función y pasa el control al proceso de telefonía de Windows® (TAPISRV.EXE)
3. El proceso de telefonía determina el proveedor de servicios de telefonía correspondiente y enruta la petición hacia él

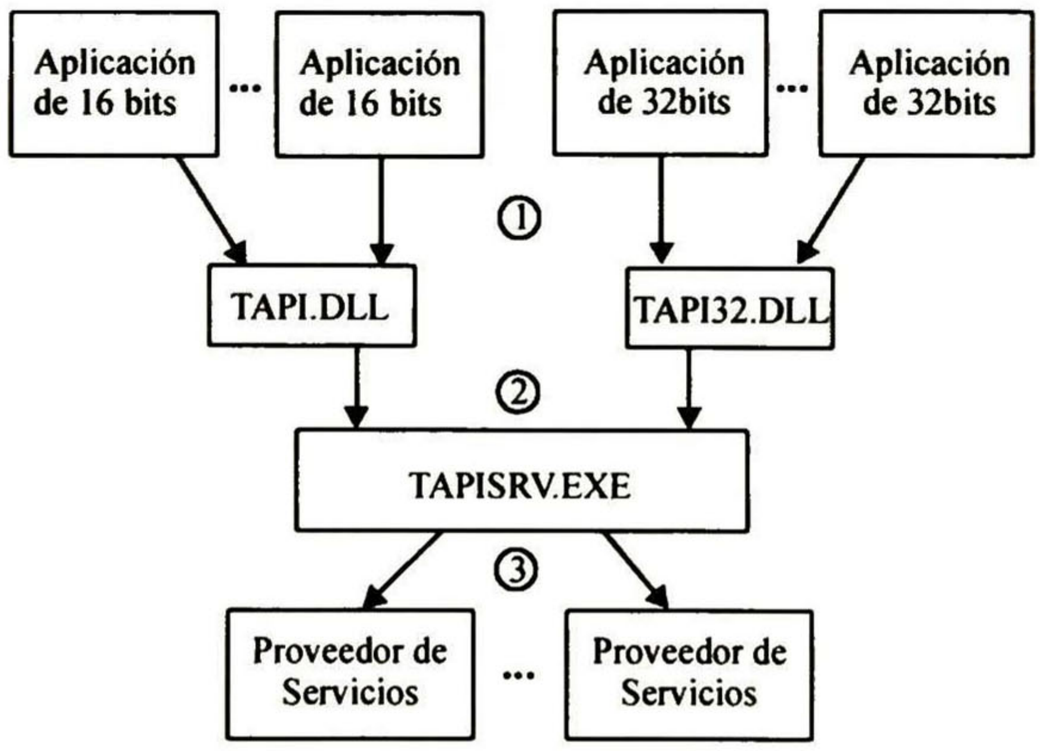


Figura 3-1. Descripción del proceso de ejecución de una función de la interfaz TAPI

3.3 Conceptos

El enfoque TAPI orientado a funciones se basa en el concepto de clases de dispositivos. Las clases de dispositivos proporcionan a las aplicaciones el acceso uniforme a los recursos y a los servicios telefónicos. TAPI define dos clases de dispositivos: dispositivos de línea (line devices – LD) y dispositivos telefónicos (phone devices PD). A continuación se definen los conceptos y el esquema de funcionamiento básicos de TAPI.

3.3.1 Clases de dispositivos

Definición 3.3.1.1 Una *clase de dispositivo (device class)* modela a un grupo de dispositivos físicos relacionados a través de los cuales las aplicaciones envían y reciben los datos y la información de una llamada telefónica.

Cada clase de dispositivo tiene asociado un conjunto de funciones para manipular los dispositivos pertenecientes a la clase. El concepto de clase de dispositivo simplifica el desarrollo de aplicaciones porque permite tratar a los dispositivos físicos que presentan propiedades semejantes de una forma similar.

Definición 3.3.1.2 La *clase dispositivo de línea (line device class)* representa una abstracción independiente de dispositivo de un canal físico compuesto por una línea telefónica y por el hardware que la controla. Por ejemplo, una tarjeta de fax, un modem, una tarjeta ISDN o el hardware del Asistente Telefónico.

Definición 3.3.1.3 Un *dispositivo de línea* es cualquier dispositivo que pertenece a la clase dispositivo de línea.

Cada dispositivo de línea tiene asociado un conjunto de capacidades[MIC97A]. Estas capacidades se almacenan en una estructura definida por TAPI. TAPI establece además una serie de mensajes y constantes mediante las cuales los proveedores de servicios de telefonía notifican a las aplicaciones acerca de cambios en el estado de los dispositivos de línea.

Definición 3.3.1.4 La *clase dispositivo telefónico (phone device class)* representa una abstracción independiente de dispositivo de un aparato telefónico.

Definición 3.3.1.5 Un *dispositivo telefónico* es la abstracción de un aparato telefónico que puede utilizarse para crear teléfonos "virtuales" dentro de una aplicación. Por ejemplo, si se tiene una PC con micrófono y bocinas puede implementarse la funcionalidad de un teléfono de escritorio.

Al igual que los dispositivos de línea, cada dispositivo telefónico tiene asociado un conjunto de capacidades, estructuras, mensajes y constantes para que las aplicaciones puedan manipularlos [MIC97A].

3.3.2 Direcciones

Definición 3.3.2.1 Una *dirección (address)* es el punto de origen o destino de una llamada telefónica.

Cada dispositivo de línea tiene asociadas una o más direcciones. Una dirección puede ser un número telefónico o un canal ISDN. Cada dirección tiene asignado un identificador *id* ∈ {0..n-1}, donde n es el número de direcciones asociadas con el dispositivo de línea. Dado que cada dirección depende de su dispositivo de línea para existir, el identificador de dirección es significativo sólo en el contexto del dispositivo de línea asociado. Algunos contextos en los que un dispositivo de línea tiene asociadas varias direcciones son los siguientes:

- *Múltiples direcciones con POTS*

  En POTS, la asignación de direcciones múltiples a una sola línea existe únicamente en sistemas que soportan timbrado distintivo o que están conectados a una troncal DID[16][ORE96][TEC99B]. En una línea residencial con servicio de timbrado distintivo, los distintos patrones de timbrado corresponden a múltiples direcciones asignadas a la misma línea. El DID también es un servicio que proporcionan las compañías telefónicas. Con DID en un sistema de correo de voz multiusuario, el número marcado es señalizado al sistema en la troncal DID antes de que el teléfono timbre. Esto le permite tocar el saludo o mensaje pregrabado correspondiente al destinatario de la llamada y almacenar cualquier mensaje entrante en el apartado de correo de voz indicado.

- *Múltiples direcciones con ISDN*

  En las líneas ISDN cada canal puede tener su propia dirección, lo cual significa que pueden existir tantas direcciones como canales proporcione la línea.

Las capacidades de un dispositivo de línea son compartidas por todas las direcciones asociadas con él. Cada dirección puede tener además características particulares.

3.3.3 Llamadas telefónicas

Definición 3.3.3.1 Una *llamada telefónica (call)* es una conexión entre dos o más direcciones con el objetivo de intercambiar información.

Definición 3.3.3.2 La *dirección origen* de una llamada telefónica es la dirección del extremo o de la estación desde la cual se realiza la llamada.

---

[16] DID: Direct Inward Dialing.

Definición 3.3.3.3 La *dirección destino* de una llamada telefónica identifica el extremo o la estación con la cual el solicitante de la llamada se desea comunicar.

TAPI define un conjunto de estados por los que las llamadas telefónicas evolucionan y una serie de eventos relacionados con los mismos. Algunos de estos estados y eventos son exclusivos de las llamadas entrantes, otros de las llamadas salientes y unos más son aplicables a ambos tipos. Varios de los estados de las llamadas proveen información adicional de estado que puede ser utilizada por las aplicaciones. Una aplicación puede tener alguno o ambos de los siguientes privilegios sobre una llamada:

- *Propietario*

  Este privilegio le concede a una aplicación el control sobre la existencia de una llamada telefónica.

- *Monitor*

  Como monitor una aplicación no puede controlar la existencia u otros aspectos de una llamada, pero puede registrar hechos acerca de la misma.

Las solicitudes de operaciones sobre llamadas telefónicas a través de TAPI – hacer y terminar llamadas, por ejemplo – se realizan mediante la invocación de algunas funciones asociadas con los dispositivos de línea, tales como *lineMakeCall* y *lineDrop*. Sin embargo, antes de realizar algunas operaciones telefónicas es necesario habilitar, obtener capacidades, abrir y configurar el dispositivo de línea y la dirección correspondientes. Por otra parte, TAPI no provee algún tipo de control para permitir la concurrencia de llamadas telefónicas. Es entonces responsabilidad de las aplicaciones implementar los procedimientos y controles necesarios para llevar a cabo estas acciones, en caso de que así lo requieran.

Los proveedores de servicios de telefonía (TSPs) pueden restringir las llamadas telefónicas a una por línea, permitir la conmutación de varias llamadas en una sola línea o manejar múltiples llamadas activas concurrentemente en una línea. Esto depende de las capacidades del dispositivo de línea que el proveedor de servicios de telefonía controle.

3.3.4 Niveles de servicios de telefonía en Windows®

TAPI define cuatro niveles de servicios:

- *Servicios de Telefonía Asistida*

  Este nivel de servicio permite incorporar la facilidad de marcación a aplicaciones que no están orientadas a telefonía, por ejemplo procesadores de texto, hojas de cálculo, etc. Debido a que este nivel de servicio es muy elemental, no incluye el uso de los dispositivos de línea y de los telefónicos.

- *Servicios Básicos de Telefonía*

Los servicios de telefonía básica constituyen un subconjunto mínimo de la especificación de telefonía para los sistemas operativos Windows®. La funcionalidad contenida en este nivel corresponde aproximadamente a la de POTS, esto es, proporciona servicios básicos para el manejo de llamadas entrantes y salientes:

- Habilitación y liberación de los dispositivos de línea
- Negociación de la versión de los dispositivos de línea
- Obtención del estado y las capacidades de los dispositivos de línea y las direcciones
- Apertura y cierre de los dispositivos de línea
- Traducción de direcciones en formato canónico a formato marcable
- Obtención de información referente a estados y eventos de llamadas telefónicas
- Realización de llamadas telefónicas
- Notificación y aceptación de llamadas entrantes
- Terminación de llamadas telefónicas

- *Servicios Suplementarios de Telefonía*

  Los servicios de telefonía suplementarios incorporan el uso de los dispositivos telefónicos así como los servicios avanzados que pueden encontrarse en los sistemas PBX, tales como "hold", "transfer", "conference", "park", etcétera. Entre sus principales funciones están:

  - Monitoreo y recolección de dígitos
  - Monitoreo de tonos
  - Generación de dígitos y tonos en banda
  - Aceptación y redireccionamiento de llamadas telefónicas
  - Transferencia de llamadas telefónicas
  - Conferencias telefónicas
  - Administración de proveedores de servicios
  - Administración de actividades e información referente a agentes (para aplicaciones de Centros de Llamadas)
  - Habilitación y liberación de los dispositivos telefónicos
  - Negociación de la versión de los dispositivos telefónicos
  - Obtención del estado y las capacidades de los dispositivos telefónicos
  - Apertura y cierre de los dispositivos telefónicos

- *Servicios Extendidos de Telefonía*

  Los servicios extendidos de telefonía (o específicos de un dispositivo) incluyen todas las extensiones a TAPI definidas por un proveedor de servicios particular. TAPI define únicamente el mecanismo de extensión. La definición del comportamiento del servicio extendido debe ser especificado completamente por el proveedor de servicios.

  Específicamente, el mecanismo de extensión de TAPI permite a los desarrolladores de proveedores de servicios definir nuevos valores para algunos tipos de enumeración y banderas. También es posible ampliar la mayoría de las estructuras de datos definidas en TAPI. La interpretación de las extensiones está sujeta al identificador de extensión del proveedor de servicios, esto es, a un identificador para la especificación del conjunto de extensiones soportadas. Este identificador único está compuesto por una

dirección de un adaptador Ethernet, un número aleatorio y la hora del día. Mensajes y funciones especiales tales como *lineDevSpecific* y *phoneDevSpecific* son propuestos en TAPI para permitir a una aplicación comunicarse directamente con el proveedor de servicios. Los parámetros de cada función son definidos también por el proveedor de servicios.

3.3.5 Acceso al medio

Definición 3.3.5.1 El *modo portador* (*bearer mode)* de una llamada es una indicación de la calidad de la conexión telefónica proporcionada principalmente por la red. Los modos portadores básicos definidos por TAPI son:

- *Voz*

  Indica un servicio de voz analógica de 3.1 kHz. Este modo portador puede soportar los media modes de fax y modem.

- *Audio G.711*

  Corresponde a la transmisión de audio G.711[ITUG.711] sobre la llamada telefónica.

- *Datos*

  Se refiere a la transferencia de datos sin restricción sobre la llamada telefónica.

Definición 3.3.5.2 El *media mode* de una llamada es una indicación del tipo de flujo de información que sea intercambia sobre la llamada. Los media modes principales que define TAPI son:

- *Voz interactiva*

  Indica la presencia de voz sobre la llamada; ésta es tratada como una llamada interactiva con personas en ambos extremos.

- *Voz automatizada*

  Señala la presencia de voz sobre la llamada y la voz es manejada localmente por una aplicación automatizada.

- *Datamodem*

  Se utiliza cuando se transfieren datos a través de un modem.

- *Fax G3*

  Un fax del grupo 3 se envía o se recibe sobre la llamada.

- *Fax G4*

  Un fax del grupo 4 se envía o se recibe sobre la llamada.

- *Datos digitales*

Datos digitales se envían o se reciben sobre la llamada.

Definición 3.3.5.3 El término *media stream* se emplea para denotar el flujo actual de información que se intercambia en una llamada telefónica.

Los dispositivos de línea y los dispositivos telefónicos son capaces de transportar un flujo de información. TAPI sólo proporciona control sobre los dispositivos de línea y los dispositivos telefónicos, pero no provee acceso al flujo de información. Para ello una aplicación debe utilizar otro grupo de APIs, tales como Media Control Interface API[MIC99C].

3.3.6 Funciones síncronas y asíncronas

La naturaleza interactiva de la telefonía requiere que TAPI proporcione un ambiente operativo de tiempo real. Muchas de las funciones TAPI necesitan completarse rápidamente y regresar resultados a la aplicación de manera síncrona. Otras funciones, tales como la marcación de dígitos, quizás no sean capaces de completarse tan rápidamente y entonces deben operar de forma asíncrona.

Definición 3.3.6.1 Una *función síncrona* realiza todo su procesamiento antes de que se le devuelva el control a la aplicación.

Definición 3.3.6.2 Una *función asíncrona* realiza parte de su procesamiento durante el tiempo que tiene el control de la aplicación y el resto en un hilo de ejecución independiente.

Una aplicación recibe dos tipos de información como resultado de invocar una función TAPI: el valor de retorno de la función y valores escritos a localidades de datos especificadas por los argumentos de la función. El valor de retorno de una función TAPI debe ser un entero positivo, un entero negativo o cero. La interpretación del valor de retorno, dada por TAPI, es la siguiente:

- *El valor de retorno es un número entero negativo*

  Indica un mensaje de error. TAPI define un conjunto de mensajes de error para las funciones asociadas con los dispositivos de línea y otro conjunto para las funciones asociadas con los dispositivos telefónicos[MIC97A].

- *El valor de retorno es cero*

  Indica a la aplicación que la función se ha completado con éxito de manera síncrona. En este caso, los valores escritos como resultado de la llamada a la función son confiables y pueden ser usados inmediatamente.

- *El valor de retorno es un número entero positivo*

  Indica a la aplicación que la función se completará de forma asíncrona. El valor de retorno es el identificador de la respuesta asíncrona pendiente (reply). TAPI notifica a una aplicación de la consumación de una función asíncrona enviándole un mensaje de

respuesta asíncrona. Una vez que la aplicación recibe este mensaje (y éste indica éxito) los parámetros de la función se consideran confiables. La mayoría de las funciones asíncronas de TAPI son aquéllas que realizan operaciones sobre llamadas telefónicas: hacer llamada, terminar llamada, aceptar llamada, entre otras.

Una aplicación procesa las funciones síncronas de manera trivial: invoca a la función e "inmediatamente" recibe un resultado de éxito o fracaso (Figura 3-2 inciso A) Pero las funciones asíncronas trabajan de manera distinta. La aplicación recibe el resultado tiempo después de que la función invocada le ha devuelto el control (Figura 3-2 inciso B). El problema es que existe una restricción: hay algunas funciones – tanto síncronas como asíncronas – que no pueden ser invocadas antes de que otras funciones asíncronas se hayan completado. Las funciones asíncronas que deben completarse son aquéllas que proporcionan a las aplicaciones los identificadores (handlers) de las llamadas telefónicas. Por lo tanto, las aplicaciones deben asegurarse que este tipo de funciones asíncronas hayan completado su ejecución antes de que otras funciones sean invocadas.

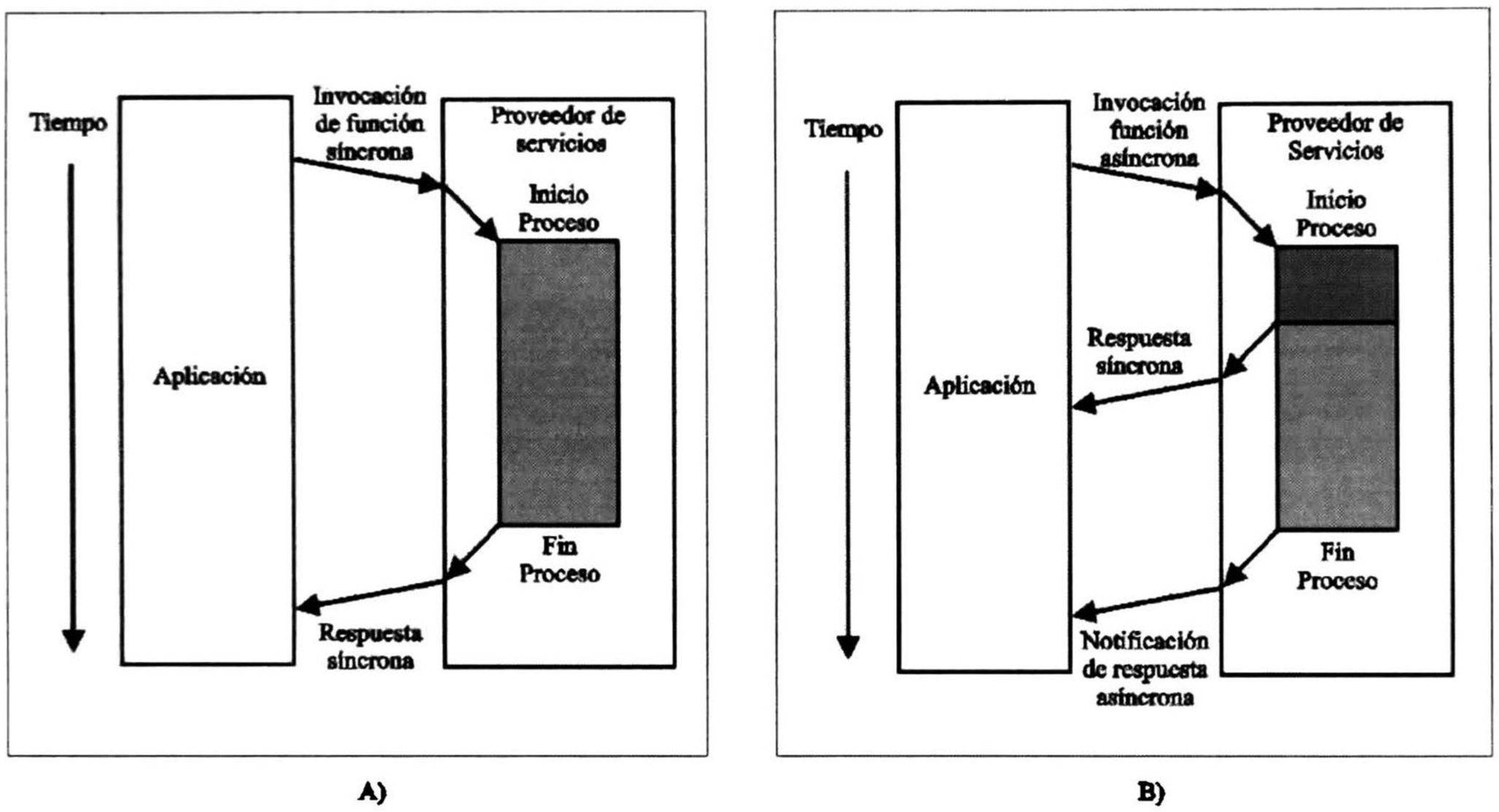


Figura 3-2. Proceso de funciones síncronas y asíncronas de TAPI

3.3.7 Mecanismos de Notificación

Una aplicación debe seleccionar uno de tres mecanismos mediante los cuales TAPI informa a la aplicación de los eventos telefónicos. Los mecanismos de notificación definidos por la versión 2.1 de TAPI son:

- *Ventana Oculta (Hidden Window)*

  Este mecanismo es el único disponible para las versiones 1.x de TAPI. Cuando se utiliza, TAPI crea una ventana en el contexto de la aplicación en la cual el sistema operativo escribe los mensajes destinados para la misma.

- *Eventos (Event Handle)*

  Cuando se utiliza este mecanismo, TAPI crea un objeto de tipo evento y retorna un handler para dicho objeto a la aplicación. TAPI señaliza este evento a la aplicación en cualquier momento que un evento de telefonía está pendiente para la misma.

- *Puertos de Terminación (Completion Port)*

  Cuando se utiliza este mecanismo, la aplicación especifica a TAPI un puerto de terminación al cual le debe enviar todos los eventos telefónicos.

4 Especificación de Requerimientos del Habilitador Telefónico basado en TAPI

4.1 Introducción

En el capítulo 2 se mencionaron las ventajas y desventajas de que las aplicaciones utilicen TAPI como interfaz para el acceso a los recursos y servicios de telefonía en Windows®. Ahí también se plantea la motivación para el desarrollo de un habilitador CTI basado en TAPI, lo cual constituye el objetivo de este trabajo de tesis. El propósito de este capítulo es presentar los requerimientos funcionales del habilitador CTI a desarrollar, al que denominaremos Habilitador Telefónico basado en TAPI (HTT).

4.2 Funcionalidad del Habilitador Telefónico basado en TAPI (HTT)

HTT pertenece a la categoría de API de alto nivel dentro de la clasificación de habilitadores CTI presentada en el capítulo 1. Puede ser utilizado por aplicaciones que requieran servicios básicos de telefonía o para desarrollar middleware CTI en ambiente Windows®. El propósito del HTT es encapsular la funcionalidad de TAPI para proporcionar a las aplicaciones una interfaz más sencilla que les permita acceder a los recursos y servicios telefónicos. Este propósito tiene dos implicaciones. La primera de ellas es que HTT proporcione la funcionalidad completa de TAPI. Esta funcionalidad es muy amplia, de acuerdo con lo que se explicó en el capítulo 2. Por lo tanto, HTT sólo dará soporte a las funciones correspondientes al nivel de servicios básicos de telefonía. Estas funciones comprenden el manejo básico de llamadas telefónicas entrantes y salientes y la administración de los dispositivos de línea y las direcciones.

La segunda implicación es que se encargue de los aspectos negativos que incrementan la complejidad del desarrollo de aplicaciones basadas en TAPI: la carencia del soporte para el proceso de múltiples llamadas telefónicas de manera simultánea, el asincronismo de las funciones que realizan operaciones sobre llamadas telefónicas, la cantidad de funciones requeridas para llevar a cabo operaciones simples y el enfoque orientado a funciones.

4.3 Ventajas del Habilitador Telefónico basado en TAPI

Una vez que se ha planteado la funcionalidad de HTT, pueden establecerse las ventajas proporcionadas por HTT a las aplicaciones basadas en TAPI:

- La disminución en complejidad del diseño de las aplicaciones, ya que todo el acceso a los recursos y servicios telefónicos se realiza a través de HTT, el cual se encarga de realizar el procesamiento de las funciones asíncronas, de proporcionar el soporte para la concurrencia de llamadas telefónicas y de reducir la cantidad de invocaciones a funciones TAPI por parte de la aplicación
- La reducción del tiempo de desarrollo dado que HTT puede ser reutilizado por cualquier aplicación
- La disminución de la probabilidad de errores de codificación debido al aislamiento de los aspectos de bajo nivel de TAPI

4.4 El Asistente Telefónico

En la sección 1.4 se presentó el proyecto denominado Asistente Telefónico. El Asistente Telefónico es una solución CTI para la administración de los recursos telefónicos en ambiente Windows®[17] Provee al usuario con facilidades como el control de llamadas telefónicas entrantes y salientes, tarificación, correo de voz, entre otras. La arquitectura del Asistente Telefónico consta de los siguientes módulos de software (ver Figura 4-1): Interfaces de Usuario, Control Administrativo de Llamadas Telefónicas (CALT), Habilitador Telefónico basado en TAPI (HTT), Proveedor de Servicios (ATSP) y Controlador Virtual (VATD). La zona sombreada corresponde al módulo HTT, el cual interactúa con el módulo CALT y con las DLLs y el proceso de servicio de telefonía de Windows® El proceso de desarrollo de HTT se mostrará desde la perspectiva del Asistente Telefónico. Se decidió utilizar este contexto para ilustrar la interacción de HTT con una aplicación real. Antes de entrar de lleno a la especificación de requerimientos de HTT se describirá brevemente la funcionalidad de cada uno de los módulos que conforman el Asistente Telefónico.

4.4.1 Interfaces de usuario

Este módulo agrupa los componentes de la aplicación con los que el usuario final de la misma interactúa directamente[CAR00]. Los componentes principales son:

- Control de llamadas entrantes y salientes en varias líneas
- Máquina contestadora

---

[17] Microsoft Windows® 95 y Windows® 98.

- Correo de voz
- Envío automático de mensajes y programación de llamadas
- Directorio personal
- Teléfono en pantalla
- Tarificación

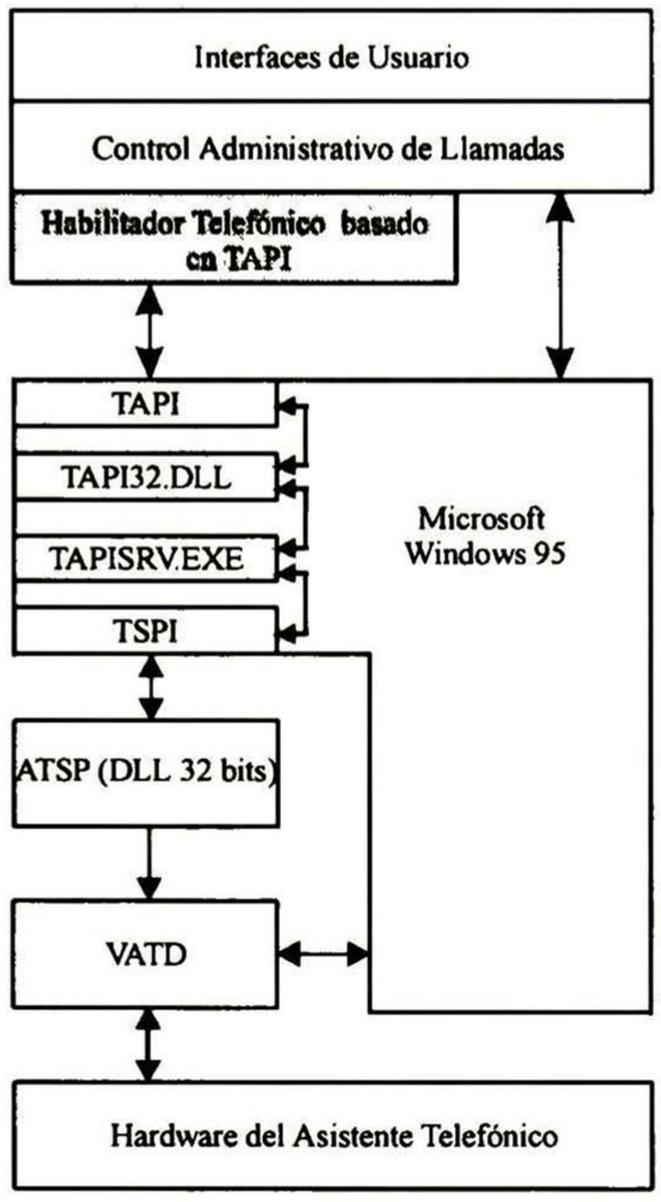


Figura 4-1. Arquitectura del Asistente Telefónico

4.4.2 Control Administrativo de Llamadas Telefónicas (CALT)

El objetivo de este módulo es lograr la identificación, tarificación y restricción tanto de las llamadas entrantes y salientes como de los usuarios telefónicos, atendiendo siempre a un conjunto de reglas e historiales que permiten efectuar estas acciones[CAR00].

4.4.3 Habilitador Telefónico basado en TAPI (HTT)

El propósito de este módulo es brindar la interfaz necesaria para que tanto las interfaces de usuario como el control administrativo de llamadas accedan a los recursos y servicios telefónicos provistos por el hardware del Asistente Telefónico. Este módulo traduce las solicitudes de los módulos superiores en mensajes definidos por TAPI y les notifica acerca de los resultados de las mismas y de cambios en el estado de los recursos telefónicos.

4.4.4 Proveedor de servicios del Asistente Telefónico (ATSP)

El ATSP[XIM00] implementa las funciones TSPI necesarias para soportar los servicios telefónicos solicitados por las DLLs de telefonía y las aplicaciones. Estas solicitudes incluyen el registro e inicialización del ATSP, información acerca del estado de los recursos y ejecución de operaciones telefónicas. De igual manera, el ATSP responde a las notificaciones de cambios en el estado del hardware del Asistente Telefónico provenientes del controlador virtual VATD.

4.4.5 Controlador virtual del hardware del Asistente Telefónico (VATD)

El VATD[FIG00] es un dispositivo virtual para Windows® 95 que implementa el conjunto de funciones de bajo nivel que se requieren para llevar a cabo la funcionalidad del Asistente Telefónico. Estas funciones de bajo nivel actúan directamente sobre el hardware del Asistente Telefónico a petición del ATSP. El VATD se encarga además de notificar al ATSP cualquier cambio en el estado del hardware.

4.5 Especificación de Requerimientos del Habilitador Telefónico basado en TAPI (HTT)

La especificación de requerimientos del software HTT que se presenta a continuación se basa en una adaptación de las recomendaciones prácticas para especificaciones de requerimientos de software de la IEEE descritas en el estándar 830[IEEE830]. Su propósito es plantear los requerimientos que debe satisfacer HTT para proveer a las aplicaciones la funcionalidad descrita en la sección 4.2. Los requerimientos de diseño de HTT relacionados con la implementación del soporte para el procesamiento de múltiples llamadas y la reducción de invocaciones a funciones TAPI por parte de las aplicaciones se tratarán en el capítulo siguiente.

4.5.1 Requerimientos de interfaces externas

Las interfaces externas están constituidas por todas aquéllas entradas al software HTT y por todas las salidas que éste produce. Las entradas consisten básicamente de solicitudes de servicios originadas en el módulo denominado Control Administrativo de Llamadas Telefónicas (CALT) y de notificaciones de eventos ocurridos en los dispositivos de línea, las direcciones y las llamadas telefónicas, provenientes de las DLLs de telefonía. Las salidas se componen de un grupo de mensajes destinados al módulo CALT y de un conjunto de invocaciones a funciones TAPI. Por cuestiones de espacio, todas las interfaces externas se describen en [PAD00]. Las descripciones de las funciones y constantes definidas por TAPI a las que se hace referencia en esta especificación de requerimientos también pueden encontrarse en [PAD00] y con mayor detalle en [MIC97A].

4.5.2 Facilidades proporcionadas por HTT

Los requerimientos de software de HTT se describen en términos de las facilidades que éste proporciona. Una *facilidad* es un servicio deseado externamente que puede requerir una secuencia de entradas para producir el resultado deseado. Para cada facilidad se definen los aspectos siguientes:

- *Propósito*

  Establece en qué consiste la facilidad.

- *Descripción*

  Presenta una breve descripción de la facilidad.

- *Secuencia estímulo/respuesta*

  Muestra mediante cronogramas la secuencia de estímulos (entradas) y respuestas (salidas) que recibe y genera HTT, respectivamente, para completar la facilidad correspondiente. Las líneas sólidas representan estímulos y las líneas punteadas respuestas.

- *Seudocódigo*

  Define las acciones fundamentales que HTT debe realizar para aceptar y procesar las entradas y para generar las salidas correspondientes: validación de las entradas, secuencia exacta de operaciones, manejo y recuperación de errores, efecto de los parámetros y relación de entradas con salidas.

Las facilidades proporcionadas por HTT pueden clasificarse en tres grupos: administración de los dispositivos de línea y de las direcciones, operaciones sobre llamadas telefónicas y procesamiento de eventos provenientes de las DLLs de telefonía. Las facilidades son:

- *Administración de los dispositivos de línea y de las direcciones*
  - Habilitación de los dispositivos de línea
  - Obtención de información de las capacidades de los dispositivos de línea y de las direcciones
  - Configuración de las direcciones para habilitar la notificación de llamadas entrantes
  - Liberación de los dispositivos de línea y de las direcciones
  - Deshabilitación de los dispositivos de línea
- *Operaciones sobre llamadas telefónicas*
  - Realización de llamadas
  - Marcación de dígitos sobre una llamada
  - Aceptación y contestación de llamadas telefónicas basadas en el callerID
  - Terminación de llamada telefónica desde software

- *Procesamiento de eventos provenientes de las DLLs de telefonía*
  - Procesamiento de las notificaciones de consumación de funciones asíncronas
  - Procesamiento de las notificaciones de cambio de estado de las llamadas telefónicas
  - Procesamiento de las notificaciones de cambio de estado de los dispositivos de línea
  - Procesamiento de las notificaciones de cambio de estado de las direcciones
  - Procesamiento de las notificaciones de llamadas salientes realizadas desde el aparato telefónico
  - Procesamiento de las notificaciones de llamadas entrantes

4.5.2.1 Habilitación de los dispositivos de línea

4.5.2.1.1 Propósito

Esta facilidad habilita a HTT para hacer uso de los dispositivos de línea y de las direcciones y le permite conocer el número de dispositivos de línea disponibles así como el "handler" que identifica a HTT como usuario de los dispositivos de línea.

4.5.2.1.2 Descripción

Los *handlers* constituyen el mecanismo mediante el cual las DLLs de telefonía identifican de manera única a las aplicaciones cliente de TAPI, a los dispositivos de línea, a las direcciones e incluso a las llamadas telefónicas. Cuando se lleva a cabo la habilitación de los dispositivos de línea también se registra ante las DLLs de telefonía el mecanismo de notificación de eventos relacionados con las llamadas telefónicas, con los dispositivos de línea y con las direcciones. El mecanismo de notificación utilizado por HTT es el de ventana oculta (ver detalles en capítulo 3).

4.5.2.1.3 Secuencia estímulo/respuesta

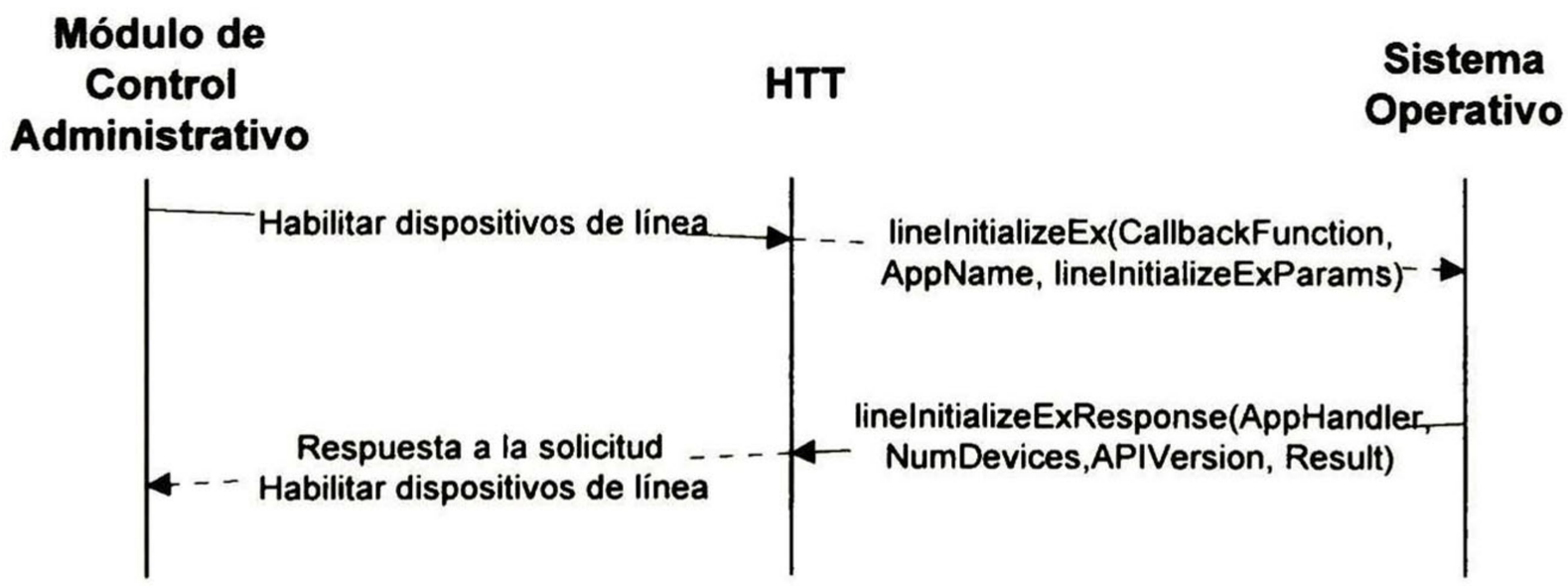


Figura 4-2. Secuencia estímulo/respuesta para la habilitación de los dispositivos de línea

4.5.2.1.4 Seudocódigo

Solicitud de habilitación de los dispositivos de línea( ):
Verificar si se ha llevado a cabo previamente la habilitación de los dispositivos de línea y las direcciones
Si ya se ha llevado a cabo la habilitación **entonces**
  enviar el mensaje ERR_TAPIALREADYINITIALIZED al módulo CALT
en otro caso
  invocar la función TAPI lineInitializeEx
  Si el valor de retorno de lineInitializeEx es igual a SUCCESS **entonces**
    Si el número de dispositivos de línea es igual a cero **entonces**
      enviar el mensaje ERR_NOAVAILABLERESOURCES al módulo CALT
    en otro caso
      almacenar el handler válido que identifica a HTT como usuario de los dispositivos de línea, el número de dispositivos de línea disponibles y la versión de TAPI con la que se implementa HTT, todos ellos valores devueltos por la función lineInitializeEx
      enviar el mensaje SUCCESS al módulo CALT
en otro caso
  enviar el mensaje ERR_TAPIINITIALIZEFAILED al módulo CALT

4.5.2.2 Obtención de información de las capacidades de los dispositivos de línea y de las direcciones

4.5.2.2.1 Propósito

Esta facilidad recupera de las DLLs de telefonía la información de las capacidades de cada dispositivo de línea y de sus direcciones asociadas. La información retornada debe ser almacenada por HTT para su uso posterior mientras el módulo CALT no solicite la deshabilitación de los dispositivos de línea.

4.5.2.2.2 Descripción

Las capacidades de los dispositivos de línea definidas por TAPI (ver Tabla 4-1) que se consideran significativas para HTT y para el módulo CALT son:

- PermanentLineID
- NumAddresses
- BearerModes
- MediaModes
- MaxNumActiveCalls
- LineState

Nombre de la Capacidad	Descripción
PermanentLineID	El identificador permanente a través del cual un dispositivo de línea es conocido en la configuración del sistema. Este identificador no cambia cuando se agregan o eliminan nuevos dispositivos de línea del sistema operativo.
NumAddresses	El número de direcciones asociadas con un dispositivo de línea.
BearerModes	Son una indicación de la calidad de la conexión telefónica proporcionada por la red. Los bearer modes soportados por el hardware del Asistente Telefónico son: • LINEBEARERMODE_VOICE Indica un servicio de voz analógica de 3.1 kHz. • LINEBEARERMODE_DATA Indica una transferencia no restringida de datos sobre la llamada.
MediaModes	Son una indicación del tipo de flujo de información que sea intercambia sobre la llamada. Los media modes soportados por el hardware del Asistente Telefónico son: • LINEMEDIAMODE_INTERACTIVEVOICE Indica la presencia de voz sobre la llamada; ésta es tratada como una llamada interactiva con personas en ambos extremos. • LINEMEDIAMODE_AUTOMATEDVOICE Señala la presencia de voz sobre la llamada y la voz es manejada localmente por una aplicación automatizada.
MaxNumActiveCalls	El número máximo de llamadas que pueden estar activas en un dispositivo de línea en cualquier momento. Una llamada activa es aquella que está en algún estado diferente de Ocioso, onhold, onholdpendingtransfer y onholdpendingconference.
LineState	Especifica los diferentes estados de los dispositivos de línea para los cuales la aplicación puede ser notificada.

Tabla 4-1. Capacidades de los dispositivos de línea definidas por TAPI

Las capacidades de las direcciones definidas por TAPI (ver Tabla 4-2) que se consideran significativas para HTT y para el módulo CALT son:

- LineDeviceID

- AddressStates
- CallerIDFlags
- CalledIDFlags
- AddressFeatures

Nombre de la Capacidad	Descripción
LineDeviceID	El identificador de dispositivo del dispositivo de línea con el cual está asociada una dirección.
AddressStates	Contiene los cambios de estado de una dirección para los cuales la aplicación puede ser notificada.
CallerIDFlags	Describe el tipo de información relacionada con el callerID que puede ser proporcionado para llamadas en una dirección. La información relevante para el habilitador CTI y para el módulo CALT es: • LINECALLPARTYID_BLOCKED La información del callerID es bloqueada por él mismo, pero podría estar disponible en otro caso. • LINECALLPARTYID_ADDRESS La información del callerID para la llamada es el número de la persona que llama y es proporcionada en el campo callerID. • LINECALLPARTYID_UNAVAIL La información del callerID no está disponible y no puede ser conocida posteriormente.
CalledIDFlags	Describe el tipo de información relacionada con el calledID que puede ser proporcionado para llamadas en una dirección. La información significativa para el habilitador CTI y para el módulo CALT es: • LINECALLPARTYID_ADDRESS La información del calledID para la llamada es el número con el que se desea establecer la comunicación y es proporcionada en el campo calledID.
AddressFeatures	Las funciones API relacionadas con dirección que pueden ser invocadas en una dirección en el estado actual. La única capacidad de este tipo definida por TAPI que soporta el hardware del Asistente Telefónico es: • LINEFEATURE_MAKECALL La dirección tiene la capacidad de realizar llamadas.

Tabla 4-2. Capacidades de las direcciones definidas por TAPI

4.5.2.2.3 Secuencia estímulo/respuesta

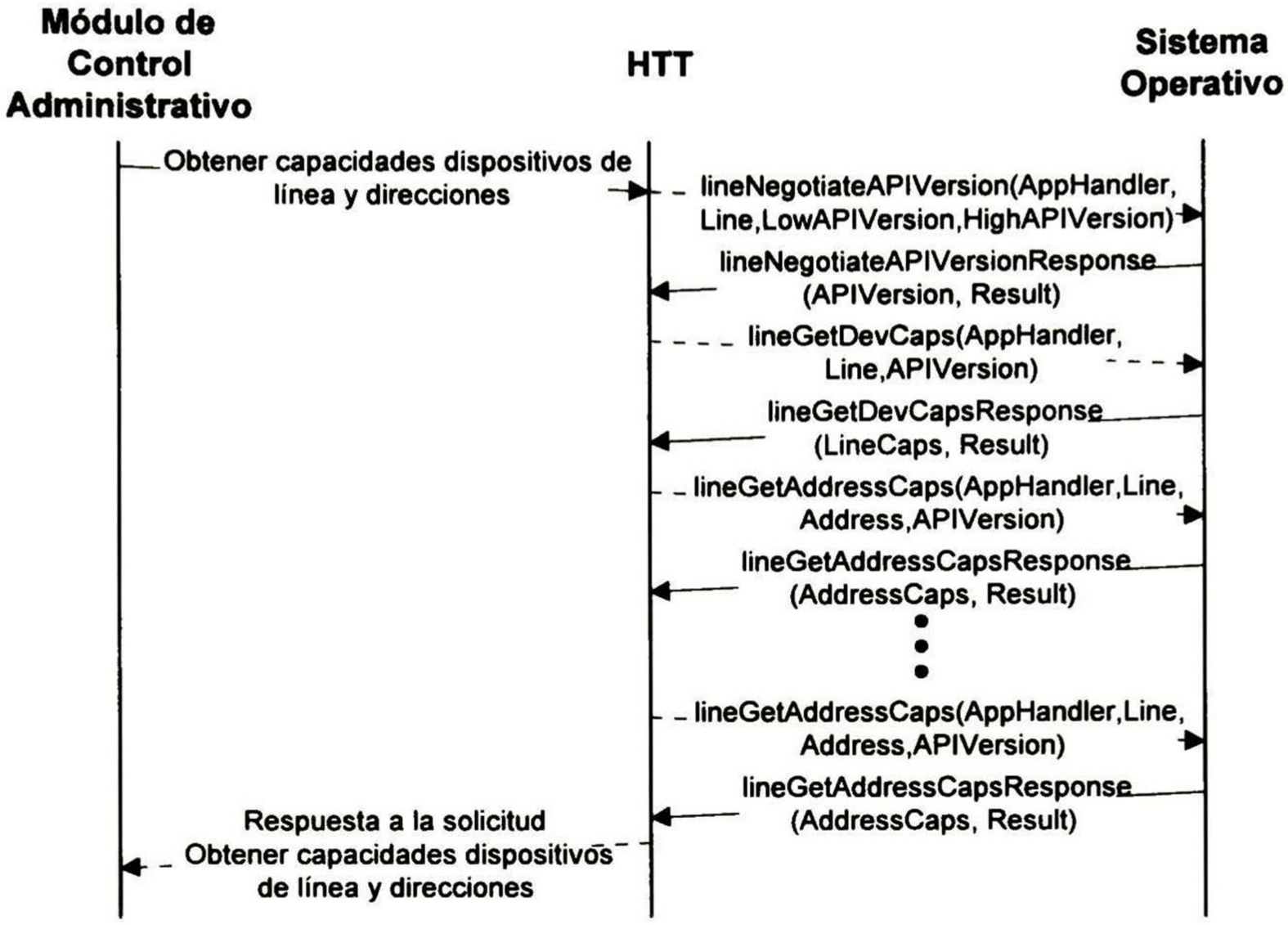


Figura 4-3. Secuencia estímulo/respuesta para la obtención de las capacidades de los dispositivos de línea y direcciones

4.5.2.2.4 Seudocódigo

Solicitud de obtención de capacidades de los dispositivos de línea( ):
Por cada dispositivo de línea realizar las siguientes acciones:
Invocar la función TAPI lineNegotiateAPIVersion
Si el valor de retorno de lineNegotiateAPIVersion es igual a SUCCESS **entonces**
  almacenar la información de versión del dispositivo de línea
  invocar la función TAPI lineGetDevCaps
  Si el valor de retorno de lineGetDevCaps es igual a SUCCESS **entonces**
    almacenar la información de capacidades del dispositivo de línea
    por cada dirección asociada con el dispositivo de línea realizar las siguientes acciones:
    invocar la función TAPI lineGetAddressCaps
    Si el valor de retorno de lineGetAddressCaps es igual a SUCCESS **entonces**
      almacenar la información de capacidades de la dirección
    en otro caso
      continuar con la siguiente dirección
  en otro caso
    continuar con el siguiente dispositivo de línea
en otro caso
  continuar con el siguiente dispositivo de línea
Si no se obtuvieron exitosamente las capacidades de por lo menos un dispositivo de línea y de sus direcciones asociadas **entonces**
  enviar el mensaje ERR_NOAVAILABLERESOURCES al módulo CALT

```
en otro caso
  enviar el mensaje SUCCESS al módulo CALT
```

4.5.2.3 Configuración de las direcciones para habilitar la notificación de llamadas entrantes

4.5.2.3.1 Propósito

Esta facilidad habilita una dirección asociada con algún dispositivo de línea para recibir notificaciones de llamadas telefónicas entrantes. Como resultado de la configuración HTT obtiene un "handler" válido asociado con la dirección especificada.

4.5.2.3.2 Descripción

El "handler" retornado se utilizará para cualquier operación que se realice sobre la dirección especificada o para realizar llamadas sobre la misma.

4.5.2.3.3 Secuencia estímulo/respuesta

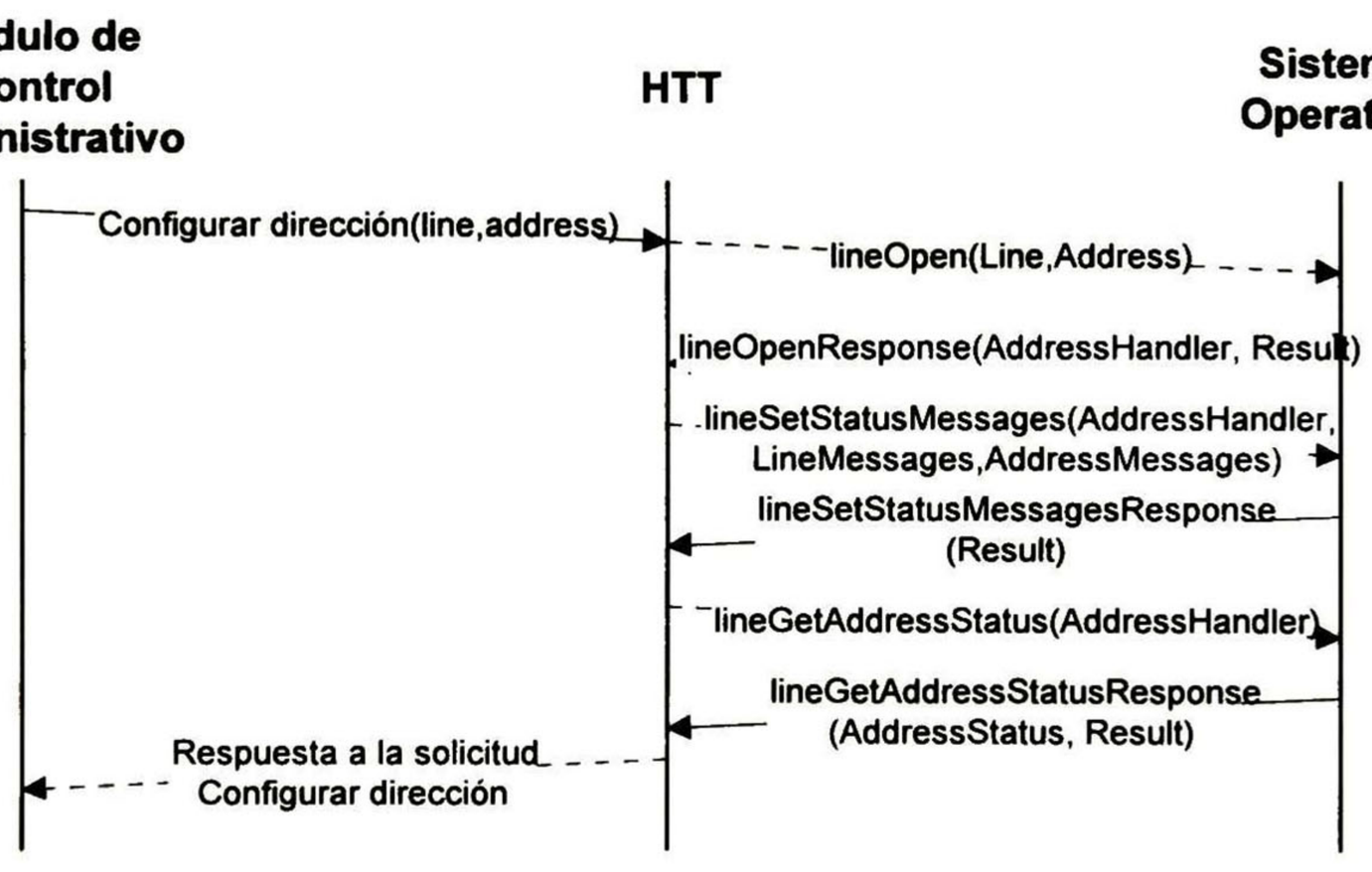


Figura 4-4. Secuencia estímulo/respuesta para la configuración de una dirección para habilitar la notificación de llamadas entrantes

4.5.2.3.4 Seudocódigo

```
Solicitud de configuración de dirección(lineID: identificadorDispositivoLínea,
                                        addressID: identificadorDirección):
Si existe un dispositivo de línea con identificador igual a lineID entonces
  Si existe una dirección con identificador igual a addressID que está asociada con el dispositivo
     de línea con identificador igual a lineID entonces
    invocar la función TAPI lineOpen
    Si el valor de retorno de lineOpen es igual a SUCCESS entonces
```

```
almacenar el handler válido asociado con la dirección con identificador igual a addressID devuelto
por lineOpen
invocar la función TAPI lineSetStatusMessages
Si el valor de retorno de lineSetStatusMessages es igual a SUCCESS entonces
  invocar la función TAPI lineGetAddressStatus
  Si el valor de retorno de lineGetAddressStatus es igual a SUCCESS entonces
    almacenar la información de estado de la dirección con identificador igual a addressID
    enviar el mensaje SUCCESS al módulo CALT
  en otro caso
    enviar el mensaje ERR_GETADDRESSSTATUSFAILED al módulo CALT
en otro caso
    en otro caso
      enviar el mensaje ERR_SETMESSAGESFAILED al módulo CALT
    en otro caso
      enviar el mensaje ERR_OPENLINEFAILED al módulo CALT
  en otro caso
    enviar el mensaje ERR_ADDRESSNOTFOUND al módulo CALT
en otro caso
  enviar el mensaje ERR_LINENOTFOUND al módulo CALT
```

4.5.2.4 Liberación de los dispositivos de línea y de las direcciones

4.5.2.4.1 Propósito

Esta facilidad libera todos los handlers asociados con los dispositivos de línea y con las direcciones.

4.5.2.4.2 Descripción

Esta facilidad debe ser invocada antes de llevar a cabo la deshabilitación de los dispositivos de línea.

4.5.2.4.3 Secuencia estímulo/respuesta

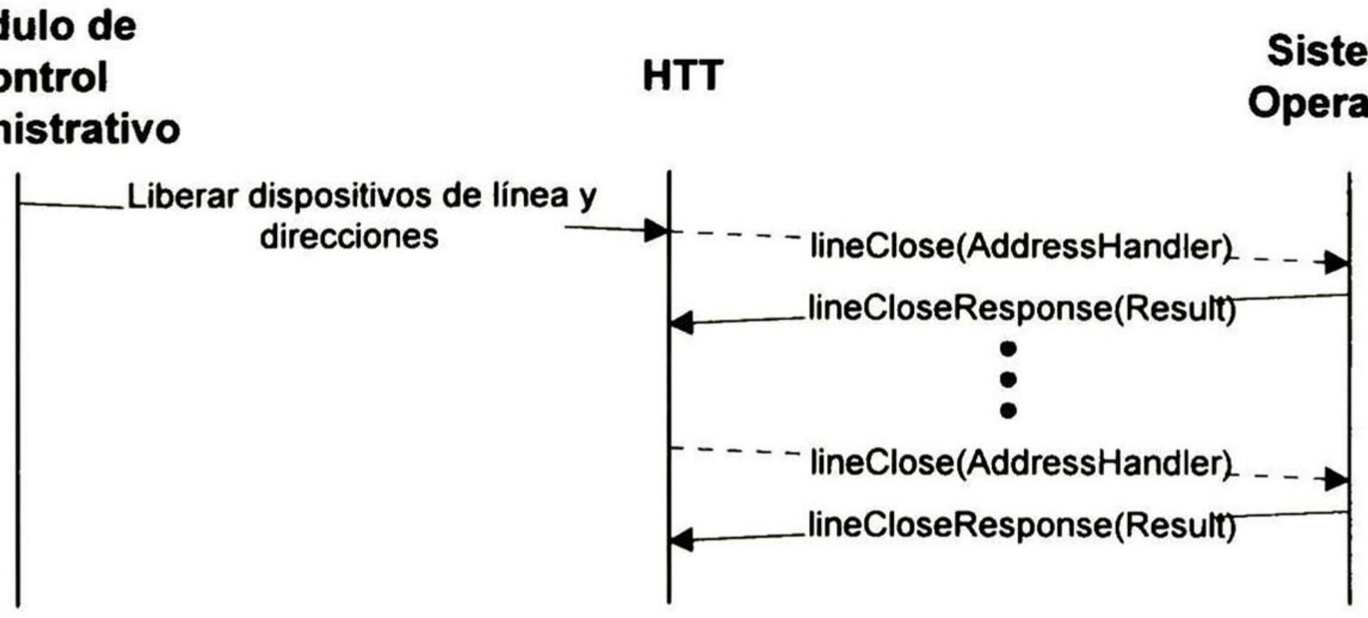


Figura 4-5. Secuencia estímulo/respuesta para la liberación de los dispositivos de línea y las direcciones

4.5.2.4.4 Seudocódigo

Liberación de los dispositivos de línea y direcciones( ):
Por cada dirección asociada con cada dispositivo de línea realizar las siguientes acciones:
Si el handler de la dirección es válido **entonces**
  invocar la función TAPI lineClose
  eliminar la información almacenada que está asociada con la dirección
Por cada dispositivo de línea realizar las siguientes acciones:
  eliminar la información almacenada que está asociada con el dispositivo de línea

4.5.2.5 Deshabilitación de los dispositivos de línea

4.5.2.5.1 Propósito

Esta facilidad libera el "handler" asociado con HTT que fue retornado durante la habilitación de los dispositivos de línea y de las direcciones.

4.5.2.5.2 Descripción

Esta facilidad debe ser invocada antes de terminar la aplicación que utiliza los dispositivos de línea y después de que han sido liberados los "handlers" asociados con los dispositivos de línea y con las direcciones.

4.5.2.5.3 Secuencia estímulo/respuesta

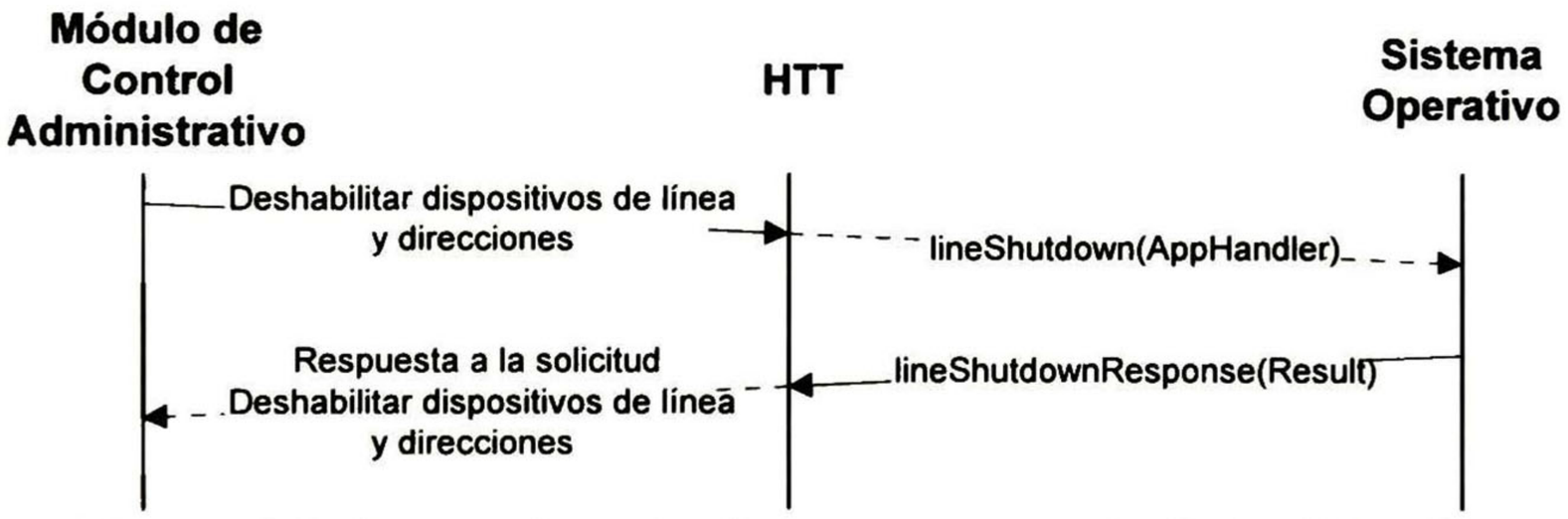


Figura 4-6. Secuencia estímulo/respuesta para la deshabilitación de los dispositivos de línea y de las direcciones

4.5.2.5.4 Seudocódigo

Deshabilitación de los dispositivos de línea y las direcciones( ):
Verificar que no se han deshabilitado los dispositivos de línea
Si ya han sido deshabilitados **entonces**
  enviar el mensaje ERR_TAPIALREADYSHUTDOWN al módulo CALT
en otro caso
  invocar la función TAPI lineShutdown
  Si el valor de retorno de lineShutdown es igual a SUCCESS **entonces**
    enviar el mensaje SUCCESS al módulo CALT

eliminar la información almacenada durante la habilitación de los dispositivos de línea
en otro caso
enviar el mensaje ERR_SHUTDOWNTAPIFAILED al módulo CALT

4.5.2.6 Realización de llamadas telefónicas

4.5.2.6.1 Propósito

Esta facilidad solicita a las DLLs de telefonía la marcación de un número telefónico para realizar una llamada. HTT recibe de las DLLs de telefonía un "handler" asociado con la llamada telefónica.

4.5.2.6.2 Descripción

La función TAPI para realizar llamadas telefónicas es asíncrona. El "handler" de llamada telefónica que retornan las DLLs de telefonía cuando se solicita esta facilidad es válido hasta que se ha recibido la notificación de consumación de dicha función asíncrona. Posteriormente se utilizará este "handler" para llevar a cabo cualquier operación sobre la llamada telefónica correspondiente.

La invocación de cualquier función TAPI asíncrona retorna un identificador de respuesta asíncrona pendiente cuando no existe error alguno en los parámetros de la función. HTT deberá guardar un registro con la siguiente información siempre que se le solicite esta facilidad: el identificador de respuesta asíncrona pendiente, el identificador de la llamada telefónica proporcionado por el módulo CALT y el nombre de la función asíncrona invocada.

4.5.2.6.3 Secuencia estímulo/respuesta

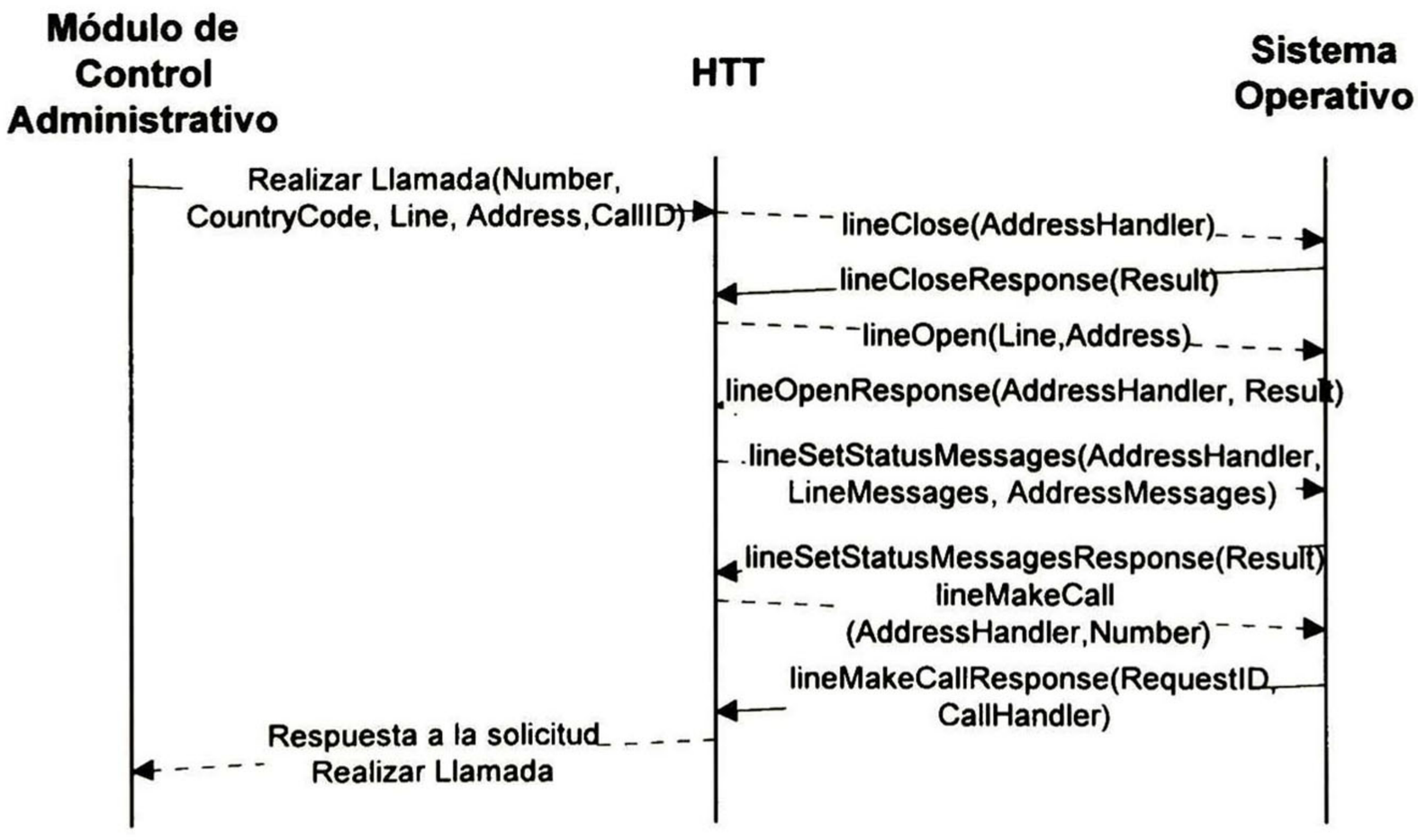


Figura 4-7. Secuencia estímulo/respuesta para la realización de una llamada telefónica

4.5.2.6.4 Seudocódigo

```
Realización de Llamada Telefónica(destNumber: cadena,
                                  countryCode: cadena,
                                  lineID: identificadorDispositivoLínea,
                                  addressID: identificadorDirección,
                                  callID: identificadorLlamada):
Si no existe una llamada telefónica con identificador igual a callID entonces
  Si existe un dispositivo de línea con identificador igual a lineID y
    su estado es diferente de LINEDEVSTATE_OUTOFSERVICE y
    soporta el modo portador LINEBEARERMODE_VOICE y
    soporta el tipo de medio LINEMEDIAMODE_INTERACTIVEVOICE y
    soporta la capacidad LINEFEATURE_MAKECALL entonces
   Si existe una dirección con identificador igual a addressID asociada con el dispositivo
      de línea con identificador igual a lineID y el número de llamadas activas en la
      dirección es menor al número máximo de llamadas activas de la dirección y soporta
      la capacidad LINEFEATURE_MAKECALL entonces
     Si la dirección está configurada para notificar de la llegada de llamadas telefónicas entrantes entonces
       invocar la función TAPI lineClose
       Si el valor de retorno de lineClose es igual a SUCCESS entonces
         invocar la función TAPI lineOpen
       en otro caso
         enviar el mensaje ERR_CLOSELINEFAILED al módulo CALT
     en otro caso
       invocar la función TAPI lineOpen
  Si el valor de retorno de lineOpen es igual a SUCCESS entonces
    almacenar el handler válido asociado con la dirección con identificador igual a address devuelto
    por lineOpen
    invocar la función TAPI lineSetStatusMessages
    Si el valor de retorno de lineSetStatusMessages es igual a SUCCESS entonces
      invocar la función TAPI lineMakeCall
      Si el valor de retorno de lineMakeCall es un número positivo entonces
        incrementar el número de llamadas activas en la dirección en 1
        asignar a la nueva llamada telefónica el identificador callID
        asignar a la nueva llamada telefónica la dirección destino destNumber
        asignar a la nueva llamada telefónica el código de país countryCode
        asignar a la nueva llamada telefónica la dirección dada por addressID
        asignar a la nueva llamada telefónica el dispositivo de línea dado por lineID
        asignar a la nueva llamada telefónica el handler devuelto por la función lineMakeCall
        almacenar la información de la respuesta pendiente asociada con la invocación de lineMakeCall
        que incluye: el valor de retorno de lineMakeCall como identificador de la respuesta pendiente,
        el identificador de la llamada telefónica dado por callID y el nombre de la función a la cual está
        asociada la respuesta pendiente, esto es, lineMakeCall
        enviar el mensaje SUCCESS al módulo CALT
      en otro caso
        enviar el mensaje ERR_CONNECTIONFAILED al módulo CALT
    en otro caso
      enviar el mensaje ERR_SETMESSAGESFAILED al módulo CALT
en otro caso
  enviar el mensaje ERR_OPENLINEFAILED al módulo CALT
```

```
    en otro caso
      enviar el mensaje ERR_ADDRESSNOTFOUND al módulo CALT
  en otro caso
    enviar el mensaje ERR_LINENOTFOUND al módulo CALT
en otro caso
  enviar el mensaje ERR_CALLIDALREADYREGISTERED al módulo CALT
```

4.5.2.7 Marcación de dígitos sobre una llamada telefónica

4.5.2.7.1 Propósito

Esta facilidad solicita a las DLLs de telefonía la marcación de dígitos sobre una llamada telefónica.

4.5.2.7.2 Descripción

La función TAPI para realizar la marcación de dígitos sobre una llamada telefónica es asíncrona. HTT deberá guardar un registro con la siguiente información siempre que se le solicite esta facilidad: el identificador de respuesta asíncrona pendiente, el identificador de la llamada telefónica proporcionado por el módulo CALT y el nombre de la función asíncrona invocada.

4.5.2.7.3 Secuencia estímulo/respuesta

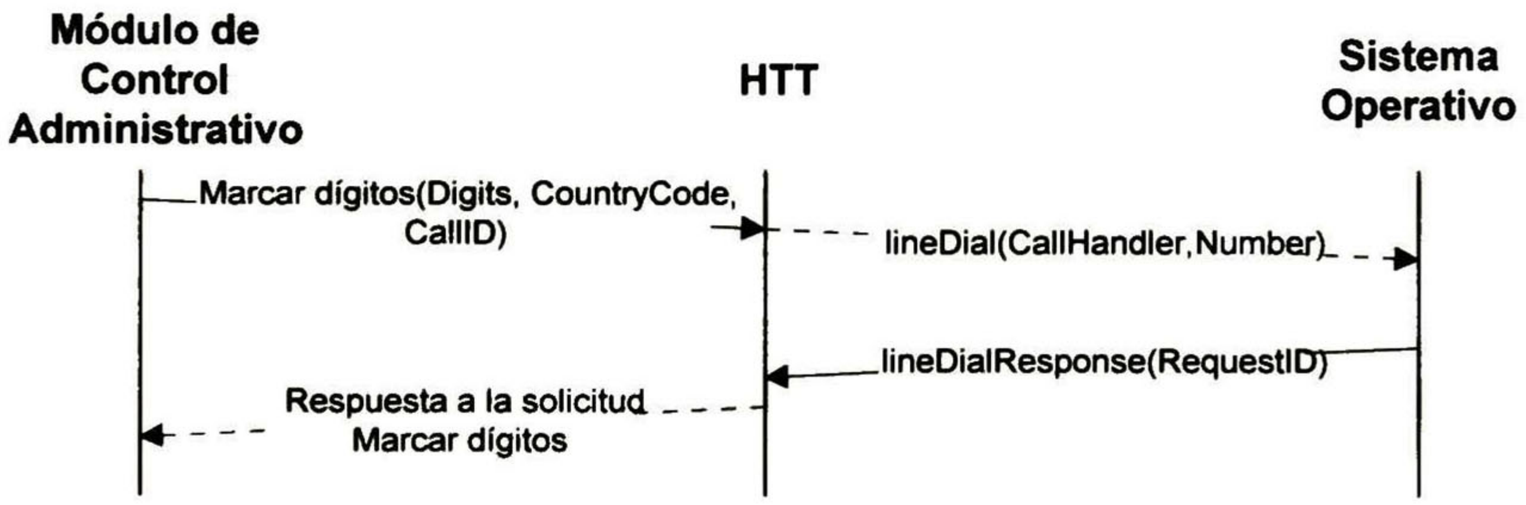


Figura 4-8. Secuencia estímulo/respuesta para la marcación de dígitos sobre una llamada telefónica

4.5.2.7.4 Seudocódigo

```
Marcación de dígitos(digits: cadena,
                     countryCode: cadena,
                     callID: identificadorLlamada):
Si existe una llamada telefónica con identificador igual a callID entonces
  Si el estado del dispositivo de línea al cual está asociada la dirección por la que se realiza
    la llamada telefónica con identificador igual a callID es diferente de
    LINEDEVSTATE_OUTOFSERVICE entonces
    Si no hay respuesta pendiente para la función TAPI asíncrona lineMakeCall entonces
```

```
    invocar la función TAPI lineDial
    Si el valor de retorno de lineDial es igual a SUCCESS entonces
      almacenar la información de la respuesta pendiente asociada con la invocación de lineDial
      que incluye: el valor de retorno de lineDial como identificador de la respuesta pendiente,
      el identificador de la llamada telefónica dado por callID y el nombre de la función a la cual está
      asociada la respuesta pendiente, esto es, lineDial
      enviar el mensaje SUCCESS al módulo CALT
    en otro caso
      enviar el mensaje ERR_DIALINGFAILED al módulo CALT
  en otro caso
    enviar el mensaje ERR_PENDANTREPLIES al módulo CALT
  en otro caso
    enviar el mensaje ERR_LINENOTFOUND al módulo CALT
en otro caso
  enviar el mensaje ERR_CALLNOTFOUND al módulo CALT
```

4.5.2.8 Aceptación y contestación de llamadas telefónicas basadas en el callerID

4.5.2.8.1 Propósito

Esta facilidad solicita a las DLLs de telefonía la aceptación y contestación de una llamada telefónica entrante.

4.5.2.8.2 Descripción

Las funciones TAPI para aceptar y contestar una llamada telefónica son asíncronas. Por cada invocación de una función asíncrona, HTT deberá guardar un registro con la siguiente información siempre que se le solicite esta facilidad: el identificador de respuesta asíncrona pendiente, el identificador de la llamada telefónica proporcionado por el módulo CALT y el nombre de la función asíncrona invocada. Esta facilidad es invocada después de que HTT ha recibido una notificación de llamada entrante proveniente de las DLLs de telefonía y de que el módulo CALT ha validado el número de callerID.

4.5.2.8.3 Secuencia estímulo/respuesta

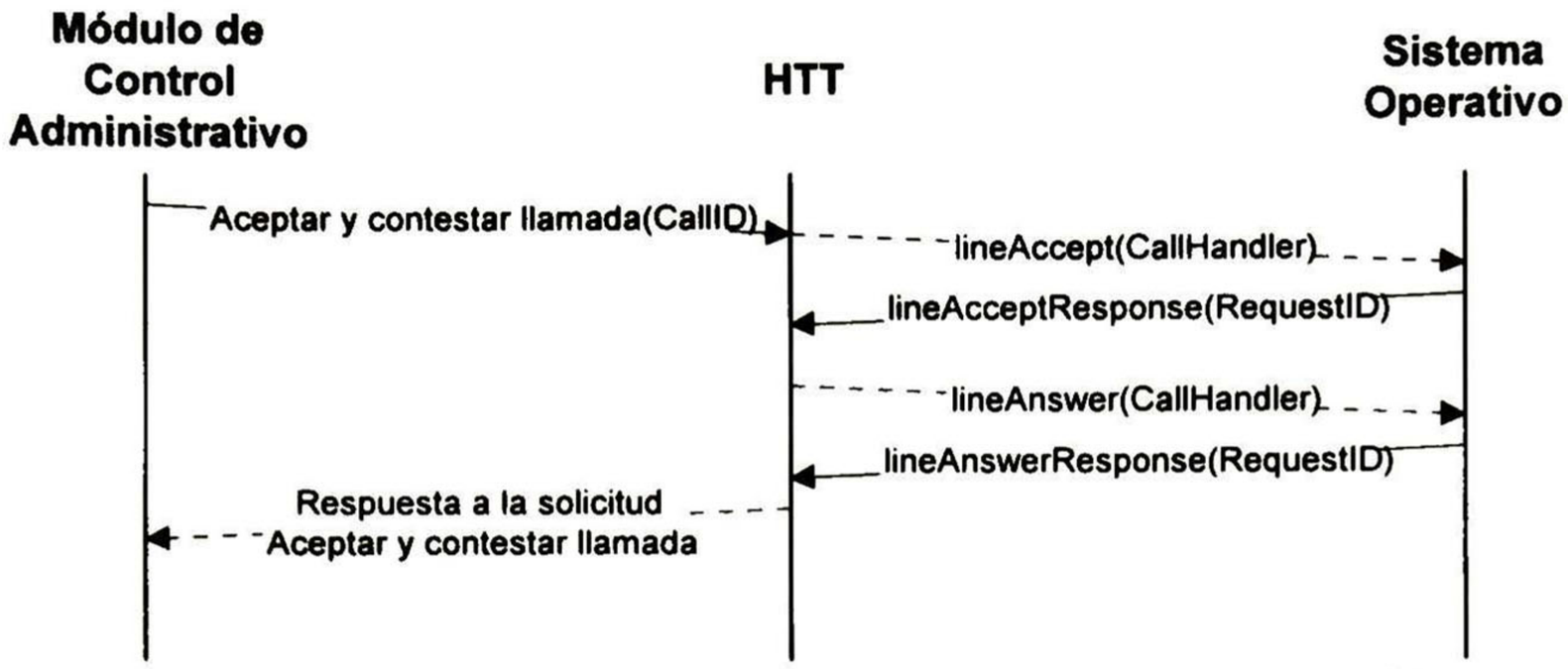


Figura 4-9. Secuencia estímulo/respuesta para la aceptación y contestación de una llamada telefónica entrante

4.5.2.8.4 Seudocódigo

Aceptación y contestación de llamada telefónica(callID: identificadorLlamada):
Si existe una llamada telefónica con identificador igual a callID **entonces**
  invocar la función TAPI lineAccept
  Si el valor de retorno de lineAccept es un número positivo **entonces**
    almacenar la información de la respuesta pendiente asociada con la invocación de lineAccept
    que incluye: el valor de retorno de lineAccept como identificador de la respuesta pendiente,
    el identificador de la llamada telefónica dado por callID y el nombre de la función a la cual está
    asociada la respuesta pendiente, esto es, lineAccept
    invocar la función TAPI lineAnswer
    Si el valor de retorno de lineAnswer es un número positivo **entonces**
      almacenar la información de la respuesta pendiente asociada con la invocación de lineAnswer
      que incluye: el valor de retorno de lineAnswer como identificador de la respuesta pendiente,
      el identificador de la llamada telefónica dado por callID y el nombre de la función a la cual está
      asociada la respuesta pendiente, esto es, lineAnswer
      enviar el mensaje SUCCESS al módulo CALT
    en otro caso
      enviar el mensaje ERR_CONNECTIONFAILED al módulo CALT
  en otro caso
    enviar el mensaje ERR_CONNECTIONFAILED al módulo CALT
en otro caso
  enviar el mensaje ERR_CALLNOTFOUND al módulo CALT

4.5.2.9 Terminación de llamada telefónica desde software

4.5.2.9.1 Propósito

Esta facilidad solicita a las DLLs de telefonía la terminación de una llamada telefónica desde software.

4.5.2.9.2 Descripción

La función TAPI para terminar una llamada telefónica es asíncrona. HTT deberá guardar un registro con la siguiente información siempre que se le solicite esta facilidad: el identificador de respuesta asíncrona pendiente, el identificador de la llamada telefónica proporcionado por el módulo CALT y el nombre de la función asíncrona invocada.

4.5.2.9.3 Secuencia estímulo/respuesta

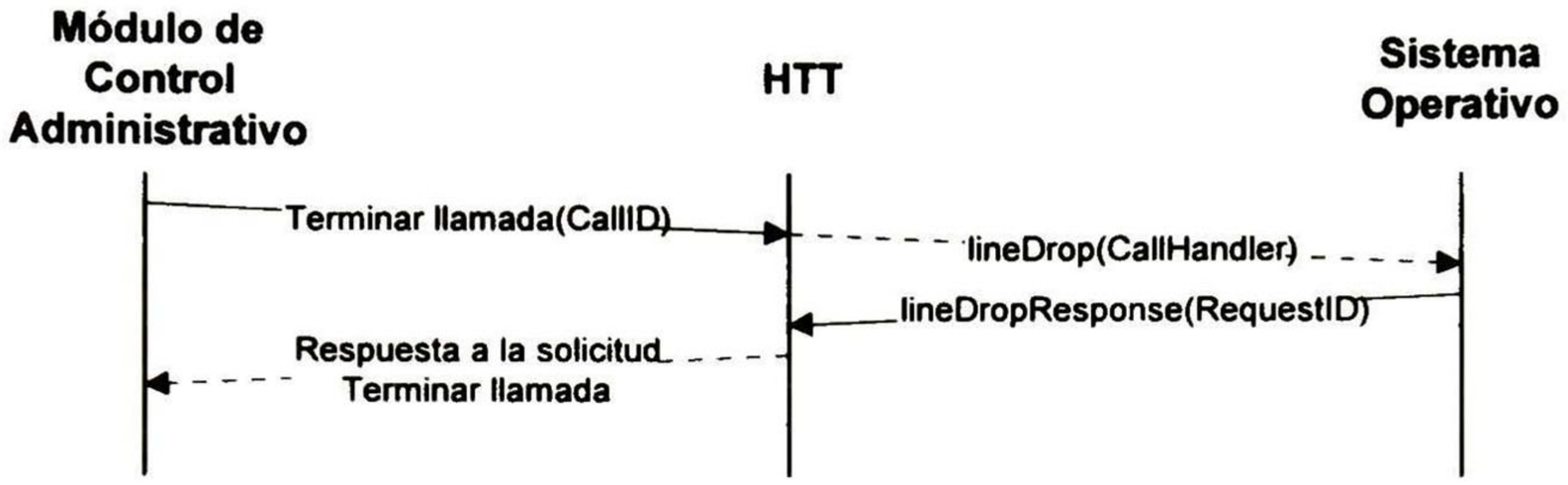


Figura 4-10. Secuencia estímulo/respuesta para la terminación de una llamada telefónica desde software

4.5.2.9.4 Seudocódigo

Terminación de llamada telefónica(callID: identificadorLlamada):
Si existe una llamada telefónica con identificador igual a callID **entonces**
  Si no se ha solicitado previamente la terminación de la llamada telefónica ni se ha recibido
    la notificación de terminación de la llamada desde el aparato telefónica **entonces**
    invocar la función TAPI lineDrop
    Si el valor de retorno de lineDrop es un número positivo **entonces**
      almacenar la información de la respuesta pendiente asociada con la invocación de lineDrop
      que incluye: el valor de retorno de lineDrop como identificador de la respuesta pendiente,
      el identificador de la llamada telefónica dado por callID y el nombre de la función a la cual está
      asociada la respuesta pendiente, esto es, lineDrop
      enviar el mensaje SUCCESS al módulo CALT
    en otro caso
      enviar el mensaje ERR_DROPCALLFAILED al módulo CALT
  en otro caso
    enviar el mensaje ERR_ALREADYDISCONNECTINGCALL al módulo CALT
en otro caso
  enviar el mensaje ERR_CALLNOTFOUND al módulo CALT

4.5.2.10 Procesamiento de las notificaciones de consumación de funciones asíncronas

4.5.2.10.1 Propósito

Esta facilidad procesa las notificaciones de consumación de las funciones TAPI asíncronas invocadas. Estas notificaciones son enviadas a HTT por las DLLs de telefonía.

4.5.2.10.2 Descripción

Las funciones asíncronas ejecutan acciones sobre llamadas telefónicas, tales como crearlas, contestarlas y colgarlas. Cuando las DLLs de telefonía notifican a HTT la respuesta asíncrona pendiente, éste busca en sus registros el que coincida con el identificador de la respuesta pendiente y lo asocia con la llamada telefónica correspondiente para realizar las acciones necesarias. El envío de mensajes al módulo CALT cuando ocurre una notificación de este tipo es opcional (ver Figura 4-11) porque depende de la función asíncrona que completó su ejecución.

Cuando el módulo CALT solicite alguna operación sobre una llamada telefónica, HTT deberá revisar que la llamada no tenga asociada una respuesta asíncrona pendiente originada por la función TAPI para realizar llamadas, *lineMakeCall*.

4.5.2.10.3 Secuencia estímulo/respuesta

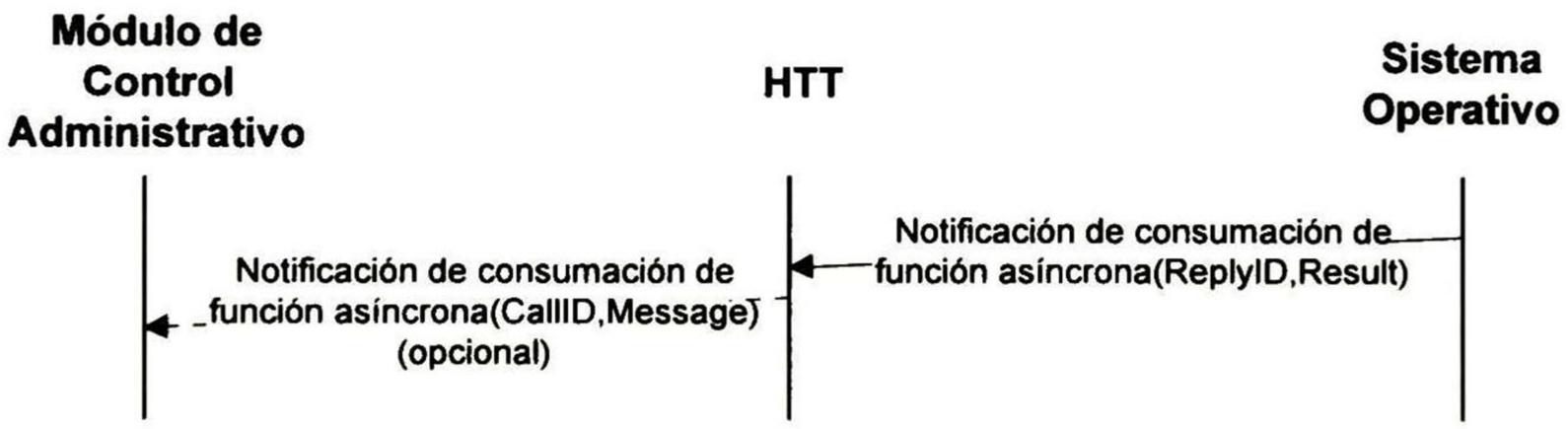


Figura 4-11. Secuencia estímulo/respuesta para la notificación de consumación de una función asíncrona

4.5.2.10.4 Seudocódigo

```
Procesamiento de respuesta asíncrona(device: TAPIdevice,
                                     message: mensaje,
                                     callbackInst: callback,
                                     param1: parámetro,
                                     param2: parámetro,
                                     param3: parámetro ):
Si existe una respuesta pendiente con identificador igual a param1 entonces
  Si existe una llamada telefónica que tiene asociada una respuesta pendiente con identificador
     igual a param1 entonces
    identificar la función TAPI asíncrona a la que está asociada la respuesta pendiente
    Si la respuesta pendiente está asociada a la función lineMakeCall entonces
```

```
Si param2 es igual a SUCCESS entonces
  enviar el mensaje CallStarted al módulo CALT con el identificador de la llamada telefónica
  asociada con la respuesta pendiente
en otro caso
  invocar la función TAPI lineClose
  Si el valor de retorno de lineClose es igual a SUCCESS entonces
    configurar nuevamente la dirección para habilitar la notificación de llamadas entrantes
  enviar el mensaje FinishCall al módulo CALT indicándole el identificador de la llamada
  telefónica asociada con la respuesta pendiente
  eliminar la información relacionada con la llamada telefónica asociada con la respuesta pendiente
eliminar el registro de la información asociada con la respuesta pendiente
en otro caso
Si la respuesta pendiente está asociada a las funciones lineDial, lineAccept o lineAnswer entonces
  Si param2 es igual a SUCCESS entonces
    eliminar el registro de la información asociada con la respuesta pendiente
  en otro caso
    invocar la función TAPI lineDeallocateCall
    invocar la función TAPI lineClose
    Si el valor de retorno de lineClose es igual a SUCCESS entonces
      configurar nuevamente la dirección para habilitar la notificación de llamadas entrantes
    enviar el mensaje FinishCall al módulo CALT indicándole el identificador de la llamada
    telefónica asociada con la respuesta pendiente
    eliminar la información relacionada con la llamada telefónica asociada con la respuesta pendiente
    eliminar el registro de la información asociada con la respuesta pendiente
en otro caso
  Si la respuesta pendiente está asociada a la función lineDrop entonces
    Si param2 es igual a SUCCESS entonces
      eliminar el registro de la información asociada con la respuesta pendiente
    en otro caso
      enviar el mensaje ERR_DROPCALLFAILED al módulo CALT
      eliminar el registro de la información asociada con la respuesta pendiente
```

4.5.2.11 Procesamiento de las notificaciones de cambio de estado de las llamadas telefónicas

4.5.2.11.1 Propósito

Esta facilidad procesa las notificaciones de cambio de estado de las llamadas telefónicas, provenientes de las DLLs de telefonía.

4.5.2.11.2 Descripción

El módulo CALT recibe notificaciones de la ocurrencia de algunos estados para efectos de tarificación (ver Figura 4-12 y sección 4.5.2.11.3). Los estados de una llamada telefónica definidos por TAPI (ver Tabla 4-3) que son signiticativos para HTT son:

- Ocioso
- Ocupado

- Conectado
- Desconectado

Estado	Descripción
Ocioso	Una llamada telefónica evoluciona al estado Ocioso cuando ha dejado de existir (aplicable a llamadas entrantes y salientes).
Ocupado	Una llamada telefónica evoluciona al estado Ocupado cuando no ha podido completarse debido a que algún recurso entre el abonado solicitante y el solicitado no está disponible, por ejemplo cuando algún switch intermedio ha alcanzado su capacidad y no puede manejar una llamada más (aplicable sólo a llamadas salientes). Su información adicional incluye: • Estación ocupada Significa que la estación del abonado solicitado está ocupada (el teléfono está descolgado). • Troncal ocupada Significa que un circuito en el switch o en la red está ocupado.
Conectado	Una llamada telefónica evoluciona al estado Conectado cuando se ha establecido la comunicación entre ambos extremos (el abonado solicitante y el solicitado) y puede comenzar a fluir la información (voz o datos) (aplicable a llamadas entrantes y salientes).
Desconectado	Una llamada telefónica evoluciona al estado Desconectado cuando el abonado remoto ha finalizado la comunicación (aplicable a llamadas entrantes y salientes). Su información adicional incluye: • LINEDISCONNECTMODE_NORMAL Una solicitud de desconexión normal ha sido realizada por el abonado remoto; la llamada es terminada normalmente. • LINEDISCONNECTMODE_BUSY La estación del abonado remoto está ocupada. • LINEDISCONNECTMODE_NOANSWER La estación del abonado remoto no responde. • LINEDISCONNECTMODE_BADADDRESS La dirección del abonado solicitado es inválida.

Tabla 4-3. Estados de las llamadas telefónicas definidos por TAPI que son procesados por HTT

4.5.2.11.3 Secuencia estímulo/respuesta

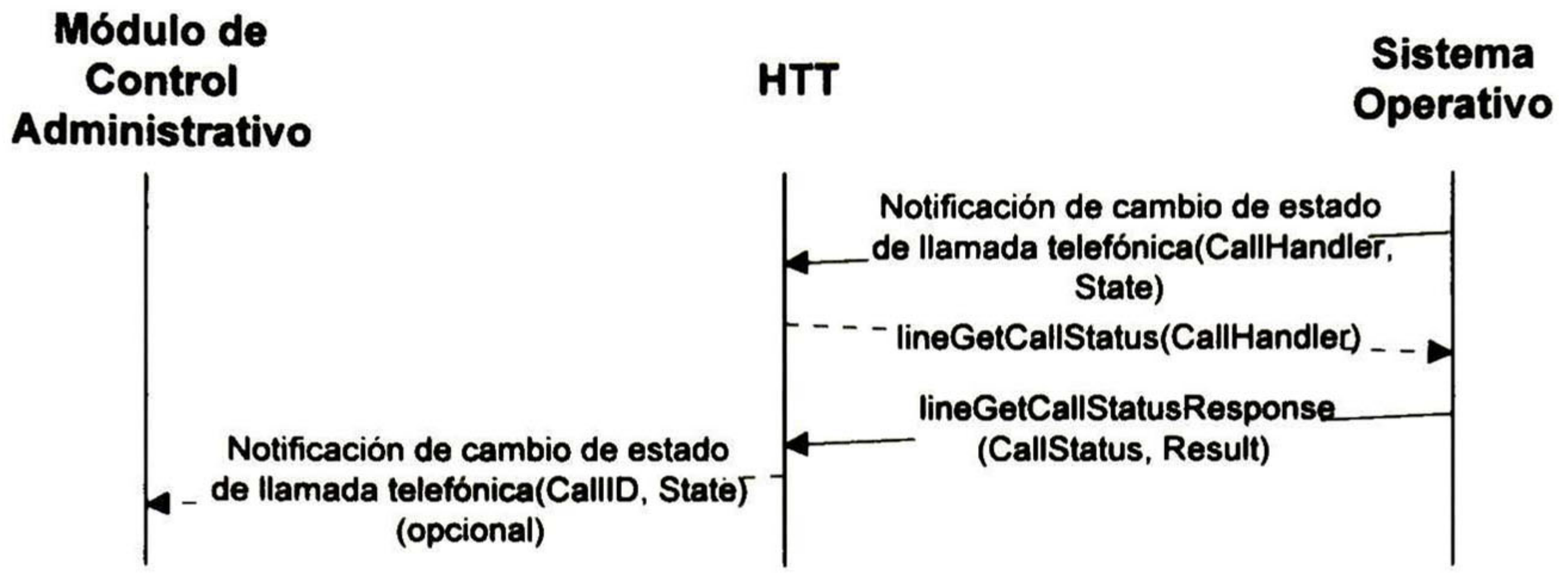


Figura 4-12. Secuencia estímulo/respuesta para la notificación de cambio de estado de una llamada telefónica

4.5.2.11.4 Seudocódigo

Procesamiento de cambio de estado de llamada telefónica(device: TAPIdevice,
message: mensaje,
callbackInst: callback,
param1: parámetro,
param2: parámetro,
param3: parámetro):

Si existe una llamada telefónica con handler igual a device **entonces**
designar param1 como el nuevo estado de la llamada telefónica con handler igual a device
invocar la función TAPI lineGetCallStatus
identificar el nuevo estado de la llamada telefónica
Si param1 es igual a LINECALLSTATE_CONNECTED **entonces**
enviar el mensaje CallConnected al módulo CALT indicando el identificador de la llamada telefónica que cambió de estado
en otro caso
Si param1 es igual a LINECALLSTATE_BUSY **entonces**
Si no se ha enviado el mensaje FinishCall al módulo CALT **entonces**
enviar el mensaje FinishCall al módulo CALT indicándole el identificador de la llamada telefónica que cambió de estado
Si no ha iniciado el proceso de terminación de la llamada **entonces**
invocar la función TAPI lineClose
Si el valor de retorno de lineClose es igual a SUCCESS **entonces**
configurar nuevamente la dirección para habilitar la notificación de llamadas entrantes
eliminar la información asociada con la llamada telefónica que cambió de estado
en otro caso
Si param1 es igual a LINECALLSTATE_DISCONNECTED **entonces**
Si no se ha enviado el mensaje FinishCall al módulo CALT **entonces**
enviar el mensaje FinishCall al módulo CALT indicándole el identificador de la llamada telefónica que cambió de estado
Si no ha iniciado el proceso de terminación de la llamada **entonces**
invocar la función TAPI lineDrop
en otro caso
Si param1 es igual a LINECALLSTATE_IDLE **entonces**

```
Si no se ha enviado el mensaje FinishCall al módulo CALT entonces
  enviar el mensaje FinishCall al módulo CALT indicándole el identificador de la llamada
invocar la función TAPI lineDeallocateCall
invocar la función TAPI lineClose
Si el valor de retorno de lineClose es igual a SUCCESS entonces
  configurar nuevamente la dirección para habilitar la notificación de llamadas entrantes
eliminar la información asociada con la llamada telefónica que cambió de estado
```

4.5.2.12 Procesamiento de las notificaciones de cambio de estado de los dispositivos de línea

4.5.2.12.1 Propósito

Esta facilidad procesa las notificaciones de cambio de estado de los dispositivos de línea. Estas notificaciones son enviadas a HTT por las DLLs de telefonía.

4.5.2.12.2 Descripción

Los estados de los dispositivos de línea definidos por TAPI (ver Tabla 4-4) que procesa HTT son:

- LINEDEVSTATE_INSERVICE
- LINEDEVSTATE_OUTOFSERVICE

Nombre del Estado	Descripción
LINEDEVSTATE_INSERVICE	El dispositivo de línea está conectado a TAPI. Esto sucede cuando TAPI es primero activado o cuando el cable del dispositivo de línea está físicamente conectado y en servicio mientras TAPI está activo.
LINEDEVSTATE_OUTOFSERVICE	El dispositivo de línea está fuera de servicio en el switch o está físicamente desconectado. TAPI no puede ser usado para operar sobre el dispositivo de línea.

Tabla 4-4. Estados de los dispositivos de línea que son procesados por HTT

4.5.2.12.3 Secuencia estímulo/respuesta

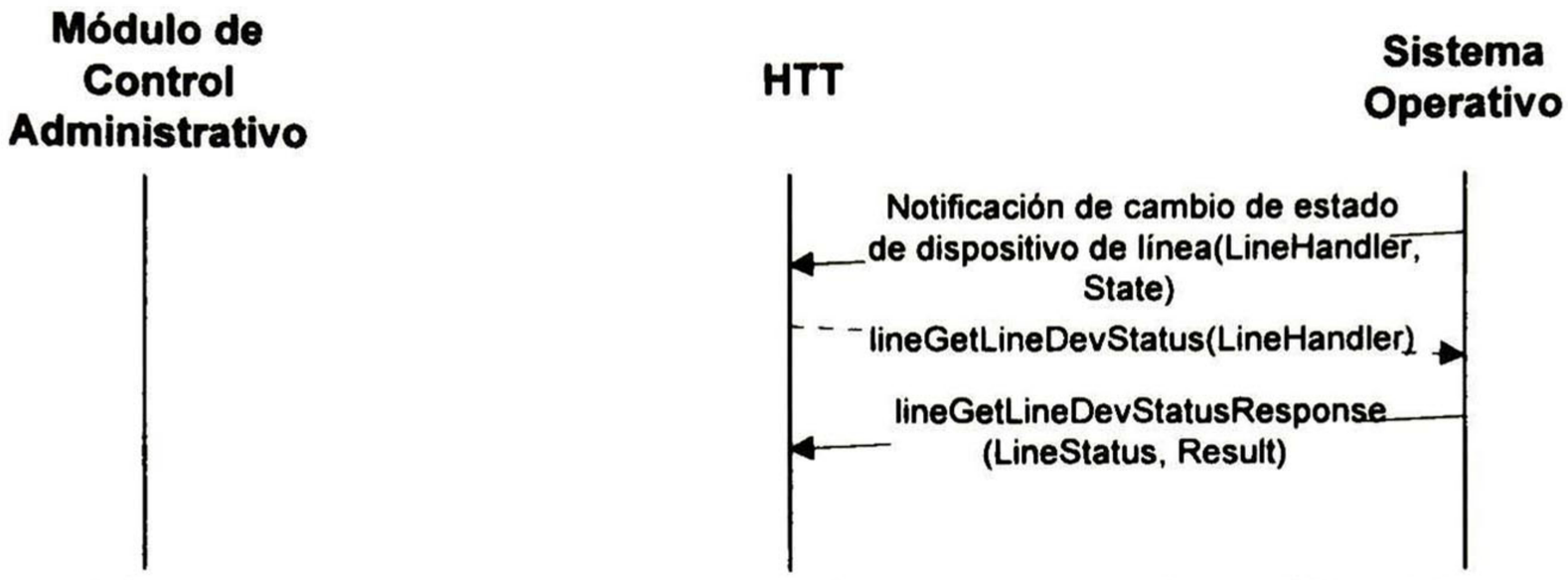


Figura 4-13. Secuencia estímulo/respuesta para la notificación de cambio de estado de un dispositivo de línea

4.5.2.12.4 Seudocódigo

Procesamiento de cambio de estado de dispositivo de línea(device: TAPIdevice,
message: mensaje,
callbackInst: callback,
param1: parámetro,
param2: parámetro,
param3: parámetro):

```
Si existe un dispositivo de línea con handler igual a device entonces
  designar param1 como el nuevo estado del dispositivo de línea con handler igual a device
  invocar la función TAPI lineGetLineDevStatus
  Si param1 es igual a LINEDEVSTATE_INSERVICE entonces
    configurar las direcciones asociadas con el dispositivo de línea con handler igual a param1
    para habilitar la notificación de llamadas telefónicas entrantes
  en otro caso
    Si param1 es igual a LINEDEVSTATE_OUTOFSERVICE entonces
      para cada dirección asociada con el dispositivo de línea con handler igual a param1 realizar
      las siguientes acciones:
      invocar la función TAPI lineClose
```

4.5.2.13 Procesamiento de las notificaciones de cambio de estado de las direcciones

4.5.2.13.1 Propósito

Esta facilidad procesa las notificaciones de cambio de estado de las direcciones. HTT recibe estas notificaciones desde las DLLs de telefonía.

4.5.2.13.2 Descripción

Los estados de las direcciones definidos por TAPI (ver Tabla 4-5) que procesa HTT son:

- LINEADDRESSSTATE_INUSEZERO
- LINEADDRESSSTATE_INUSEONE

Nombre del Estado	Descripción
LINEADDRESSSTATE_INUSEZERO	Una dirección ha cambiado a "idle" (está en uso por cero estaciones).
LINEADDRESSSTATE_INUSEONE	Una dirección ha cambiado de estar en "idle" o de estar en uso por varias estaciones a estar en uso por una sola estación.

Tabla 4-5. Estados de las direcciones que son procesados por HTT

4.5.2.13.3 Secuencia estímulo/respuesta

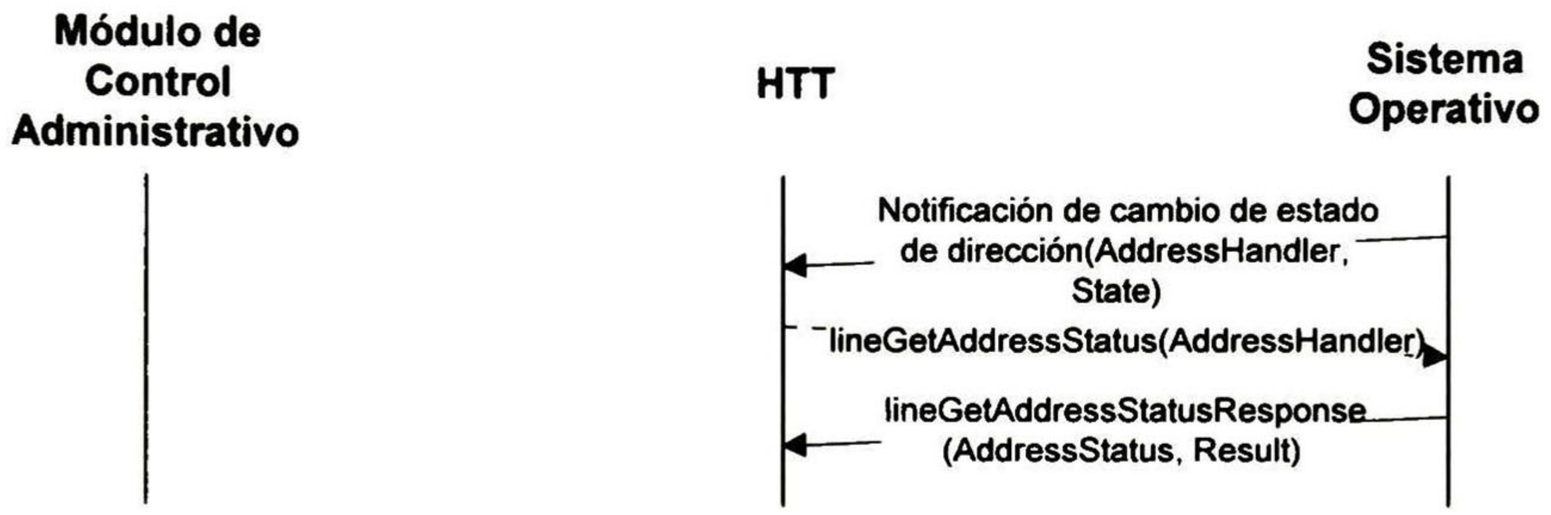


Figura 4-14. Secuencia estímulo/respuesta para la notificación de cambio de estado de una dirección

4.5.2.13.4 Seudocódigo

Procesamiento de cambio de estado de dirección(device: TAPIdevice,
message: mensaje,
callbackInst: callback,
param1: parámetro,
param2: parámetro,
param3: parámetro):
Si existe una dirección con handler igual a device **entonces**
designar param2 como el nuevo estado de la dirección con handler igual a device
invocar la función TAPI lineGetAddressStatus

4.5.2.14 Procesamiento de las notificaciones de llamadas telefónicas realizadas desde el aparato telefónico

4.5.2.14.1 Propósito

Esta facilidad procesa las notificaciones de nuevas llamadas que se realizan desde el aparato telefónico. Las notificaciones de llamadas realizadas desde el teléfono son remitidas a HTT por las DLLs de telefonía.

4.5.2.14.2 Descripción

Cuando se recibe una notificación de llamada saliente realizada desde el aparato telefónico se envía un mensaje al módulo CALT para que éste valide el número telefónico marcado.

4.5.2.14.3 Secuencia estímulo/respuesta

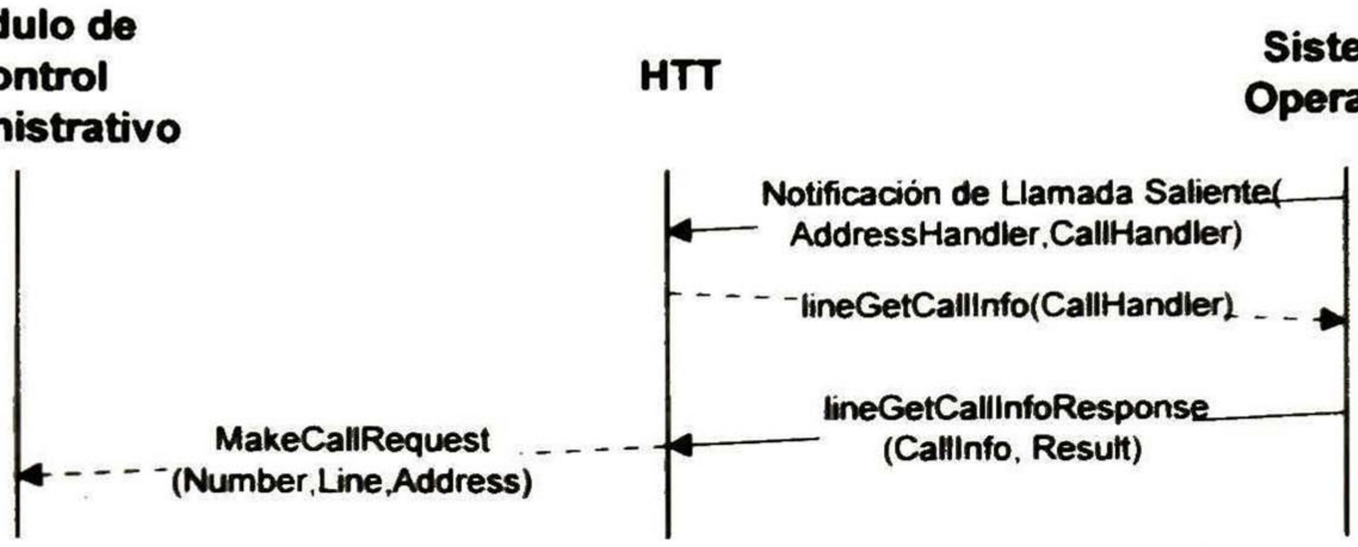


Figura 4-15. Secuencia estímulo/respuesta para la notificación de llamada saliente realizada desde el aparato telefónico

4.5.2.14.4 Seudocódigo

Procesamiento de notificación de llamada saliente desde teléfono(device: TAPIdevice,
message: mensaje,
callbackInst: callback,
param1: parámetro,
param2: parámetro,
param3: parámetro):

Si existe una dirección con handler igual a device **entonces**
  invocar la función TAPI lineGetCallInfo
  Si el valor de retorno de lineGetCallInfo es igual a SUCCESS **entonces**
    designar device como handler de la dirección por la cual se realiza la nueva llamada telefónica
    designar param1 como identificador de la dirección por la cual se realiza la nueva llamada telefónica
    designar param2 como handler de la nueva llamada telefónica
    enviar el mensaje MakeCallRequest al módulo CALT

4.5.2.15 Procesamiento de las notificaciones de llamadas telefónicas entrantes

4.5.2.15.1 Propósito

Esta facilidad procesa las notificaciones de nuevas llamadas entrantes que las DLLs de telefonía envían a HTT.

4.5.2.15.2 Descripción

Cuando se recibe una notificación de llamada entrante se envía un mensaje al módulo CALT indicándole el número telefónico de la persona que está llamando (callerID).

4.5.2.15.3 Secuencia estímulo/respuesta

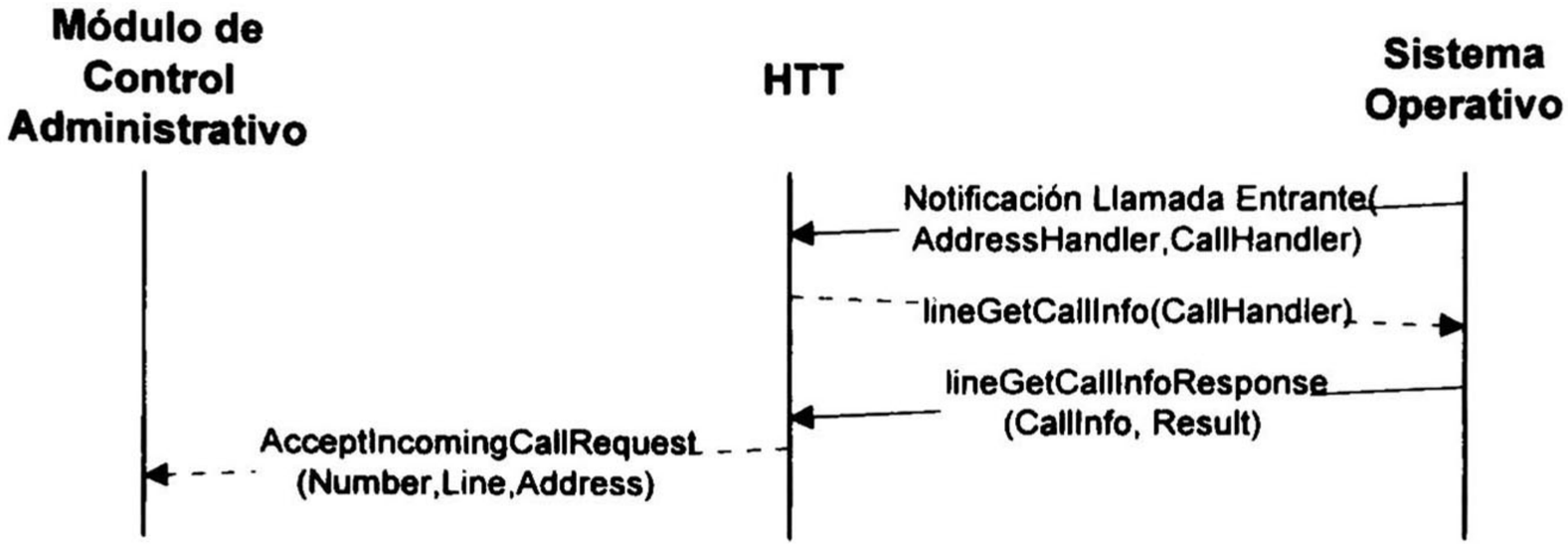


Figura 4-16. Secuencia estímulo/respuesta para la notificación de llamada entrante

4.5.2.15.4 Seudocódigo

Procesamiento de notificación de llamada entrante(device: TAPIdevice,
message: mensaje,
callbackInst: callback,
param1: parámetro,
param2: parámetro,
param3: parámetro):

Si existe una dirección con handler igual a device **entonces**
  invocar la función TAPI lineGetCallInfo
  Si el valor de retorno de lineGetCallInfo es igual a SUCCESS **entonces**
    designar device como handler de la dirección por la cual se realiza la nueva llamada telefónica
    designar param1 como identificador de la dirección por la cual se realiza la nueva llamada telefónica
    designar param2 como handler de la nueva llamada telefónica
    enviar el mensaje AcceptIncomingCallRequest al módulo CALT

5 Desarrollo en Fusion del Habilitador Telefónico basado en TAPI (HTT)

5.1 Introducción

El capítulo 4 describe los requerimientos funcionales que HTT debe satisfacer. En este capítulo se presenta el proceso de transformación de este grupo de requerimientos en una arquitectura de objetos mediante la aplicación del método Fusion para desarrollo de software orientado a objetos[COL94].

5.2 Desarrollo de software orientado a objetos

Es bien reconocido que el desarrollo de sistemas de software a gran escala y la coordinación de las distintas tareas implicadas en este desarrollo es una labor compleja. Esta complejidad es la razón principal que propicia el retraso, exceso de presupuesto, baja calidad y otros problemas frecuentemente asociados con el software. La ingeniería de software tiene como propósito administrar esta complejidad mediante el empleo de herramientas sofisticadas, técnicas de representación (de requerimientos de usuario, arquitecturas y diseños), lenguajes de programación así como enfoques, métodos y prácticas seguidas durante el proceso de desarrollo de software[KRI99].

El enfoque más utilizado por la ingeniería de software es el uso de tecnologías orientadas a objetos (OOT). La tecnología orientada a objetos ofrece metodologías, técnicas y herramientas que proporcionan ventajas claras para desarrolladores de software y usuarios por igual. OOT permite el desarrollo de aplicaciones de manera rápida y rentable mediante el uso de componentes reusables, proveen conectividad a través de plataformas de computación heterogéneas y soporta la integración de datos, aplicaciones y procesos de negocios[BHA98].

Los sistemas orientados a objetos tienen muchas propiedades deseables. Dado que los objetos ocultan su representación interna, es posible cambiar la implementación de un objeto sin afectar la comunicación con sus clientes. Además, el agrupamiento de un conjunto de rutinas de acceso con los datos que manipulan permite a los diseñadores descomponer problemas en una colección de agentes que interactúan[SHA96].

Por otra parte, las técnicas de especificación formal (FSTs) pueden proporcionar la precisión y las herramientas necesarias para realizar un análisis riguroso de las propiedades modeladas. A pesar de su fortaleza, el uso de las FSTs disponibles actualmente tiene sus problemas. Frecuentemente, interpretar un modelo formal puede ser difícil dado el esfuerzo necesario para relacionar expresiones matemáticas con conceptos del mundo real. Además,

las FSTs no tienen un conjunto tan rico en estructuras de construcción como las técnicas estructuradas gráficamente. Lo anterior sugiere que las FSTs y las técnicas gráficas orientadas a objetos pueden jugar papeles complementarios en el desarrollo de software[FRA97].

El método orientado a objetos Fusion proporciona herramientas que auxilian a los desarrolladores en el manejo de la complejidad cuando se desarrollan grandes sistemas. Fusion cubre tres etapas del proceso de desarrollo de software: análisis, diseño e implementación. En las etapas de análisis y diseño define una serie de modelos que deben ser construidos así como la notación a utilizar. En la implementación establece una serie de consideraciones a tomar en cuenta para traducir el diseño al lenguaje de programación elegido.

Fusion pretende integrar los mejores aspectos de varios métodos. Sus principales influencias son OMT[RUM88], Booch[BOO91], CRC[BEC89] y los métodos formales. La fase de análisis se basa en gran parte en OMT. Fusion adoptó las especificaciones de precondiciones y postcondiciones de algunos métodos formales (como VDM[JON90]) para especificar declarativamente la funcionalidad del sistema. CRC y Booch son los fundamentos de la fase de diseño. Fusion incorpora verificaciones para consistencia y completud dado que hay un fundamento semántico subyacente para los tres modelos del análisis[COL94].

5.3 Desarrollo orientado a objetos del Habilitador Telefónico basado en TAPI

5.3.1 Análisis

El objetivo principal de la etapa de análisis en Fusion es describir qué hace el sistema. El análisis de HTT está constituido por los siguientes elementos: el modelo objeto del sistema, el modelo de operación y el modelo de ciclo de vida. Para construir el modelo objeto se identificaron los objetos participantes en el sistema y las relaciones entre ellos. El modelo de operación, en conjunto con los escenarios, refleja las interacciones entre el sistema y su ambiente. Por último, el modelo de ciclo de vida describe las secuencias de operaciones del sistema y eventos que son permitidas en el sistema.

En nuestro caso, el sistema está conformado por HTT y el papel del ambiente lo representan el sistema operativo (que incluye las DLLs de telefonía) y el Control Administrativo de Llamadas Telefónicas.

5.3.1.1 Modelo Objeto

El modelo objeto muestra las relaciones existentes entre los objetos del sistema y las relaciones de los objetos del sistema con los objetos del ambiente. Para identificar los objetos involucrados se analizó la especificación de requerimientos de HTT presentada en

el capítulo 4. Los objetos obtenidos (ver Tabla 5-1) fueron clasificados de la siguiente manera:

- Objetos que representan elementos conceptuales de TAPI: dispositivos de línea, direcciones y respuestas pendientes de funciones asíncronas.
- Objetos que representan a las llamadas telefónicas, de las cuales se consideran dos tipos: entrantes y salientes.
- Objetos contenedores o arreglos, que conforman los contenedores de dispositivos de línea, de llamadas telefónicas, de direcciones y de respuestas pendientes
- Objetos controladores, que coordinan a todos los anteriores para proveer la funcionalidad requerida.
- Objetos que modelan el medio ambiente, los cuales son: el sistema operativo (que contiene las DLLs de telefonía), el Control Administrativo de Llamadas Telefónicas y los mensajes que éste procesa.

Nombre del Objeto	Representa
Line	Dispositivo de línea
Address	Dirección
Reply	Respuesta pendiente de función asíncrona
Call	Llamada telefónica
IncomingCall	Llamada telefónica entrante
OutgoingCall	Llamada telefónica saliente
LineArray	Contenedor de dispositivos de línea
AddressArray	Contenedor de direcciones
CallArray	Contenedor de llamadas telefónicas
ReplyArray	Contenedor de respuestas pendientes
Tapi	Controlador de los objetos del sistema
Sistema Operativo	Sistema Operativo
Management Control	Control Administrativo de Llamadas Telefónicas
CallMessage	Mensaje procesado por el Ctrl. Admivo.

Tabla 5-1. Clasificación de objetos de HTT y de su ambiente

Una vez que se han identificado los objetos tanto del sistema como del ambiente, deben establecerse las relaciones entre ellos. La asociación física, la comunicación, la contención y las acciones son el origen de posibles relaciones [COL94]. La mayor parte de las relaciones están implícitas en la especificación de requerimientos del sistema. En nuestro caso, se obtuvieron a partir del análisis de la descripción de TAPI (Capítulo 3) y de la descripción de requerimientos de HTT (Capítulo 4). Las relaciones obtenidas a partir de la descripción de TAPI son las siguientes:

- Un dispositivo de línea tiene asociadas una o más direcciones
- La ejecución de una función asíncrona tiene asociado un identificador de respuesta pendiente

- Una llamada telefónica se realiza sobre una dirección
- Una notificación de consumación de función asíncrona está asociada con una respuesta pendiente

En cuanto a las relaciones derivadas de la descripción de requerimientos de HTT, podemos distinguir dos grupos:

- *Internas*

  Involucran exclusivamente a objetos pertenecientes al sistema. Comprenden las siguientes relaciones:
  - Los dispositivos de línea están contenidos en un arreglo
  - Cada dispositivo de línea tiene un arreglo de direcciones asociadas a él
  - Las llamadas telefónicas (tanto entrantes como salientes) que se están procesando en un momento determinado están contenidas en un arreglo
  - Las respuestas pendientes están contenidos en un arreglo
  - El objeto Tapi administra los arreglos de dispositivos de línea, direcciones, llamadas telefónicas y el de respuestas pendientes, auxiliándose de las notificaciones de eventos proporcionadas por el sistema operativo
  - El objeto Tapi asigna una dirección a una llamada telefónica realizada desde software
  - El objeto Tapi administra las solicitudes de operaciones sobre llamadas telefónicas: realizar llamadas, marcar dígitos sobre una llamada existente, aceptar una llamada entrante y terminar una llamada
  - Las llamadas telefónicas solicitan servicios al sistema operativo
  - Las notificaciones de consumación de funciones asíncronas actualizan las propiedades asociadas con las respuestas pendientes (ver [PAD00]) de las llamadas telefónicas
  - Las notificaciones de cambio de estado de las llamadas telefónicas actualizan el estado de las llamadas telefónicas
  - Las notificaciones de cambio de estado de los dispositivos de línea actualizan el estado de los dispositivos de línea
  - Las notificaciones de cambio de estado de las direcciones actualizan el estado de las direcciones

- *Externas*

  Relacionan objetos del sistema con objetos pertenecientes al ambiente: el Control Administrativo y el sistema operativo. Estas son:
  - El usuario usa las interfaces de usuario para crear llamadas telefónicas y para realizar operaciones sobre llamadas telefónicas
  - El Control Administrativo procesa las solicitudes de las interfaces de usuario
  - El Control Administrativo transfiere solicitudes de operaciones telefónicas al objeto Tapi

- El Control Administrativo procesa los mensajes relacionados con las llamadas telefónicas provenientes del objeto Tapi
- Las notificaciones de consumación de funciones asíncronas generan mensajes destinados al Control Administrativo
- Las notificaciones de cambios de estado de las llamadas telefónicas generan mensajes destinados al Control Administrativo
- El sistema operativo notifica al objeto Tapi de la realización de llamadas telefónicas desde el aparato telefónico y de nuevas llamadas entrantes

Dados los objetos y las relaciones existentes entre ellos se establecen los contextos generales en los que ocurren estas relaciones. Los contextos definidos para HTT son:

- *Organización y administración de los objetos*
  - Estructura
  - Administración general
- *Comunicación con el Control Administrativo de Llamadas*
  - Solicitudes provenientes del Control Administrativo para la ejecución de operaciones sobre llamadas telefónicas: realizar una llamada, marcar dígitos, aceptar y contestar una llamada y terminar una llamada
  - Notificaciones de consumación de funciones asíncronas
  - Notificaciones de cambios de estado de las llamadas telefónicas al Control Administrativo
- *Comunicación con el sistema operativo*
  - Notificaciones provenientes del sistema operativo de nuevas llamadas telefónicas entrantes y de llamadas salientes realizadas desde el aparato telefónico
  - Notificaciones provenientes del sistema operativo referentes a la consumación de funciones asíncronas
  - Notificaciones provenientes del sistema operativo relacionadas con cambios de estado de las llamadas telefónicas, de los dispositivos de línea y de las direcciones

La Tabla 5-2 lista los títulos de los modelos objeto realizados. Con el fin de ejemplificar el trabajo de esta sección, se describirán cuatro modelos objeto representativos. El resto de los modelos objeto se localiza en [PAD00]. En la notación del modelo objeto, el recuadro punteado delimita la frontera entre HTT y su ambiente. La parte interior del recuadro punteado contiene los objetos y relaciones entre objetos pertenecientes a HTT. La parte exterior del recuadro punteado muestra los objetos y las relaciones con objetos del ambiente.

Las llamadas telefónicas se modelan mediante los objetos denominados Call. Un objeto contenedor de llamadas telefónicas (CallArray) almacena los objetos Call asociados con las llamadas telefónicas en curso. El objeto controlador Tapi manipula los objetos Call de acuerdo con las solicitudes de operaciones y las notificaciones de eventos telefónicos

provenientes del ambiente, esto es, del Control Administrativo y del sistema operativo. De esta manera, HTT puede procesar múltiples llamadas telefónicas de manera simultánea.

Título del Modelo Objeto
Estructura general de HTT
Administración general de HTT
Solicitudes de operaciones sobre llamadas telefónicas
Solicitud para realizar una llamada telefónica
Notificación de nueva llamada telefónica (llamada saliente desde el aparato telefónico o llamada entrante)
Notificación de respuestas pendientes
Notificación de cambio de estado de una llamada telefónica
Notificación de cambio de estado de un dispositivo de línea
Actualización del estado de una dirección

Tabla 5-2. Título de los modelos objeto para HTT

La Figura 5-1 muestra el contexto para las solicitudes de operaciones sobre llamadas telefónicas provenientes del Control Administrativo. Estas solicitudes son originadas por el usuario a través de las interfaces de usuario[CAR00]. Las interfaces de usuario transfieren las solicitudes al Control Administrativo, el cual a su vez las envía al objeto Tapi. El objeto Tapi se encarga de localizar la llamada telefónica sobre la cual se desea realizar la operación y ésta solicita el servicio al sistema operativo. Dado que las operaciones telefónicas provistas por HTT constituyen funciones asíncronas, tienen asociado una respuesta pendiente (ver Figura 5-4).

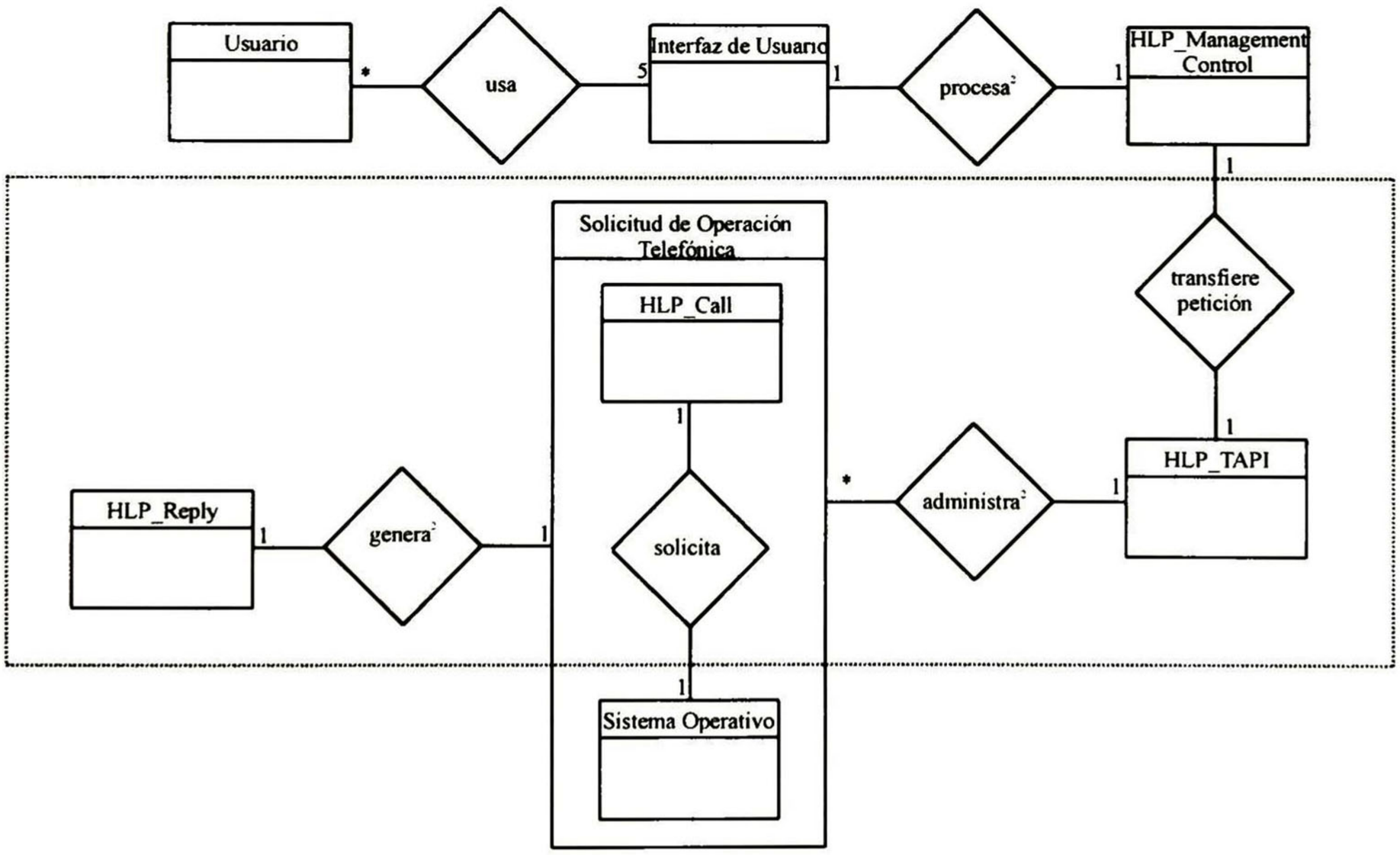


Figura 5-1. Modelo objeto para las solicitudes de operaciones sobre llamadas telefónicas provenientes del Control Administrativo

Antes de poder ejecutar alguna operación sobre una llamada telefónica, ésta debe haber sido creada anteriormente. Existen dos formas de crear llamadas telefónicas salientes: desde las interfaces de usuario[HEL99] (Figura 5-2) y a través del aparato telefónico (Figura 5-3). Cuando se crea una llamada saliente desde alguna interfaz de usuario, el objeto Tapi crea un objeto de llamada saliente (OutgoingCall) y le asigna una dirección. La llamada saliente se encarga de solicitar al sistema operativo la marcación de los dígitos. El sistema operativo asigna un identificador de respuesta pendiente a la solicitud de marcación.

En el caso de las llamadas salientes desde el aparato telefónico, el sistema operativo notifica al objeto Tapi que se ha realizado una marcación. El objeto Tapi crea un objeto de llamada saliente y registra la dirección sobre la cual se ha hecho la marcación. El objeto Tapi genera, además, un mensaje destinado al Control Administrativo para informarle que se desea realizar una llamada saliente. El Control Administrativo decidirá si el usuario está autorizado a llamar al número telefónico marcado. Si este es el caso, solicitará al objeto Tapi que proceda la marcación.

Los objetos correspondientes a llamadas entrantes se generan de la misma manera que las llamadas salientes realizadas desde el aparato telefónico (Figura 5-3). Cuando el objeto Tapi es informado por el sistema operativo de que hay una nueva llamada entrante en alguna dirección, crea un objeto de llamada entrante (IncomingCall). También genera un mensaje para el Control Administrativo notificándole acerca de la nueva llamada. El Control Administrativo verifica que el número del abonado solicitante no esté restringido por el usuario, en cuyo caso, solicita al objeto Tapi que acepte la llamada.

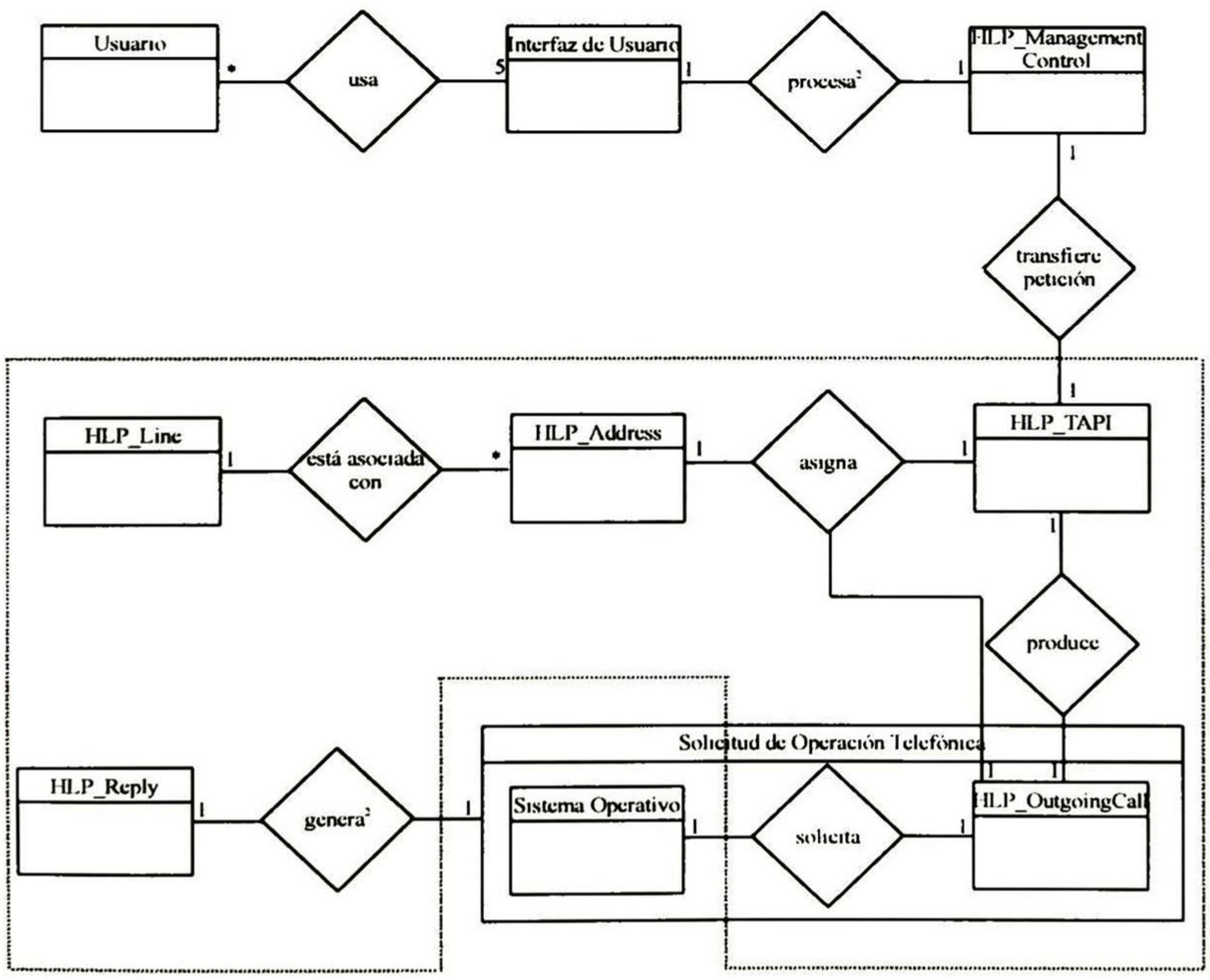


Figura 5-2. Modelo objeto para la creación de llamadas telefónicas desde las interfaces de usuario

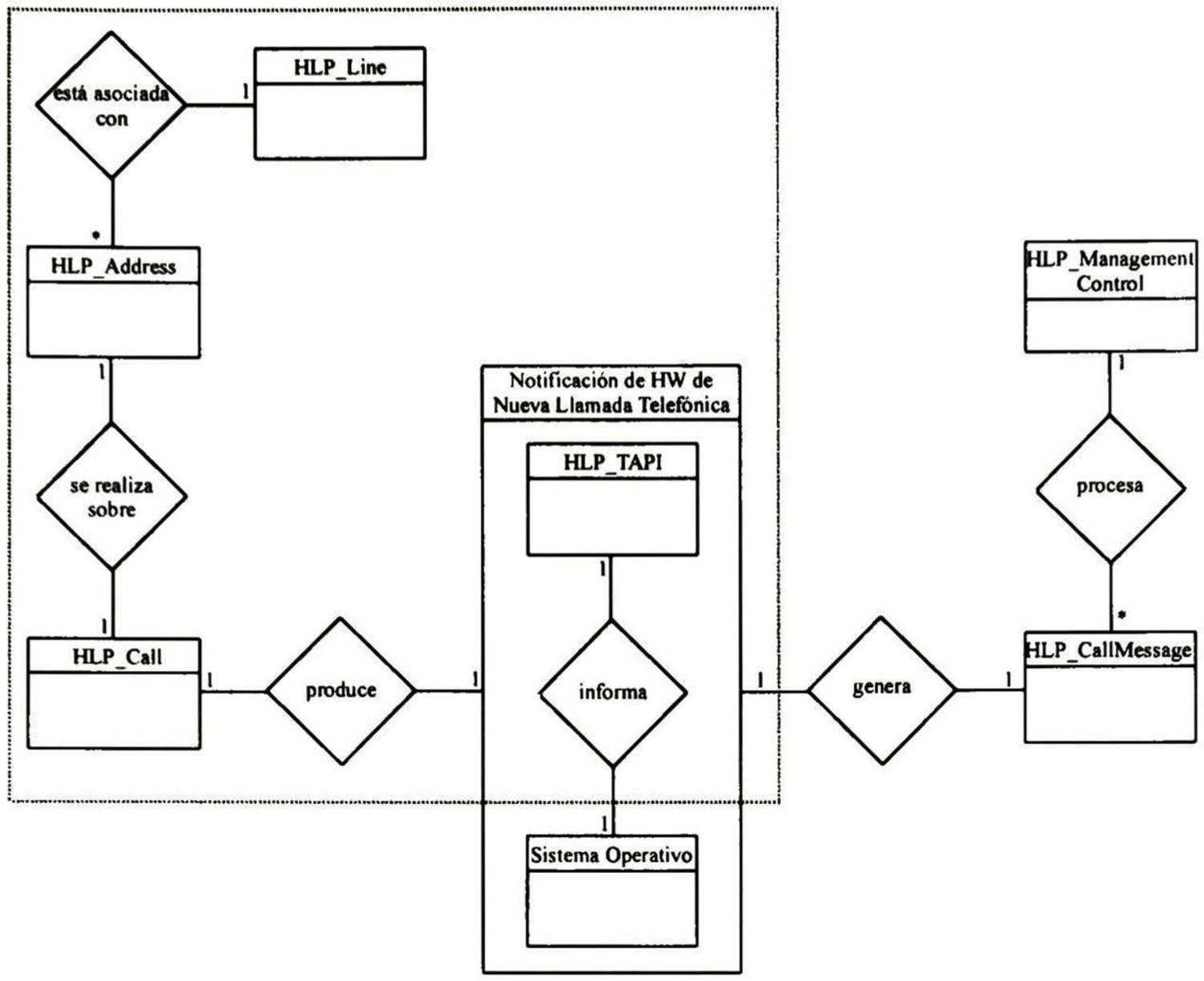


Figura 5-3. Modelo objeto para la notificación de nueva llamada telefónica (entrante o saliente realizada desde el aparato telefónico)

Las respuestas pendientes de funciones asíncronas se representan como objetos (Reply) y se almacenan en un objeto contenedor de respuestas pendientes (ReplyArray). El objeto controlador Tapi previene la ejecución de funciones sobre llamadas telefónicas que tienen asociadas respuestas pendientes de funciones que devuelven "handlers"

Por cada invocación exitosa de alguna función asíncrona se crea un objeto Reply. Este objeto se almacena en un contenedor de respuestas asíncronas pendientes hasta que el sistema operativo notifica al objeto Tapi de la consumación de la función asíncrona invocada. Cuando dicha notificación ocurre, el objeto Reply correspondiente es localizado en el contenedor. Luego se identifica el objeto Call asociado con la respuesta asíncrona pendiente y de acuerdo con el resultado de la función – éxito o fracaso – el objeto Tapi realiza el procesamiento adecuado.

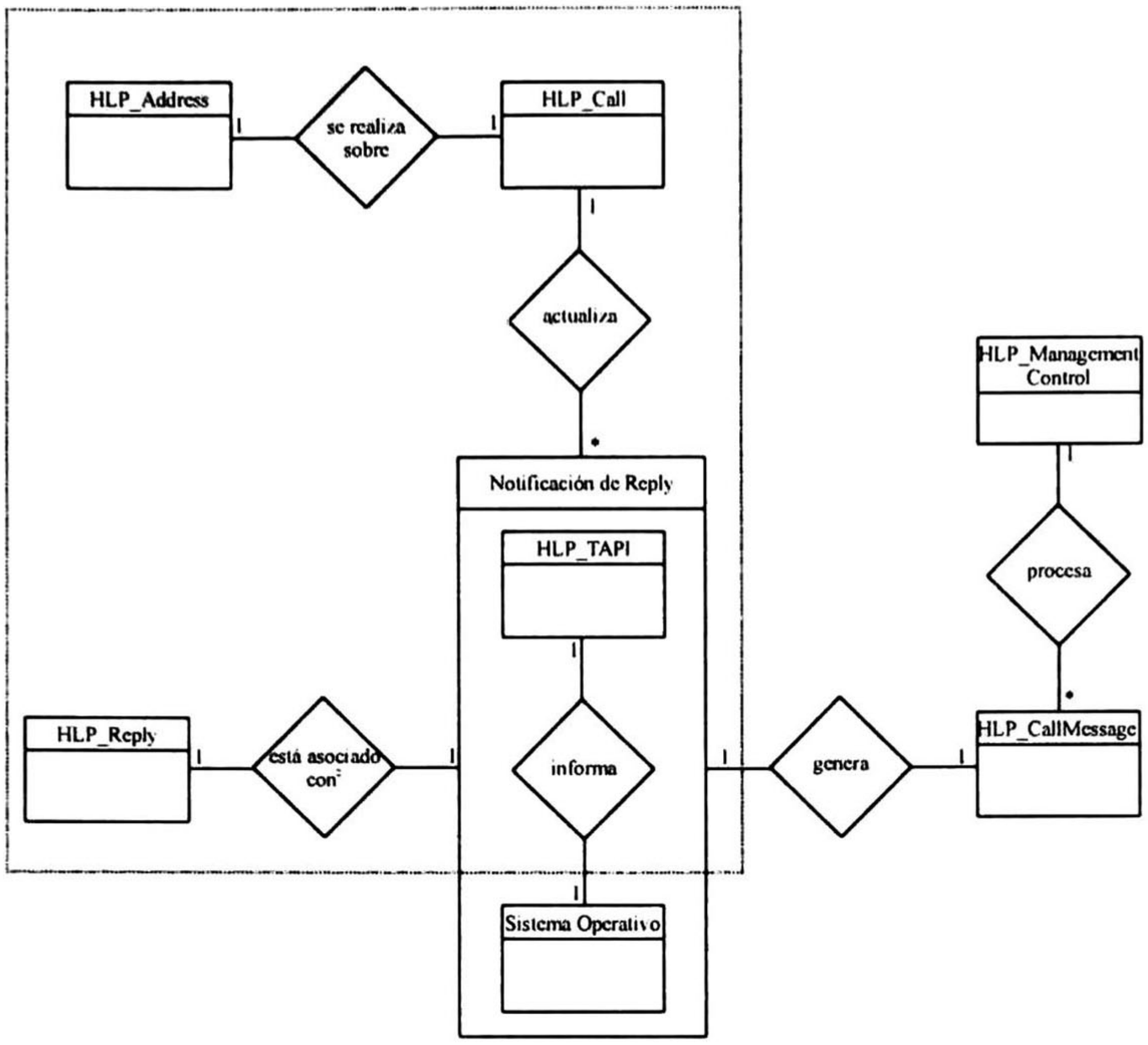


Figura 5-4. Modelo objeto para la notificación de respuestas pendientes de funciones asíncronas

5.3.1.2 Escenarios

El propósito de los escenarios es presentar la secuencia de eventos en el tiempo entre el sistema y los módulos que constituyen su ambiente: el Control Administrativo y el sistema operativo. Los eventos del sistema pueden ser de dos tipos: de entrada, esto es, los que recibe, y de salida, o los que envía. A continuación se presenta una clasificación de las acciones que generan eventos. Además se especifican los eventos de entrada y salida relacionados con cada acción. Estas acciones fueron obtenidas de la descripción funcional de HTT presentada en el capítulo 4.

1. Administración de los dispositivos de línea y las direcciones

Eventos de entrada relacionados:

- InitializeTAPI
- EnumerateResources
- $\text{PrepareLine}_{\text{specifiedAddress}}$
- $\text{PrepareLine}_{\text{allLines}}$
- CloseLines
- ShutdownTAPI
- $\text{UpdateLineState}_{\text{inservice-outofservice}}$
- UpdateAddressState

- lineInitializeExResponse
- lineNegotiateAPIVersionResponse
- lineGetDevCapsResponse
- lineGetAddressCapsResponse
- lineOpenResponse
- lineSetStatusMessagesResponse
- lineCloseResponse
- lineGetLineDevStatusResponse
- lineGetAddressStatusResponse
- lineShutdownResponse

Eventos de salida relacionados:

- lineInitializeEx
- lineNegotiateAPIVersion
- lineGetDevCaps
- lineGetAddressCaps
- lineOpen
- lineSetStatusMessages
- lineClose
- lineGetLineDevStatus
- lineGetAddressStatus
- lineShutdown
- ERR_TAPIALREADYINITIALIZED
- ERR_NOAVAILABLERESOURCES
- ERR_TAPIALREADYSHUTDOWN
- ERR_ADDRESSNOTFOUND
- ERR_LINENOTFOUND

2. *Las solicitudes para realizar llamadas salientes*

   Eventos de entrada relacionados:

   - $\text{MakeCall}_{\text{specifiedAddress}}$
   - $\text{MakeCall}_{\text{unspecifiedAddress}}$
   - lineMakeCallResponse

   Eventos de salida relacionados:

   - lineMakeCall
   - ERR_CALLIDALREADYREGISTERED
   - ERR_ADDRESSNOTFOUND
   - ERR_LINENOTFOUND

3. *Las notificaciones de llamadas entrantes*

   Eventos de entrada relacionados:

   - $\text{CreateCall}_{\text{incoming}}$
   - AcceptIncomingCallResponse

- lineGetCallInfoResponse

Eventos de salida relacionados:

- AcceptIncomingCallRequest
- lineGetCallInfo

4. *Las notificaciones de llamadas salientes realizadas desde el aparato telefónico*

   Eventos de entrada relacionados:

   - $CreateCall_{outgoing}$
   - MakeCallResponse
   - lineGetCallInfoResponse

   Eventos de salida relacionados:

   - MakeCallRequest
   - lineGetCallInfo

5. *Las solicitudes para ejecutar operaciones sobre las llamadas*

   Eventos de entrada relacionados:

   - DialNumber
   - AcceptIncomingCall
   - HangUpCall
   - lineDialResponse
   - lineAcceptResponse
   - lineAnswerResponse
   - lineDropResponse
   - lineDeallocateCallResponse

   Eventos de salida relacionados:

   - lineDial
   - lineAccept
   - lineAnswer
   - lineDrop
   - lineDeallocateCall
   - ERR_ALREADYDISCONNECTINGCALL
   - ERR_CALLNOTFOUND

6. *Las notificaciones relacionadas con la evolución de las llamadas*

   Eventos de entrada relacionados:

   - $CompleteAsynchronousFunction_{makeCall\text{-}dial\text{-}accept\text{-}answer\text{-}drop}$
   - $UpdateCallState_{connected\text{-}busy\text{-}disconnnected\text{-}idle}$
   - lineGetCallStatusResponse

   Eventos de salida relacionados:

   - lineGetCallStatus
   - InsertMessage

Exceptuando el primero, los puntos anteriores están relacionados con acciones que suceden durante el ciclo de vida de las llamadas telefónicas. Éstas pueden ser creadas y terminadas de formas distintas, lo cual genera diferentes secuencias de acciones. Los escenarios desarrollados (ver Tabla 5-3) reflejan estas secuencias así como el manejo de los dispositivos de línea y las direcciones. Enseguida se muestran algunos escenarios de HTT y su descripción. El total de los mismos puede consultarse en [PAD00].

Título del Escenario
Inicialización de los dispositivos de línea y las direcciones
Liberación de los dispositivos de línea y las direcciones
Llamada entrante terminada por software
Llamada entrante terminada desde el teléfono local
Llamada entrante terminada desde el teléfono remoto
Llamada saliente desde software terminada por software
Llamada saliente desde software terminada desde el teléfono local
Llamada saliente desde software terminada desde el teléfono remoto
Llamada saliente desde el aparato telefónico terminada por software
Llamada saliente desde el aparato telefónico terminada desde el teléfono local
Llamada saliente desde el aparato telefónico terminada desde el teléfono remoto

Tabla 5-3. Escenarios definidos para HTT

La Figura 5-5 presenta el escenario para la habilitación de los dispositivos de línea. El Control Administrativo solicita que se habilite el uso de los dispositivos de línea y las direcciones (punto 1). El sistema operativo regresa un handler con el que TAPI identifica a la aplicación, el número de dispositivos de línea disponibles y la versión más actual de TAPI que soporta la aplicación. Enseguida, el Control Administrativo indica que para cada dispositivo de línea y cada dirección se obtengan las capacidades telefónicas y que se negocie la versión de TAPI con la que se manejarán (punto 2). Por último, se configura cada dirección para que pueda recibir llamadas entrantes (punto 3).

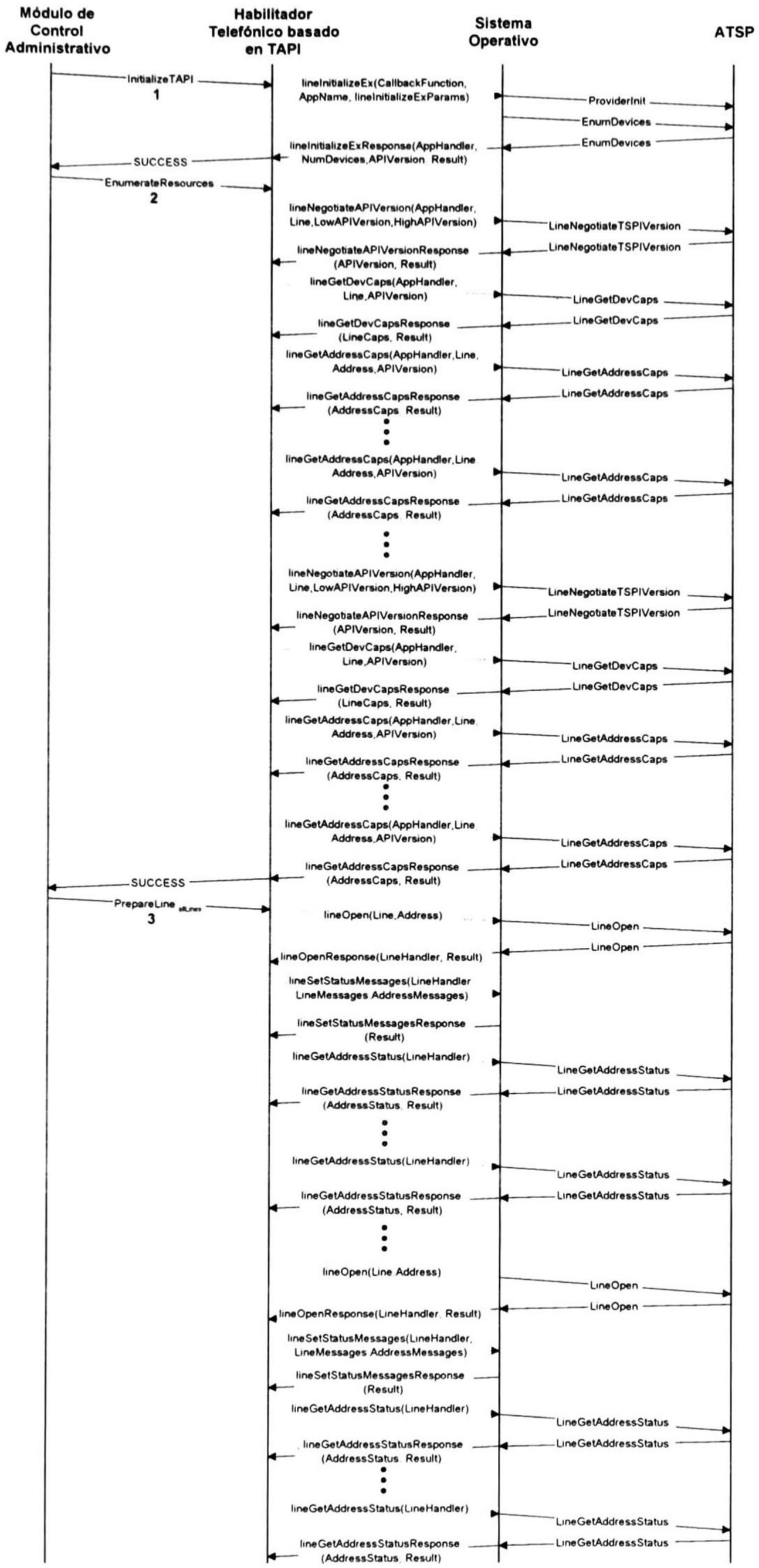


Figura 5-5. Escenario para la habilitación de los dispositivos de línea

La Figura 5-6 muestra el escenario para llamadas entrantes terminada desde software. La parte del diagrama que se localiza sobre la línea punteada refleja las operaciones del sistema y los eventos que se realizan antes y durante el establecimiento de la llamada. El sistema operativo notifica a HTT que hay una nueva llamada telefónica entrante y le envía un handler para que TAPI pueda procesar cualquier acción posterior que se realice sobre la misma (punto 1). Además se indica el handler de la dirección por la cual se recibió la llamada. HTT le solicita información adicional para la llamada, principalmente el callerID (punto 2). Una vez que se tiene esta información, HTT notifica al Control Administrativo de la nueva llamada entrante para que determine si ésta puede ser aceptada y respondida (punto 3). En caso afirmativo, el Control Administrativo envía a HTT el mensaje correspondiente (punto 4) y HTT, a su vez, genera los mensajes necesarios para indicar al sistema operativo que se acepte y conteste la llamada telefónica. Cuando la llamada alcanza el estado Connected, se le notifica al Control Administrativo para que inicie la tarificación sobre esa llamada (punto 5).

La parte del diagrama que se localiza bajo la línea punteada muestra lo que sucede cuando el Control Administrativo indica que se termine la llamada. El Control Administrativo comunica a HTT que termine la llamada telefónica (punto 6). Esta petición se envía al sistema operativo, revisando previamente si la llamada no ha sido finalizada aún por hardware. La terminación de la llamada se comunica a HTT mediante el estado Idle (punto 7). Después, se libera el handler de TAPI para la llamada y los recursos asociados con ella. Por último, se configura de nuevo la dirección por la que se realizó la llamada para que pueda recibir nuevas llamadas entrantes.

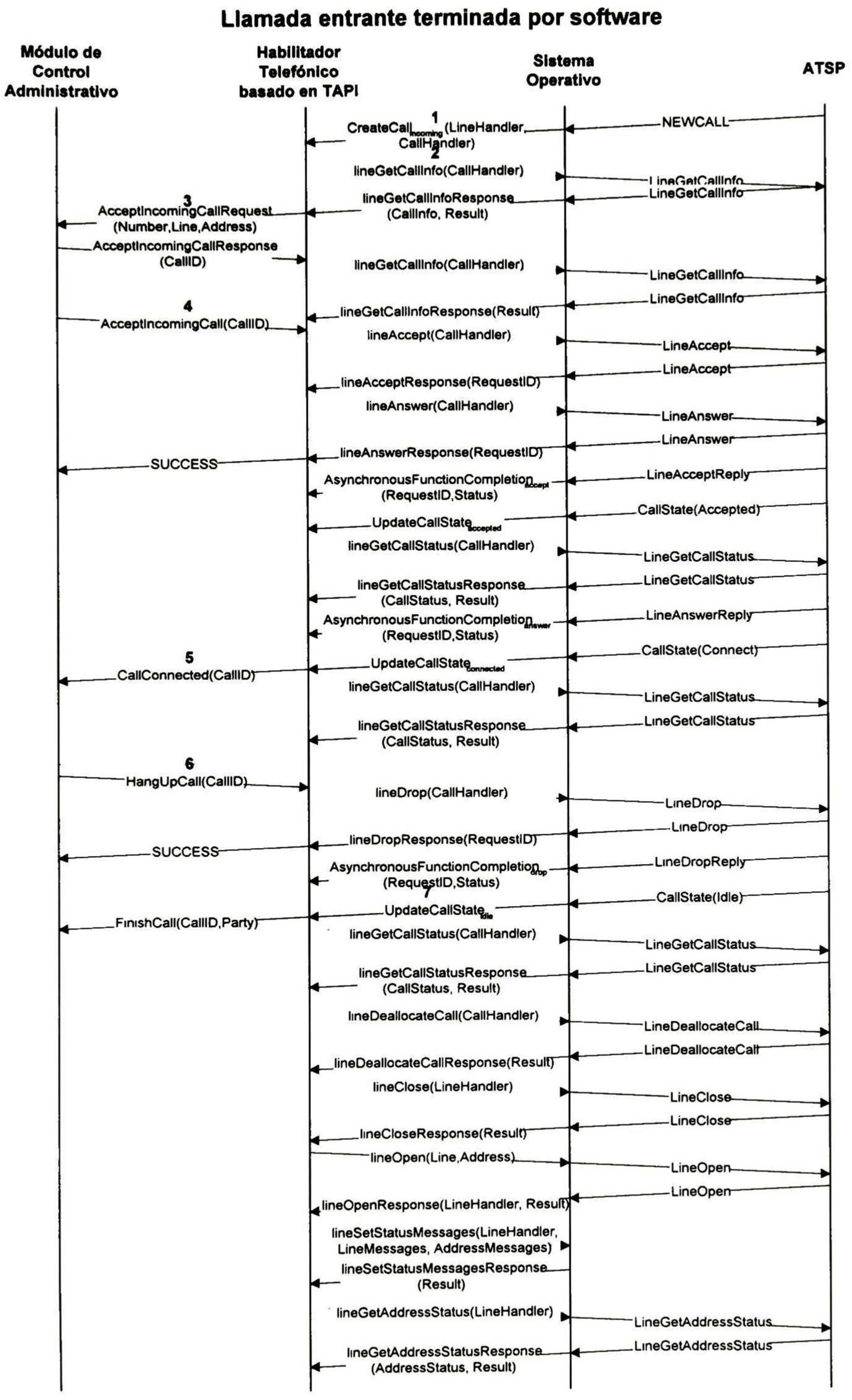


Figura 5-6. Escenario de llamada entrante terminada desde software

La Figura 5-7 presenta el escenario para llamada saliente terminada desde el aparato telefónico local. La parte del diagrama que se localiza sobre la línea punteada refleja las operaciones del sistema y los eventos que se realizan antes y durante el establecimiento de la llamada. El Control Administrativo indica a HTT que realice una llamada telefónica saliente y le informa la dirección por la cual realizar la llamada (punto 1). HTT reconfigura el handler de la dirección correspondiente para realizar una llamada y enseguida solicita al sistema operativo que realice la marcación del número (punto 2). El sistema operativo notifica a HTT los estados por los que atraviesa la llamada antes de alcanzar el estado Connected. Cuando esto ocurre, HTT envía un mensaje al Control Administrativo para que comience la tarificación sobre esa llamada (punto 3).

La parte del diagrama que se localiza bajo la línea punteada muestra lo que ocurre cuando el sistema operativo notifica a HTT que la llamada telefónica ha sido terminada por el teléfono local (ya que se recibe el estado Idle). HTT envía un mensaje al Control Administrativo para que finalice la tarificación (punto 4). Enseguida, se libera el handler de TAPI para la llamada y los recursos asociados con ella. Por último, se configura de nuevo la dirección por la que se realizó la llamada para que pueda recibir llamadas entrantes.

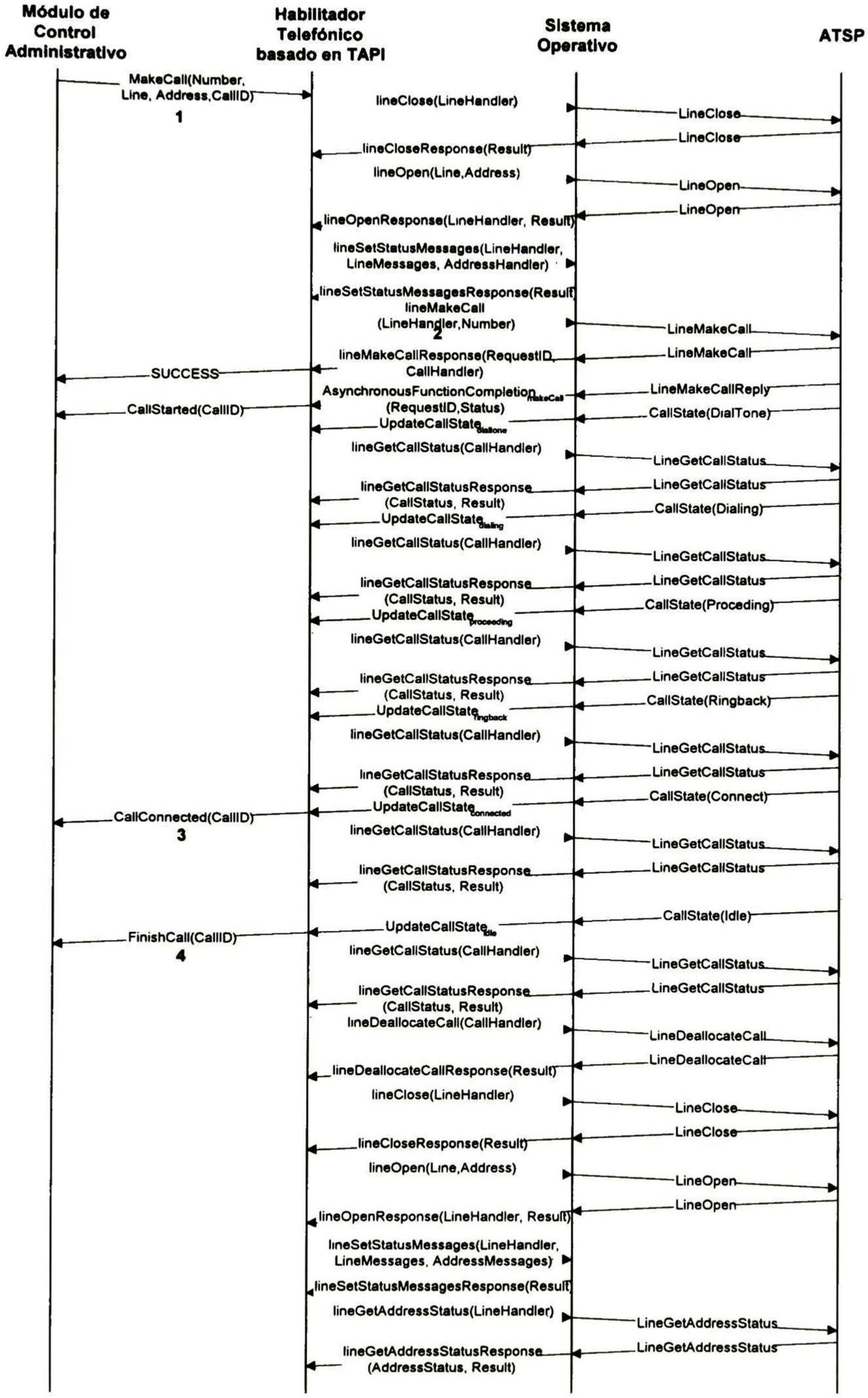


Figura 5-7. Escenario de llamada saliente terminada desde el telefóno local

5.3.1.3 Modelo de Ciclo de Vida

El modelo de ciclo de vida describe los patrones de comunicación entre el sistema y su medio ambiente. Este modelo se plantea en términos de expresiones de ciclo de vida que generalizan los escenarios. Las expresiones de ciclo de vida son extensiones simples de las expresiones regulares o gramáticas[COL94]. Las expresiones de ciclo de vida son más expresivas que los diagramas de tiempo porque pueden representar repetición, concatenación, alternativa y entrelazamiento. Las expresiones de ciclo de vida se construyen mediante los siguientes elementos:

- *Alfabeto*

  Constituido por los eventos de entrada y de salida. Los eventos de salida llevan como prefijo el símbolo #.

- *Operadores*

  Dadas *x* y *y* expresiones de ciclo de vida, entonces:
  - *x* . *y* denota *y* se sigue de *x*
  - *x* | *y* denota *x* o *y* ocurren
  - *x** denota cero o más ocurrencias de *x*
  - *x*+ denota una o más ocurrencias de *x*
  - [*x*] denota *x* es opcional
  - *x* || *y* denota *x* y *y* se pueden ejecutar en orden indistinto

  La precedencia de operadores en orden decreciente es : [ ] , *, +, . , | , ||

- *Sustituciones*

  A una expresión de ciclo de vida se le puede asignar un nombre. Este nombre puede ser usado en otras expresiones. Las sustituciones no deben de ser recursivas.

La Figura 5-8 muestra un resumen del ciclo de vida de HTT. El ciclo de vida se compone de siete expresiones principales. Las tres primeras denotan una secuencia para la inicialización, la enumeración y la configuración de los dispositivos de línea y las direcciones, respectivamente. Una vez que se han llevado a cabo estas operaciones el sistema puede realizar llamadas salientes o recibir llamadas entrantes indistintamente. En la última parte del ciclo de vida del sistema se cierran todos los dispositivos de línea y las direcciones y finaliza su uso.

```
lifecycle HTT:                      LineDevicesInitialization .
                                    ResourcesEnumeration .
                                    LineDevicesConfiguration .
                                    ( IncomingCall ||
                                      SoftwareOutgoingCall ||
                                      HardwareOutgoingCall
                                    )*
                                    LineDevicesShutdown

LineDevicesInitialization =         InitializeTAPI .
                                    ( #ERR_TAPIALREADYINITIALIZED |
```

```
                                    #lineInitializeEx .
                                    lineInitializeExResponse .
                                    ( #SUCCESS |
                                      #ERR_NOAVAILABLERESOURCES |
                                      #ERR_INITIALIZETAPIFAILED
                                    )
                                  )

ResourcesEnumeration =            EnumerateResources .
                                  ( #lineNegotiateAPIVersion .
                                    lineNegotiateAPIVersionResponse .
                                    #lineGetDevCaps .
                                    lineGetDevCapsResponse .
                                    ( #lineGetAddressCaps .
                                      lineGetAddressCapsResponse
                                    )+
                                  )* .
                                  ( #SUCCESS |
                                    #ERR_NOAVAILABLERESOURCES
                                  )

LineDevicesConfiguration =        SpecificLineDeviceConfiguration |
                                  AllLineDevicesConfiguration

SpecificLineDeviceConfiguration = PrepareLine specifiedAddress .
                                  ( #ERR_LINENOTFOUND |
                                    #ERR_ADDRESSNOTFOUND |
                                    ( #lineOpen .
                                      lineOpenResponse .
                                      ( #ERR_OPENLINEFAILED |
                                        ( #lineSetStatusMessages .
                                          lineSetStatusMessagesResponse .
                                          ( #ERR_SETMESSAGESFAILED |
                                            ( #lineGetAddressStatus .
                                              lineGetAddressStatusResponse .
                                              ( #SUCCESS |
                                                #ERR_GETADDRESSSTATUSFAILED
                                              )
                                            )+
                                          )
                                        )
                                      )
                                    )
                                  )

LineDevicesShutdown =             CloseLines.
                                  ( #lineClose .
                                    lineCloseResponse
                                  )*
                                  ShutdownTAPI .
                                  ( #ERR_TAPIALREADYSHUTDOWN |
                                    ( #lineShutdown .
                                      lineShutdownResponse .
                                      ( #SUCCESS |
                                        #ERR_SHUTDOWNTAPIFAILED
                                      )
```

```
                                    )
                                )

IncomingCall =                      IncomingCallNotification .
                                    IncomingCallAcceptance .
                                    (NumberDialing)*
                                    CallTermination

SoftwareOutgoingCall =              SoftwareOutgoingCallConnection .
                                    (NumberDialing)*
                                    CallTermination

HardwareOutgoingCall =              HardwareOutgoingCallNotification .
                                    HardwareOutgoingCallAcceptance .
                                    (NumberDialing)*
                                    CallTermination

CallTermination =                   LocalPhoneCallTermination | RemotePhoneCallTermination |
                                    SoftwareCallTermination
```

Figura 5-8. Resumen del ciclo de vida de HTT

5.3.1.4 Modelo de Operación

Este modelo define el conjunto de operaciones del sistema. Las operaciones del sistema están constituidas por los eventos de entrada que recibe el sistema y el efecto que éstos tienen en su estado. Por lo tanto, el conjunto de operaciones del sistema para HTT (ver Tabla 5-4) está formado por la suma de los distintos eventos de entrada de cada uno de los escenarios desarrollados.

El efecto que una operación del sistema tiene en el estado del sistema se describe mediante uno o más esquemas. Un esquema es un formato en el cual se establece para cada operación del sistema la siguiente información[18]:

- Nombre
- Descripción
- Entradas
- Modificaciones
- Salidas
- Precondiciones
- Postcondiciones

[18] La descripción de los campos de un esquema puede consultarse en el Apéndice B.

Operación del Sistema
InitializeTAPI
EnumerateResources
$PrepareLine_{specifiedAddress}$
$PrepareLine_{allLines}$
CloseLines
ShutdownTAPI
$MakeCall_{specifiedAddress}$
$MakeCall_{unspecifiedAddress}$
DialNumber
AcceptIncomingCall
HangUpCall
$CompleteAsynchronousFunction_{makeCall}$
$CompleteAsynchronousFunction_{dial\text{-}accept\text{-}answer}$
$CompleteAsynchronousFunction_{drop}$
$UpdateCallState_{connected}$
$UpdateCallState_{busy}$
$UpdateCallState_{disconnected}$
$UpdateCallState_{idle}$
$UpdateLineState_{inservice\text{-}outofservice}$
UpdateAddressState
$CreateCall_{outgoing}$
$CreateCall_{incoming}$
Todas las respuestas a funciones TAPI (ver Diccionario de Datos en [PAD00]).

Tabla 5-4. Operaciones del sistema de HTT

A continuación se exponen algunos de los esquemas más representativos de HTT. La totalidad de los mismos puede ser consultada en [PAD00].

Operación:	InitializeTAPI
Descripción:	El Control Administrativo solicita el registro de la aplicación ante TAPI para hacer uso de los dispositivos de línea y las direcciones.
Entradas:	
Modificaciones:	*tapi*
Salidas:	ManagementControl:{ERR_TAPIALREADYINITIALIZED, ERR_NOAVAILABLERESOURCES, SUCCESS} Sistema Operativo: {lineInitializeEx}
Precondiciones:	No se han inicializado los dispositivos de línea.
Postcondiciones:	**Si** *tapi*.appHandler es válido **entonces** ERR_TAPIALREADYINITIALIZED se ha enviado a ManagementControl. **en otro caso** lineInitializeEx se ha enviado al Sistema Operativo, **Si** se han inicializado exitosamente los dispositivos de línea **entonces** **Si** *tapi*.numDevices es igual a 0 **entonces** ERR_NOAVAILABLERESOURCES se ha enviado a

	ManagementControl. **en otro caso** SUCCESS se ha enviado a ManagementControl. **en otro caso** ERR_INITIALIZETAPIFAILED se ha enviado a ManagementControl

Operación:	EnumerateResources
Descripción:	El Control Administrativo solicita que se obtenga información referente a los recursos telefónicos disponibles para la aplicación (dispositivos de línea y direcciones), así como las capacidades de cada uno de ellos.
Entradas:	*tapi*
Modificaciones:	(**new** *line*: HLP_Line)*, (**new** *addressArray*: HLP_AddressArray)*, (**new** *address*: HLP_Address)*, lineArray, administra, está asociada con
Salidas:	Sistema Operativo: {lineNegotiateAPIVersion, lineGetDevCaps, lineGetAddressCaps} ManagementControl: {SUCCESS, ERR_NOAVAILABLERESOURCES}
Precondiciones:	Se han inicializado exitosamente los dispositivos de línea.
Postcondiciones:	Se han inicializado exitosamente los dispositivos de línea. Por cada dispositivo de línea notificado mediante lineInitialize se ha creado un objeto *line*. **Para** cada objeto *line* lineNegotiateAPIVersion se ha enviado al Sistema Operativo, **Si** se ha negociado exitosamente la versión de *line* **entonces** lineGetDevCaps se ha enviado al Sistema Operativo, **Si** se han obtenido exitosamente las capacidades de *line* **entonces** Por cada dirección asociada a *line* se ha creado un objeto *address*. **Para** cada objeto *address* lineGetAddressCaps se ha enviado al Sistema Operativo, **Si** se han obtenido exitosamente las capacidades de *address* **entonces** *address* se ha insertado en *line*.addressArray. **Si** *line*.addressArray tiene al menos un objeto *address* **entonces** *line* se ha insertado en lineArray **Si** lineArray tiene al menos un objeto *line* **entonces** SUCCESS se ha enviado a ManagementControl **en otro caso** ERR_NOAVAILABLERESOURCES se ha enviado a ManagementControl.

Operación:	$MakeCall_{specifiedAddress}$
Descripción:	El Control Administrativo solicita la realización de una llamada telefónica especificando la dirección por la que se llevará a cabo.
Entradas:	**supplied** *destNumber*: cadena, **supplied** *countryCode*: cadena, **supplied** *lineID*: identificadorLineDevice, **supplied** *addressID*: identificadorDirección, **supplied** *callID*: identificadorLlamada, *line* con line.ID igual a *lineID*, *line*.addressarray, lineArray

Modificaciones:	callArray replyArray *address* contenida en *line*.addressArray con *address*.ID igual a *addressID* **new** *outgoing*: HLP_OutgoingCall solicita administra[2] genera[2] produce asigna se realiza sobre
Salidas:	ManagementControl:{ERR_CALLIDALREADYREGISTERED, ERR_LINENOTFOUND, ERR_ADDRESSNOTFOUND, ERR_CLOSELINEFAILED, ERR_OPENLINEFAILED, ERR_SETMESSAGESFAILED, ERR_CONNECTIONFAILED, SUCCESS} Sistema Operativo:{lineClose, lineOpen, lineSetStatusMessages, lineMakeCall}
Precondiciones:	Se han inicializado exitosamente los dispositivos de línea.
Postcondiciones:	Se han inicializado exitosamente los dispositivos de línea. **Si** existe un objeto call de tipo HLP_Call en callArray con call.ID igual a *callID* **entonces** ERR_CALLIDALREADYREGISTERED se ha enviado a ManagementControl **en otro caso** **Si** *line* está contenido en lineArray **entonces** **Si** *line*.status es diferente de LINEDEVSTATE_OUTOFSERVICE y *line*.bearerModes soporta LINEBEARERMODE_VOICE y *line*.mediaModes soporta LINEMEDIAMODE_INTERACTIVEVOICE y *line*.lineFeatures soporta LINEFEATURE_MAKECALL **entonces** **Si** *address* está contenida en *line*.addressArray y *address*.numActiveCalls < *address*.maxNumActiveCalls y *address*.addressFeatures soporta LINEFEATURE_MAKECALL **entonces** Se ha enviado lineClose al Sistema Operativo, **Si** *address* se ha cerrado exitosamente **entonces** lineOpen se ha enviado al Sistema Operativo, **Si** *address* se ha abierto exitosamente **entonces** lineSetStatusMessages se ha enviado al Sistema Operativo, **Si** se ha establecido exitosamente la notificación de mensajes para *address* **entonces** lineMakeCall se ha enviado al Sistema Operativo, **Si** la solicitud de realizar llamada se ha completado síncronamente con éxito **entonces** **final** *address*.numActiveCalls = **initial** *address*.numActiveCalls + 1, *outgoing*.ID igual a *CallID*, *outgoing*.calledID igual a *destNumber*, *outgoing*.status igual a LINECALLSTATE_IDLE, *outgoing* se ha insertado en callArray, SUCCESS se ha enviado a ManagementControl **en otro caso** ERR_CONNECTIONFAILED se ha enviado a ManagementControl **en otro caso** ERR_SETMESSAGESFAILED se ha enviado a ManagementControl

en otro caso
ERR_OPENLINEFAILED se ha enviado a ManagementControl
en otro caso
ERR_CLOSELINEFAILED se ha enviado a ManagementControl
en otro caso
ERR_ADDRESSNOTFOUND se ha enviado a ManagementControl
en otro caso
ERR_LINENOTFOUND se ha enviado a ManagementControl.

Operación:	HangUpCall
Descripción:	El Control Administrativo solicita la terminación de una llamada telefónica.
Entradas:	**supplied** *callID*: identificadorLlamada, callArray
Modificaciones:	*call* con *call*.ID igual a *callID* solicita administra[2] genera[2]
Salidas:	ManagementControl:{ERR_CALLNOTFOUND, ERR_DISCONNECTINGCALL, ERR_DROPCALLFAILED, SUCCESS} Sistema Operativo:{lineDrop}
Precondiciones:	Se han inicializado exitosamente los dispositivos de línea.
Postcondiciones:	Se han inicializado exitosamente los dispositivos de línea. **Si** *call* está contenido en callArray **entonces** **Si** *call*.stoppingCall igual a false **entonces** **Si** *call*.status diferente de LINECALLSTATE_IDLE y de LINECALLSTATE_DISCONNECTED **entonces** lineDrop se ha enviado al Sistema Operativo, **Si** la solicitud de terminar llamada se ha completado síncronamente con éxito **entonces** *call*.stoppingCall igual a true, SUCCESS se ha enviado a ManagementControl **en otro caso** ERR_DROPCALLFAILED se ha enviado a ManagementControl **en otro caso** ERR_ALREADYDISCONNECTINGCALL se ha enviado a ManagementControl **en otro caso** ERR_CALLNOTFOUND se ha enviado a ManagementControl.

Operación:	$CompleteAsynchronousFunction_{makeCall}$
Descripción:	El Sistema Operativo informa de la consumación de la función TAPI asíncrona lineMakeCall.
Entradas:	**supplied** *device*: TAPIdevice, **supplied** *message*: mensaje, **supplied** *callbackInst*: callback, **supplied** *param1*: parámetro, **supplied** *param2*: parámetro, **supplied** *param3*: parámetro, callArray *line* con *line*.ID igual a *call*.associatedLine *line*.addressArray
Modificaciones:	*reply* con *reply*.requestID igual a *param1* *call* con *call*.ID igual a *reply*.callID

	address contenido en *line*.addressArray con *address*.ID igual a *call*.addressID **new** *callMessage*: HLP_CallMessage callArray, replyArray se realiza sobre informa está asociado con[2] genera
Salidas:	ManagementControl:{ *callMessage*.CallStarted, *callMessage*.FinishCall }
Precondiciones:	Se han inicializado exitosamente los dispositivos de línea.
Postcondiciones:	Se han inicializado exitosamente los dispositivos de línea. **Si** *reply* está contenido en replyArray **entonces** **Si** *call* está contenido en callArray **entonces** **Si** *reply*.requestType igual a MAKECALLREQUEST **entonces** **Si** *param2* igual a SUCCESS **entonces** *callMessage*.sender igual a "*tapi*", *callMessage*.callID igual a *callID*, *callMessage*.function igual a "CallStarted", *callMessage* se ha enviado a ManagementControl, **final** *call*.pendantReplies igual a **initial** *call*.pendantReplies-1, **en otro caso** lineClose se ha enviado al Sistema Operativo, **Si** *address* se ha cerrado exitosamente **entonces** *address* se ha configurado para recibir llamadas telefónicas entrantes (ver $PrepareLine_{specifiedAddress}$), *callMessage*.sender igual a "*tapi*", *callMessage*.callID igual a *callID*, *callMessage*.function igual a "FinishCall", *callMessage*.party igual a "LOCAL_PARTY_DISCONNECTED", *callMessage* se ha enviado a ManagementControl, *call* ha sido eliminada de callArray. *reply* se ha eliminado de replyArray.

5.3.2 Diseño

El objetivo de la etapa de diseño es describir cómo el sistema realiza la funcionalidad establecida durante el análisis. Por ello, las actividades de diseño se centran en la definición de las clases y en la descripción de la interacción entre los objetos para alcanzar la funcionalidad requerida. Una clase es un modelo de un conjunto de objetos que tienen los mismos atributos y funciones. Por ejemplo, todos los dispositivos de línea tienen asociadas ciertas propiedades y un conjunto de operaciones que pueden realizarse sobre ellos. Entonces podemos definir una abstracción que caracterice a cualquier dispositivo de línea.

Para cada grupo distinto de objetos involucrados en HTT se definió una clase que lo representa. Estas clases se listan en la Tabla 5-5.

Nombre del Objeto	Clase que lo representa
Line	HLP_Line
Address	HLP_Address
Reply	HLP_Reply
Call	HLP_Call
IncomingCall	HLP_IncomingCall
OutgoingCall	HLP_OutgoingCall
LineArray	HLP_LineArray
AddressArray	HLP_AddressArray
CallArray	HLP_CallArray
ReplyArray	HLP_ReplyArray
Tapi	HLP_TAPI
Sistema Operativo	Sistema Operativo
Management Control	HLP_ManagementControl
CallMessage	HLP_CallMessage

Tabla 5-5. Clases del HTT

El diseño de HTT consiste de cuatro modelos: los grafos de interacción entre objetos, los grafos de visibilidad, las descripciones de clases y los grafos de herencia. Cada uno de estos modelos define los siguientes aspectos estructurales del software, respectivamente:

- Secuencia de mensajes entre objetos
- Estructura de referencias
- Clases
- Estructuras de herencia

5.3.2.1 Grafos de Interacción entre objetos

Este modelo muestra la secuencia de paso de mensajes entre los objetos del sistema para llevar a cabo la funcionalidad de cada una de las operaciones del sistema (consultar). A cada operación del sistema le corresponde un diagrama de interacción entre objetos. El objeto que recibe la solicitud de una operación del sistema en un grafo de interacción se denomina controlador. Los objetos que trabajan en conjunto con el controlador para realizar la operación del sistema son llamados colaboradores.

Dado que el objeto Tapi proporciona toda la interfaz hacia el ambiente, aparece como controlador en todos los grafos de interacción. El resto de los objetos del sistema actúan entonces como colaboradores. De esta forma, el objeto Tapi recibe las solicitudes de operaciones del sistema y se encarga de delegar funciones a otros objetos mediante el paso de mensajes. Éstos a su vez pueden enviar mensajes a otros objetos, y así sucesivamente. A continuación se presentan algunos grafos de interacción del sistema. El resto puede ser consultado en [PAD00].

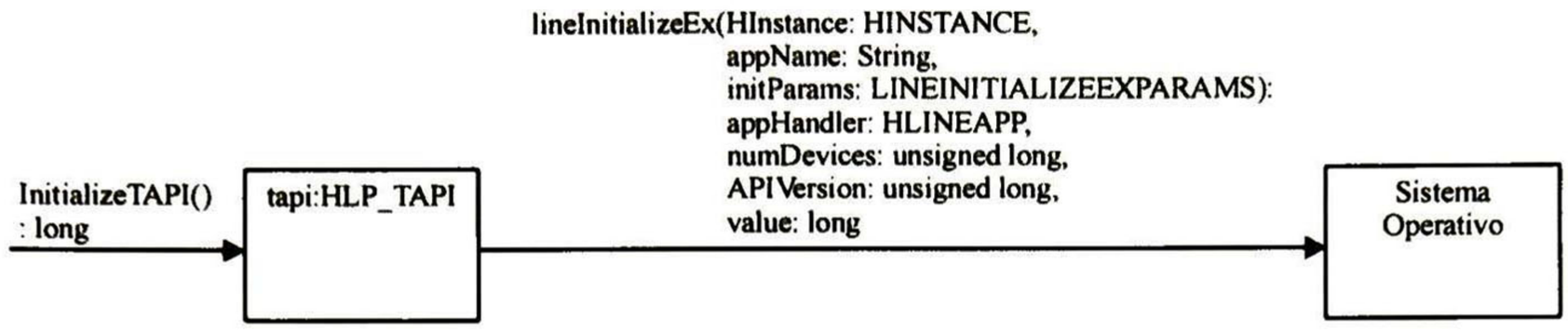


Figura 5-9. Grafo de interacción para la operación del sistema InitializeTAPI

Descripción:

Tipo	Nombre	Descripción	No. de Sec.
operación del sistema	HLP_TAPI : InitializeTAPI( ): long	Indica a la interfaz TAPI que el Control Administrativo desea hacer uso de los dispositivos de línea y las direcciones.	
función TAPI	lineInitializeEx(HInstance:HINSTANCE, appName: String, initParams: LINEINITIALIZEEXPARAMS): appHandler: HLINEAPP, numDevices: unsigned long, value: long	El Control Administrativo registra a a la aplicación de usuario con el nombre appName y establece los parámetros de inicialización a initParams. El número de dispositivos de línea disponibles para el uso del Control Administrativo es numDevices, el handler de TAPI para la aplicación es appHandler y el resultado de la ejecución de la función es value.	1

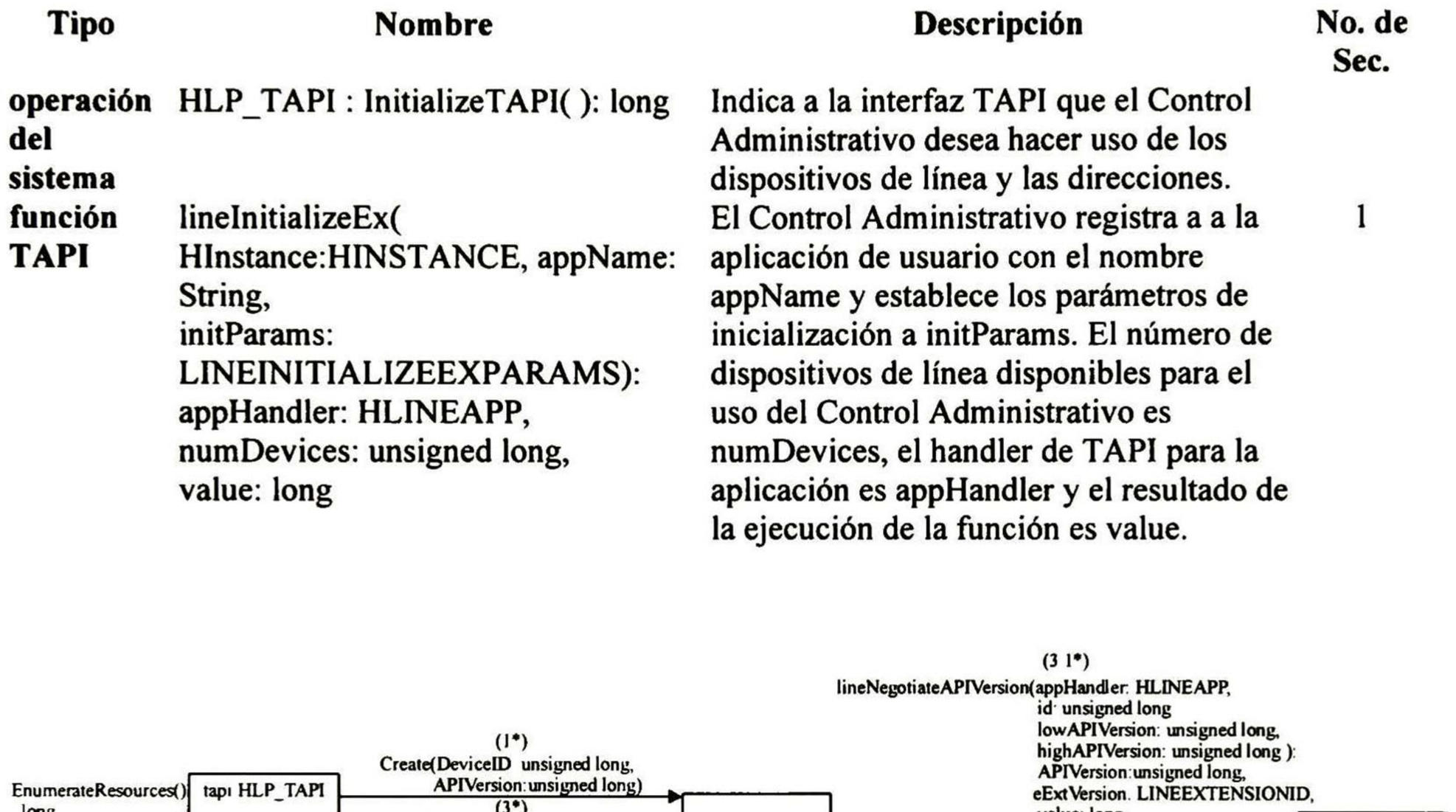


Figura 5-10. Grafo de interacción para la operación del sistema EnumerateResources

Descripción:

Tipo	Nombre	Descripción	No. de Sec.
operación del sistema	HLP_TAPI: EnumerateResources( ): long	Enumera los recursos telefónicos disponibles para el Control Administrativo, así como las capacidades de cada uno de ellos. Crea HLP_TAPI.numDevices objetos de tipo HLP_Line y los inserta en el contenedor de dispositivos de línea HLP_TAPI.lineArray. Por cada dispositivo de línea crea HLP_Line.numAddresses objetos de tipo HLP_Address y los inserta en el objeto HLP_Line.addressArray correspondiente.	
método	HLP_Line:NegotiateAPIVersion(appHandler: HLINEAPP): value:long	Realiza la negociación de la versión de TAPI con la que se maneja un dispositivo de línea.	3*
función TAPI	lineNegotiateAPIVersion(appHandler: HLINEAPP, id: unsigned long, lowAPIVersion: unsigned long, highAPIVersion: unsigned long): APIVersion: unsigned long, extVersion: LINEEXTENSIONID, value: long	Permite al Control Administrativo negociar la versión de TAPI que se usará para el dispositivo de línea con identificador id. El parámetro lowAPIVersion indica la mínima versión de TAPI y highAPIVersion la máxima versión de TAPI soportadas por la aplicación de usuario. La versión negociada se localiza en APIVersion y en caso de que se utilicen extensiones específicas del dispositivo de línea la versión negociada correspondiente se encuentra en extVersion. El éxito o fracaso de esta operación se indica en value.	3.1*
método	HLP_Line:GetDeviceCaps(appHandler: HLINEAPP): value: long	Obtiene las capacidades telefónicas de un dispositivo de línea.	4*
función TAPI	lineGetDevCaps(appHandler: HLINEAPP, id: unsigned long, APIVersion: unsigned long, extVersion: unsigned long): lineCaps: LINEDEVCAPS, value: long	Realiza una consulta sobre el dispositivo de línea con identificador id para conocer sus capacidades telefónicas. Éstas se localizan en la estructura lineCaps. El éxito o fracaso de esta operación se indica en value.	4.1*
método	HLP_Line:GetNumAddresses( ): NumAddresses: int	Obtiene el número de direcciones asociadas con un dispositivo de línea.	5*
método	HLP_Line:GetAPIVersion( ): APIVersion: unsigned long	Devuelve la versión de TAPI con la que se maneja un dispositivo de línea.	6*
método	HLP_Address:GetCaps( ): value:long	Obtiene las capacidades telefónicas de una dirección.	8*
función TAPI	lineGetAddressCaps(appHandler: HLINEAPP, lineID: unsigned long, addressID: unsigned long, APIVersion: unsigned long, extVersion: LINEEXTENSIONID):	Consulta las capacidades telefónicas de la dirección con identificador addressID asociada con el dispositivo de línea con identificador ID. Las capacidades correspondientes se localizan en la estructura addressCaps. El éxito o fracaso	8.1*

	addressCaps: LINEADDRESSCAPS, value: long	de esta operación se indica en value.	
método	HLP_AddressArray: Insert(address: HLP_Address)	Inserta un objeto de tipo HLP_Address en un contenedor de direcciones.	9*
método	HLP_LineArray:Insert(line: HLP_Line)	Inserta un objeto de tipo HLP_Line en un contenedor de dispositivos de línea.	10*

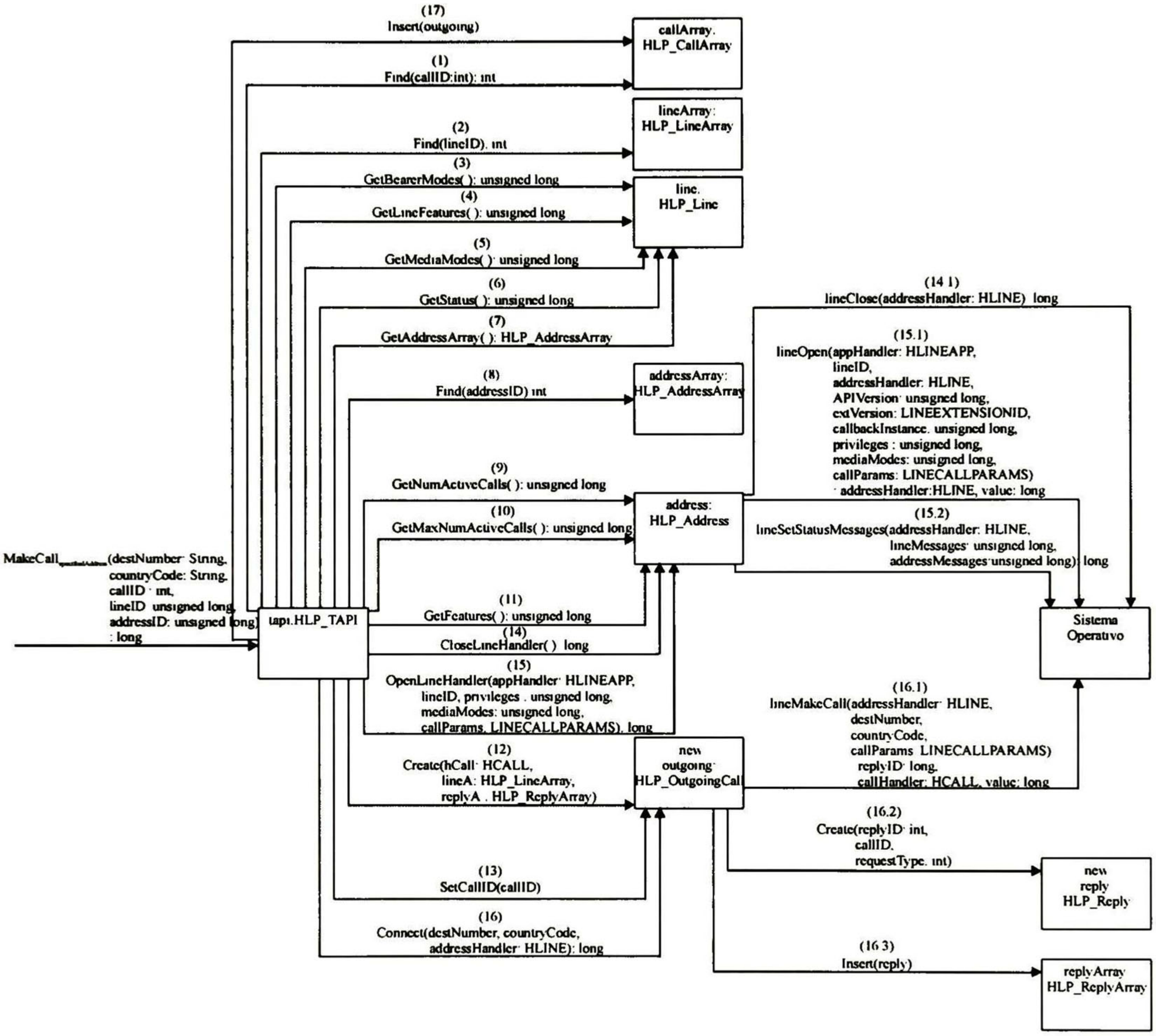


Figura 5-11. Grafo de interacción para la operación del sistema MakeCall

Descripción:

Tipo	Nombre	Descripción	No. de Sec.
operación del sistema	HLP_TAPI:MakeCall$_{specifiedAddress}$(destNumber: String, countryCode: String, callID: int,	Realiza una llamada telefónica especificando la dirección por la cual se llevará a cabo.	

	lineID: unsigned long, addressID: unsigned long): long		
método	HLP_CallArray:Find(callID: int): position: int	Encuentra en el contenedor de llamadas telefónicas la posición donde se localiza la llamada con identificador igual a callID.	1
método	HLP_LineArray:Find(lineID): position: int	Encuentra el dispositivo de línea con identificador igual a lineID en el contenedor de dispositivos de línea. La posición donde lo encontró se localiza en position.	2
método	HLP_Line:GetBearerModes( ): bearerModes: unsigned long	Obtiene los bearer modes soportados por un dispositivo de línea.	3
método	HLP_Line:GetLineFeatures( ): lineFeatures: unsigned long	Obtiene las funciones que se pueden ejecutar sobre un dispositivo de línea.	4
método	HLP_Line:GetMediaModes( ): mediaModes: unsigned long	Obtiene los media modes que soporta un dispositivo de línea.	5
método	HLP_Line:GetStatus( ): status: unsigned long	Regresa el estado actual de un dispositivo de línea.	6
método	HLP_Line:GetAddressArray( ): addressArray: HLP_AddressArray	Devuelve el contenedor de direcciones asociado con un dispositivo de línea.	7
método	HLP_AddressArray:Find(addressID) : position: int	Encuentra la posición donde se localiza la dirección con identificador igual a addressID.	8
método	HLP_Address:GetNumActiveCalls( ): activeCalls: unsigned long	Regresa el número de llamadas activas en una dirección.	9
método	HLP_Address:GetMaxNumActiveC alls( ): maxActiveCalls: unsigned long	Regresa el número máximo de llamadas activas que soporta una dirección.	10
método	HLP_Address:GetFeatures( ): addressFeatures: unsigned long	Devuelve las funciones que se pueden realizar sobre una dirección.	11
método	HLP_OutgoingCall:SetCallID(callI D)	Establece el identificador de una llamada telefónica.	13
método	HLP_Address:CloseLineHandler( ): value: long	Libera el handler asociado con una dirección.	14
función TAPI	lineClose(addressHandler: HLINE)	Cierra el handler addressHandler. El resultado de la ejecución de la función es value.	14.1
método	HLP_Address:OpenLineHandler(appHandler:HLINEAPP, lineID, privileges: unsigned long, mediaModes: unsigned long, callParams: LINECALLPARAMS): value: long	Obtiene un handler para la dirección especificada en callParams y que está asociada con el dispositivo de línea con identificador igual a lineID. El handler tiene los privilegios establecidos en privileges y para utilizar los media modes indicados en mediaModes.	15
función TAPI	lineOpen(appHandler: HLINEAPP, lineID, addressHandler: HLINE, APIVersion:unsigned long, extVersion: unsigned long, callbackInstance: unsigned long, privileges: unsigned long, mediaModes: unsigned long, callParams: LINECALLPARAMS) addressHandler: HLINE, value: long	Abre la dirección indicada en CallParams asociada con el dispositivo de línea especificado por su identificador lineID y regresa el handler addressHandler para cualquier uso posterior antes de cerrarlo. El parámetro privileges especifica los privilegios que se desean para el dispositivo de línea y mediaModes los media modes que se desean utilizar con el dispositivo de línea. El éxito o fracaso de esta operación se indica en value.	15.1

función TAPI	lineSetStatusMessages(addressHandler: HLINE, lineMessages: unsigned long, addressMessages: unsigned long): value: long	Habilita al Control Administrativo para especificar cuáles mensajes de estado desea que se le notifiquen para el handler addressHandler. El éxito o fracaso de esta operación se indica en value.	15.2
método	HLP_OutgoingCall:Connect(destNumber, countryCode, addressHandler: HLINE): value: long	Realiza la marcación del número telefónico del abonado solicitado.	16
función TAPI	lineMakeCall(addressHandler: HLINE, destNumber, countryCode, callParams: LINECALLPARAMS): replyID: long, callHandler: HCALL, value: long	Realiza una llamada saliente en el dispositivo de línea y dirección relacionados con el handler AddressHandler. El número del abonado solicitado es destNumber, el código del país del número es countryCode. El handler correspondiente a la llamada telefónica es callHandler. El identificador de la respuesta pendiente o en su defecto el error ocurrido durante la ejecución de la función es replyID.	16.1
método	HLP_ReplyArray:Insert(reply: HLP_Reply)	Inserta un objeto reply en el contenedor de respuestas pendientes.	16.3
método	HLP_CallArray : Insert(outgoing: HLP_OutgoingCall)	Inserta una llamada telefónica en el contenedor de llamadas.	17

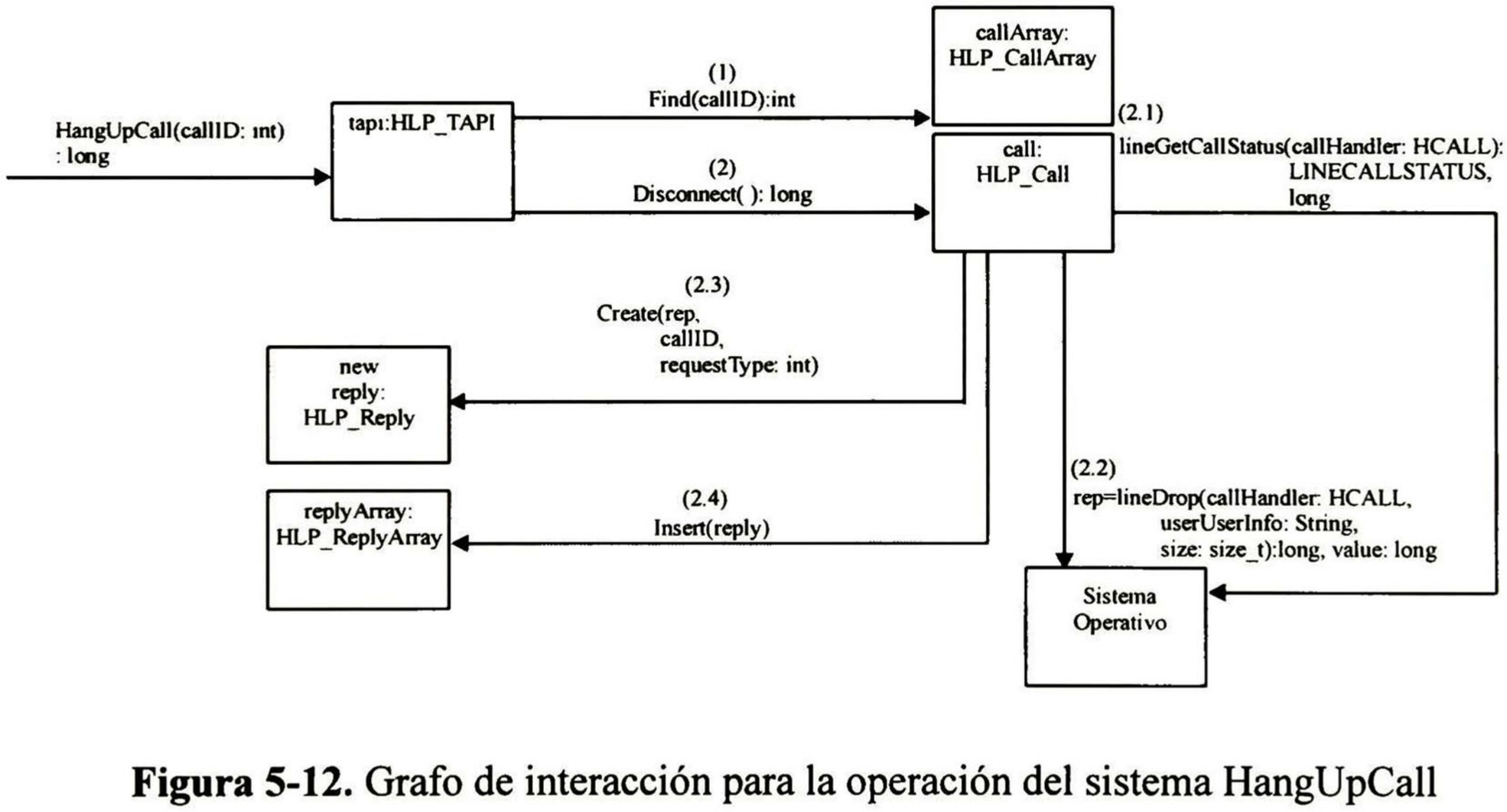


Figura 5-12. Grafo de interacción para la operación del sistema HangUpCall

Descripción:

Tipo	Nombre	Descripción	No. de Sec.
operación del sistema	HLP_TAPI:HangUpCall(callID: int): long	Termina una llamada telefónica.	

método	HLP_CallArray:Find(callID): position: int	Encuentra en el contenedor de llamadas telefónicas la posición donde se localiza la llamada con identificador igual a callID.	1
método	HLP_Call : Disconnect( ): value: long	Termina una llamada telefónica.	2
función TAPI	lineGetCallStatus(callHandler: HCALL): callStatus: LINECALLSTATUS, value: long	Obtiene información de estado de la llamada telefónica relacionada con el handler callHandler. La información correspondiente se localiza en la estructura callStatus y el resultado de la ejecución de la función es value.	2.1
función TAPI	lineDrop(callHandler: HCALL, userUserInfo: String, size: size_t): reply: long, value: long	Termina la llamada telefónica relacionada con el handler callHandler. El identificador de la respuesta pendiente o en su defecto el error al ejecutar la función es value.	2.2
método	HLP_ReplyArray : Insert(reply)	Inserta un objeto reply en el contenedor de respuestas pendientes.	2.4

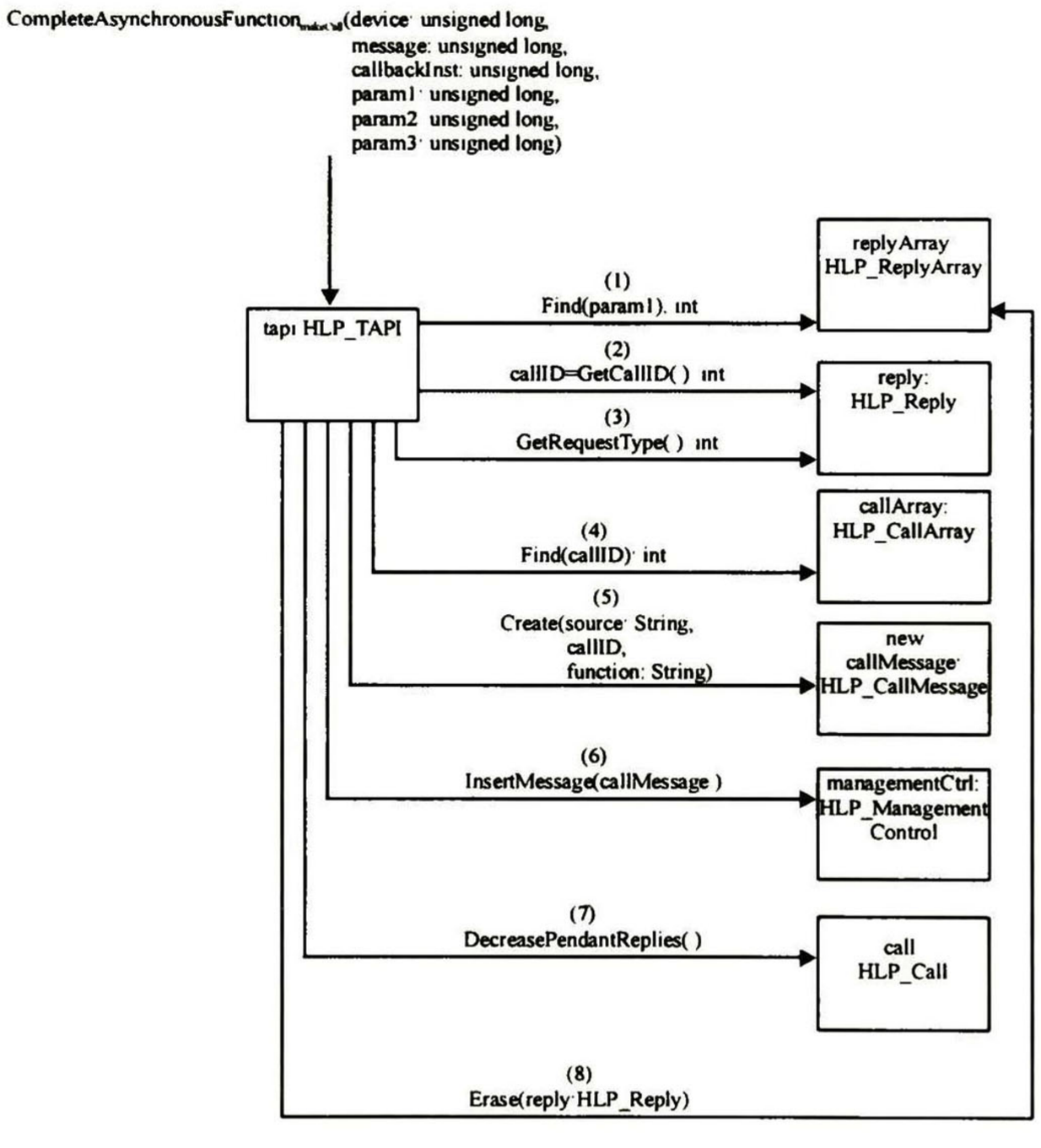


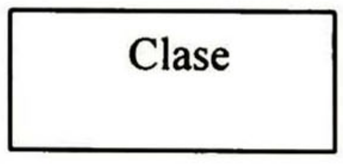
Figura 5-13. Grafo de interacción para la operación del sistema $CompleteAsynchronousFunction_{makeCall}$

Descripción:

Tipo	Nombre	Descripción	No. de Sec.
operación del sistema	HLP_TAPI: CompleteAsynchronousFunction$_{makeCall}$(device: unsigned long, message: unsigned long, callbackInst: unsigned long, param1: unsigned long, param2: unsigned long, param3: unsigned long)	Consumación de la función asíncrona para realizar una llamada telefónica.	
método	HLP_ReplyArray:Find(param1): position: int	Localiza el objeto reply con identificador igual a param1 en el contenedor de respuestas pendientes.	1
método	HLP_Reply:GetCallID( ): callID: int	Obtiene el identificador de la llamada asociada con un objeto reply.	2
método	HLP_Reply:GetRequestType( ): requestType: int	Obtiene el tipo de petición que generó un objeto reply.	3
método	HLP_CallArray : Find(callID: int): position: int	Encuentra en el contenedor de llamadas telefónicas la posición donde se localiza la llamada con identificador igual a callID.	4
método	HLP_ManagementControl: InsertMessage(callMessage: HLP_CallMessage)	Inserta un mensaje con información relativa a una llamada telefónica en la cola de mensajes del Control Administrativo.	6
método	HLP_Call:DecreasePendantReplies( )	Decrementa el número de respuestas asociados con una llamada telefónica.	7
método	HLP_ReplyArray:Erase(reply: HLP_Reply)	Elimina un objeto reply del contenedor de respuestas pendientes.	8

5.3.2.2 Grafos de visibilidad

Un objeto debe tener acceso a otro objeto si se requiere que exista alguna comunicación entre ellos. El referenciamiento es el mecanismo definido por Fusion que permite el paso de mensajes entre objetos. Los grafos de visibilidad describen gráficamente la estructura de referencias para cada una de las clases del sistema. Una referencia tiene asociadas cuatro propiedades: tiempo de vida de la referencia, visibilidad del objeto servidor, ligadura del objeto servidor y mutabilidad de la referencia. Los grafos de visibilidad se construyen a partir de los diagramas de interacción entre objetos, ya que en éstos se define la comunicación requerida entre objetos. La Figura 5-14 muestra la notación para los grafos de visibilidad.

Nombre	Representación
Clase Cliente	

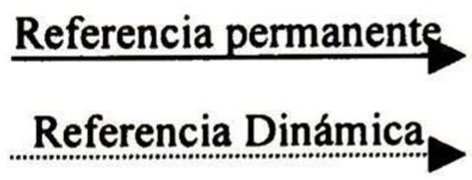
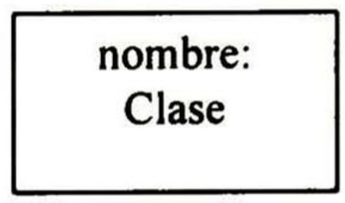
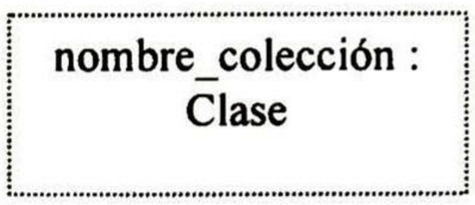
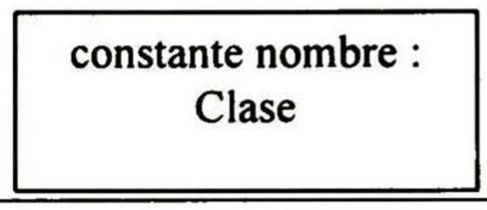
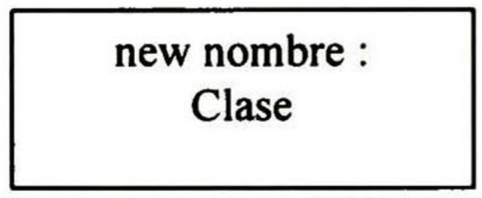
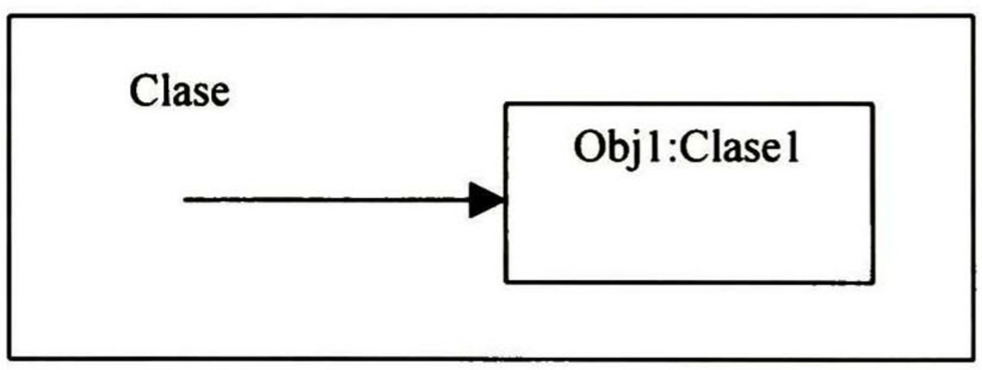
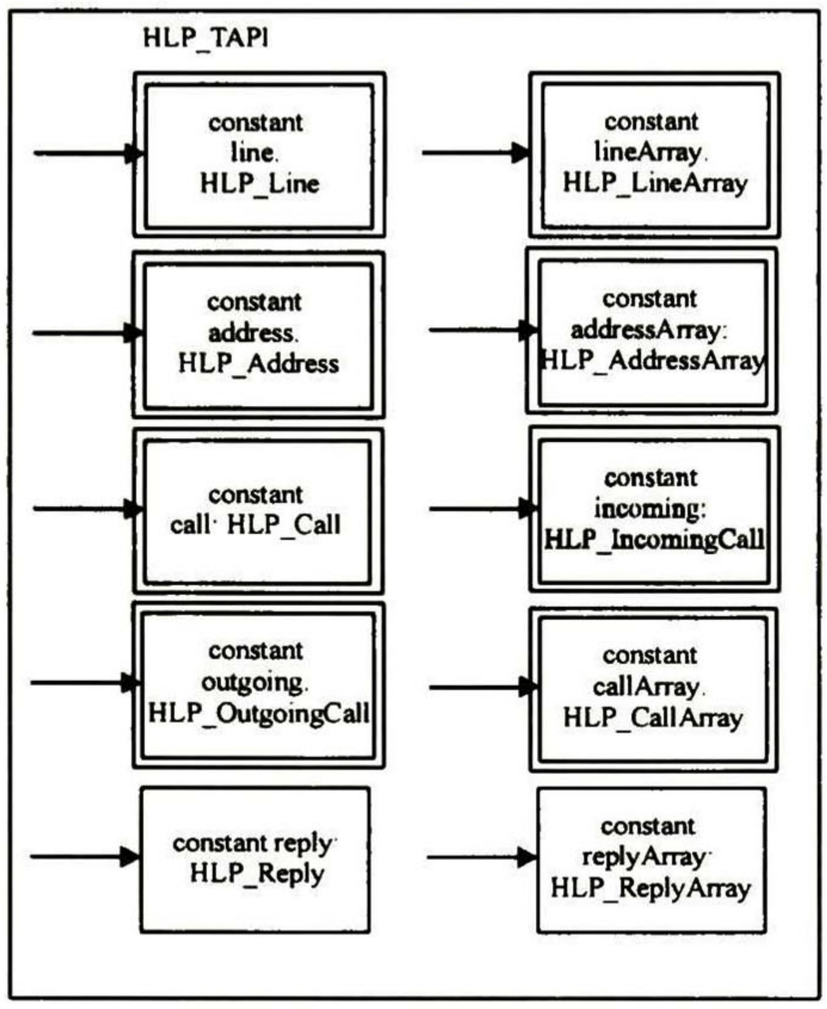
Visibilidad de Referencias	
Objeto Servidor	
Colección de Objetos Servidores	
Objeto Servidor Constante	
Objeto Creado Dinámicamente	
Objeto Servidor cuya vida está desligada de la vida de la clase Cliente	
Objeto Servidor cuya vida está ligada a la vida de la clase Cliente	
Referencia Exclusiva a un Objeto Servidor	
Referencia Exclusiva a una Colección de Objetos Servidores	

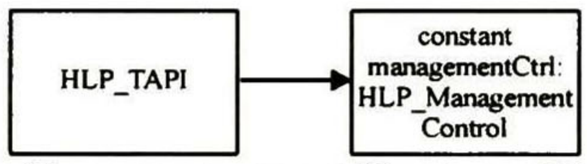
Figura 5-14. Notación para los grafos de visibilidad

La Figura 5-15 muestra el grafo de visibilidad para la clase HLP_TAPI. En él se aprecia que la mayoría de los objetos del sistema son accedidos exclusivamente por esta clase, que su tiempo de vida está ligado al de la clase HLP_TAPI y que las referencias son permanentes y constantes. Los objetos reply y replyArray son compartidos con la clase

HLP_Call y sus subclases (ver Figura 5-16, Figura 5-17 y Figura 5-18). El objeto que representa al Control Administrativo de Llamadas es compartido con otros objetos que están fuera del alcance de este sistema y su tiempo de vida está desligado del de HLP_TAPI.

Cabe aclarar que las únicas referencias directas de la clase HLP_TAPI son aquellas correspondientes a los objetos lineArray, callArray y replyArray. Los objetos line, address y reply son accedidos mediante sus respectivos contenedores. Los objetos addressArray son





accedidos a través de los objetos line correspondientes. Sin embargo, dado que los grafos de visibilidad se construyen a partir de los diagramas de interacción entre objetos, estas referencias indirectas son incluidas en la Figura 5-15.

Figura 5-15. Grafo de visibilidad para la clase HLP_TAPI

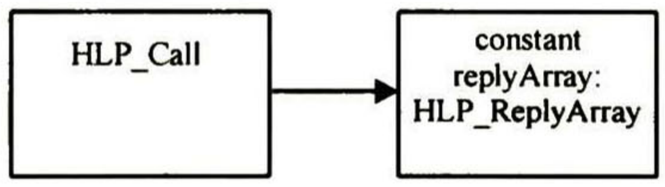


Figura 5-16. Grafo de visibilidad para la clase HLP_Call

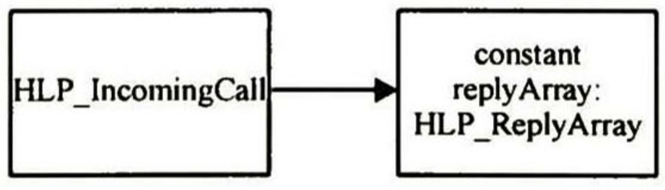


Figura 5-17. Grafo de visibilidad para la clase HLP_IncomingCall

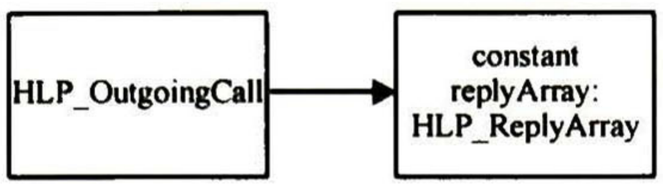


Figura 5-18. Grafo de visibilidad para la clase HLP_OutgoingCall

5.3.2.3 Descripciones de clases

Las descripciones de clases proveen la especificación de cada clase del sistema: sus atributos, sus métodos y su posición en la jerarquía de herencia. Esta información se recolecta de los modelos restantes del diseño y del diccionario de datos (ver [PAD00]) de la siguiente manera: del diccionario de datos se obtienen los atributos; de los diagramas de interacción entre objetos los métodos de la interfaz; de los grafos de visibilidad las características de las referencias y de los grafos de herencia las superclases (en caso de existan).

En esta sección se muestran únicamente las descripciones de clases para HLP_TAPI, HLP_LineArray, HLP_Call y HLP_IncomingCall. El resto puede ser consultado en [PAD00].

clase
HLP_TAPI
atributo managementCtrl: **constante, compartida, no ligada** HLP_ManagementControl
atributo appHandler: HLINEAPP
atributo appName: String
atributo numDevices: unsigned long
atributo APIVersion: unsigned long
atributo lineArray: **constante, exclusiva, ligada** HLP_LineArray
atributo callArray: **constante, exclusiva, ligada** HLP_CallArray
atributo replyArray: **constante, compartida, ligada** HLP_ReplyArray
atributo shuttingDown: bool
atributo initializing: bool
atributo lineInitializeExParams: LPLINEINITIALIZEEXPARAMS
método HLP_TAPI(MgeCtrl: HLP_ManagementControl,
hInstance: HINSTANCE, sAppName: String)
método InitializeTAPI( ): long
método EnumerateResources( ): long
método static void CALLBACK CallbackFunction(Device: unsigned long,
Msg: unsigned long,
CallbackInstance: unsigned long,
Param1: unsigned long,
Param2: unsigned long,
Param3: unsigned long)
método UpdateCallState(Device: unsigned long,
Msg: unsigned long,
CallbackInstance: unsigned long,
Param1: unsigned long,
Param2: unsigned long,
Param3: unsigned long)

método UpdateLineState(Device: unsigned long,
Msg: unsigned long,
CallbackInstance: unsigned long,
Param1: unsigned long,
Param2: unsigned long,
Param3: unsigned long)
método UpdateAddressState(Device: unsigned long,
Msg: unsigned long,
CallbackInstance: unsigned long,
Param1: unsigned long,
Param2: unsigned long,
Param3: unsigned long)
método GetTapiAppHandler( ): LPHLINEAPP
método GetAPIVersion( ): unsigned long
método GetNumDevs( ): unsigned long
método GetReplyArray( ): HLP_ReplyArray
método GetCallArray( ): HLP_CallArray
método GetLineArray( ): HLP_LineArray
método ShutdownTAPI( ): long
método ~HLP_TAPI( )
método CreateIncomingCall(hCall: HCALL): long
método CreateOutgoingCall(hCall: HCALL): long
método GetControl( ): HLP_ManagementControl
método MakeCall(DestNumber: String,
CountryCode: unsigned long,
CallID: int,
LineID: unsigned long,
AddressID: unsigned long): long
método DialNumber(Digits: String,
CountryCode: unsigned long,
CallID: int): long
método AcceptIncomingCall(CallID: int): long
método HangUpCall(CallID: int): long
método PrepareLine(LineToOpen: unsigned long,
AddressToOpen: unsigned long): long
método CloseLines( ): long
finclase

clase
HLP_Line
atributo numAddresses: unsigned long
atributo bearerModes: unsigned long
atributo mediaModes: unsigned long
atributo lineFeatures: unsigned long
atributo deviceID: unsigned long
atributo handler: HLINE
atributo addressArray: HLP_AddressArray
atributo APIVersion: unsigned long
atributo lowAPIVersion: unsigned long
atributo highAPIVersion: unsigned long
atributo lineExtensionID: LINEEXTENSIONID
atributo lineDevCaps: LPLINEDEVCAPS
atributo lineDevStatus: LPLINEDEVSTATUS
atributo privileges: unsigned long
atributo status: unsigned long

método HLP_Line( )
método NegotiateVersion(hApp: HLINEAPP): long
método GetDevCaps(hApp: HLINEAPP): long
método Open(hApp: HLINEAPP,
deviceID: unsigned long,
privileges: unsigned long,
mediaModes: unsigned long,
callParams: LPLINECALLPARAMS): long
método Close( ): long
método SetStatusMessages( ): long
método SetStatus( ): long
método GetStatus( ): long
método GetLineHandler( ): LPHLINE
método GetID( ): unsigned long
método GetNumAddresses( ): unsigned long
método GetBearerModes( ): unsigned long
método GetMediaModes( ): unsigned long
método GetLineFeatures( ): unsigned long
método GetAddressArray( ): unsigned long
método GetAPIVersion( ): unsigned long
método ~HLP_Line( )
finclase

clase
HLP_Call
atributo addressHandler: HLINE
atributo associatedLine: unsigned long
atributo addressID: unsigned long
atributo origin: unsigned long
atributo countryCode: unsigned long
atributo ID: int
atributo hCall: HCALL
atributo callParams: LPLINECALLPARAMS
atributo callStatus: LPLINECALLSTATUS
atributo callInfo: LPLINECALLINFO
atributo callerID: String
atributo calledID: String
atributo pendantReplies: int
atributo stoppingCall: bool
atributo mngmntCtrlNotified: bool
atributo replyArray: **constante, compartida, no ligada** HLP_ReplyArray
atributo status: unsigned long
método HLP_Call( )
método virtual Connect(DestNumber: String,
CountryCode: unsigned long,
AddressHandler: HLINE): long
método Drop( ): long
método Deallocate( ): long
método Disconnect( ): long
método GetCallID( ): int
método SetCallID(CallID: int)
método GetCallerID( ): String
método GetCalledID( ): String
método GetOrigin( ): unsigned long
método GetState( ): unsigned long

método SetState( ): long
método GetCallHandler( ): LPHCALL
método GetAssociatedLine( ): unsigned long
método GetAssociatedAddress( ): unsigned long
método GetCallInfo( ): long
método GetStopFlag( ): bool
método SetStopFlag(Value: unsigned long): long
método GetMngmntCtrlFlag( ): bool
método SetMngmntCtrlFlag(Value: bool)
método IncreasePendantReplies( )
método DecreasePendantReplies( )
método ~HLP_Call( )
finclase

Clase
HLP_IncomingCall es una HLP_Call
método HLP_IncomingCall(hCall: HCALL,
ReplyArray: HLP_ReplyArray)
método Accept( ): long
método Answer( ): long
método Connect(DestNumber: String,
CountryCode: String,
AddressHandler: HLINE): long
método ~HLP_IncomingCall( )
finclase

5.3.2.4 Grafos de Herencia

Los grafos de herencia presentan las estructuras de herencia para el sistema. La herencia es el mecanismo que permite definir una clase como una especialización de otra. El proceso para determinar las relaciones de herencia se basa en:

- Las estructuras de generalización y especialización incluidas en el modelo objeto del sistema, desarrollado durante el análisis; y
- El análisis de las clases para identificar abstracciones comunes, en la etapa de diseño.

En nuestro caso, el modelo objeto que describe la estructura general de HTT (ver [PAD00]) involucra una estructura de especialización referente a las llamadas telefónicas. Esta especialización indica que hay dos tipos: las entrantes y las salientes, las cuales son subtipos del concepto general de llamada telefónica. Esta a su vez constituye el supertipo. El triángulo relleno indica que una llamada telefónica debe ser necesariamente entrante o saliente.

El grafo de herencia para las clases representativas de las llamadas telefónicas se presenta en la Figura 5-19. La clase HLP_Call constituye la superclase y las clases HLP_IncomingCall y HLP_OutgoingCall se denominan subclases. El triángulo relleno significa que cualquier objeto que represente una llamada telefónica debe ser una instancia de alguna de las subclases.

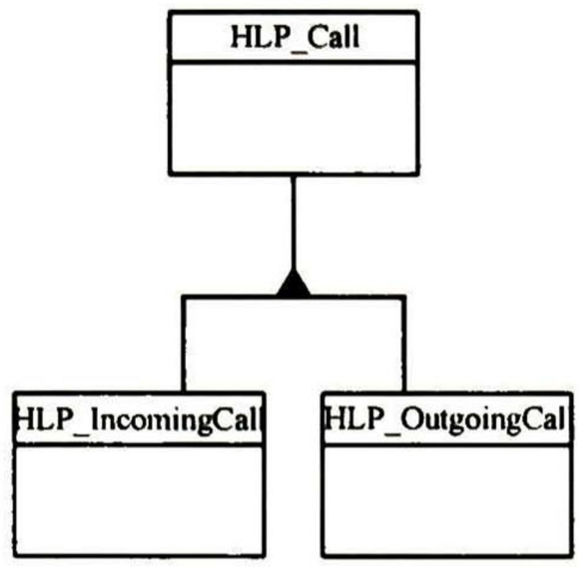


Figura 5-19. Clase HLP_Call y sus subclases

5.3.3 Implementación

La fase final de Fusion es transformar el diseño en una implementación en algún lenguage de programación. La semántica de las clases se extrae de los grafos de interacción entre objetos y de los grafos de visibilidad. La implementación de las clases definidas durante el diseño se realizó en lenguaje C++.

5.3.4 Reuso de HTT

Como resultado del trabajo de análisis y diseño, sabemos que el control y la coordinación de las clases que constituyen HTT lo realiza la clase controladora, denominada HLP_TAPI. La interfaz de esta clase provee un conjunto de métodos mediante los cuales se accede a los servicios de telefonía descritos en los capítulos anteriores. Por lo tanto, cualquier aplicación o habilitador CTI de más alto nivel que requiera el uso de tales servicios debe crear una instancia de la clase HLP_TAPI. Una vez que se ha creado la instancia deben realizarse las siguientes acciones:

- *Habilitar el uso de los dispositivos de línea*

  Mediante la invocación del método HLP_TAPI::InitializeTAPI.

- *Obtener la información necesaria para hacer uso de los dispositivos de línea y las direcciones*

  A través de la invocación al método HLP_TAPI::EnumerateResources.

- *Configurar los dispositivos de línea y las direcciones para habilitar la notificación de llamadas entrantes (en caso de ser necesario)*

  Con una invocación al método HLP_TAPI::PrepareLine.

Una vez que se han ejecutado exitosamente los métodos anteriores puede invocarse el método que proporcione el servicio deseado con los parámetros indicados. HTT asume que las aplicaciones tienen las siguientes responsabilidades:

- *Derivación y redefinición de métodos de la clase abstracta HLP_ManagementControl*

  La clase HLP_ManagementControl es la interfaz mediante la cual HTT notifica a las aplicaciones de los eventos telefónicos y los resultados de la ejecución de funciones asíncronas. Por lo tanto, las aplicaciones que utilicen HTT deberán crear una clase derivada de HLP_ManagementControl y redefinir sus métodos para poder interactuar con HTT. La especificación de la clase HLP_ManagementControl puede ser consultada en [CAR00].

- *Asignación de identificadores únicos a las llamadas telefónicas*

  Uno de los parámetros requeridos en todos los métodos que ejecutan operaciones sobre llamadas telefónicas es un identificador de llamada telefónica. El habilitador CTI supone que la aplicación cliente lleva un control para la asignación de identificadores únicos para cada llamada telefónica. Estos identificadores únicos son utilizados por el habilitador para distinguir los distintos objetos de tipo llamada telefónica.

- *Conocimiento de los dispositivos de línea disponibles y de sus direcciones respectivas*

  Con el fin de tener la opción de selección de la dirección por la cual se desea realizar llamadas salientes.

Finalmente, cuando las aplicaciones ya no requieran hacer uso de los servicios telefónicos o antes de terminar su ejecución deberán realizar los siguientes pasos:

- *Liberar todos los dispositivos de línea y las direcciones*

  Mediante la invocación del método HLP_TAPI::CloseLines.

- *Deshabilitar el uso de los dispositivos de línea*

  Con el método HLP_TAPI::ShutdownTAPI.

6 Conclusiones y Trabajo Futuro

6.1 Conclusiones

En esta tesis se presentó el desarrollo de una arquitectura orientada a objetos de un habilitador CTI basado en TAPI. Éste se denomina Habilitador Telefónico basado en TAPI (HTT). El análisis y diseño de la arquitectura se llevó a cabo utilizando el método Fusion para desarrollo de software orientado a objetos. La implementación se realizó en lenguaje C++ y proporciona la funcionalidad correspondiente al nivel de servicios básicos de telefonía. Dicha implementación trabaja adecuadamente con el proveedor de servicios de telefonía y con el control administrativo de llamadas telefónicas del sistema Asistente Telefónico.

HTT encapsula el manejo de los dispositivos de línea, las direcciones y las respuestas pendientes de funciones asíncronas definidos en TAPI. Además proporciona a las aplicaciones o habilitadores CTI de más alto nivel una interfaz sencilla para el acceso a las operaciones básicas sobre llamadas telefónicas entrantes y salientes.

Como resultado del estudio de las distintas versiones de TAPI – 1.3, 1.4, 2.0 y 2.1 – se identificaron las dificultades siguientes: TAPI no proporciona el soporte para la concurrencia de llamadas telefónicas; no maneja el asincronismo de algunas de sus funciones; la realización de operaciones telefónicas sencillas requiere de la ejecución de múltiples funciones y su enfoque orientado a funciones dificulta el diseño de las aplicaciones que hacen uso de esta interfaz.

La arquitectura propuesta resuelve los siguientes problemas:

- *Manejo de funciones asíncronas*

  HTT se encarga de manejar las funciones asíncronas de TAPI mediante un conjunto de objetos, con lo cual se libera de esta responsabilidad a las aplicaciones.

- *Soporte para el proceso de múltiples llamadas de manera simultánea*

  TAPI por sí misma no provee el control necesario para permitir la concurrencia de llamadas telefónicas. La arquitectura de HTT proporciona este soporte a través de la clase controladora y el contenedor de llamadas telefónicas.

- *Reducción del número de invocaciones a funciones para la realización de operaciones telefónicas*

  Las aplicaciones cliente de TAPI requieren normalmente de la invocación de siete funciones para realizar una llamada saliente. La terminación de una llamada telefónica involucra de tres a seis invocaciones de funciones. Y lo mismo sucede con otras

operaciones telefónicas sencillas. HTT proporciona el acceso a cada una de estas operaciones mediante la invocación de un solo método de la clase controladora.

- *Diseño orientado a objetos vs. funciones*

  HTT fue desarrollado utilizando un método orientado a objetos para lograr una arquitectura fácilmente extensible. Dado que los objetos ocultan su representación interna, es posible cambiar la implementación de un objeto sin afectar la comunicación con sus clientes.

6.2 Trabajo Futuro

El diseño actual del habilitador contempla únicamente el manejo de los dispositivos de línea y las direcciones que están conectadas directamente a la computadora en la cual reside el código del habilitador. Sin embargo, la versión 2.1 de TAPI provee los medios para acceder a los dispositivos telefónicos remotos. Como trabajo futuro puede realizarse investigación en las siguientes áreas:

1. Extender el habilitador CTI para proporcionar acceso a dispositivos telefónicos remotos
2. Implementar nuevos servicios tales como conferencias telefónicas y "call forwarding"
3. Mejorar la reusabilidad del habilitador creando componentes a partir de las clases diseñadas

7 Referencias

[BEC89] K. Beck and W. Cunningham, A laboratory for teaching object-oriented thinking, ACM OOPSLA ´89 Conference Proceedings, 1989.

[BHA98] A. Bhattacherjee & J. Gerlach, Understanding and Managing OOT Adoption, IEEE Software, Vol. 15, Num.3, May/Jun 1998.

[BOO91] G. Booch, Object-Oriented Design with Applications, Benjamin Cummings, Redwood City, CA. 1991.

[CAR00] A. Carranza, Análisis y diseño de una aplicación CTI, Tesis en desarrollo.

[COL94] D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes and P Jeremaes, Object-Oriented Development, The Fusion Method, PrenticeHall, Englewood Cliffs, NJ. 1994.

[COL95] D. Coleman, Fusion with Use Cases – Extending Fusion For Requirements Modelling, HP Labs 1995, dc@hplsrd.hpl.hp.com.

[EDG97] B. Edgar, PC Telephony, Parity Software Development Corporation, 1997.

[FIG99] M. Figueroa, Acerca de los VxDs, Documento Técnico, 1999.

[FIG00] M. Figueroa, Diseño de una tarjeta telefónica para bus ISA, Tesis en desarrollo.

[FRA97] R. France, J.M. Bruel, M. Larrondo-Petrie, E. Grant & M. Saksena, Toward a Rigorous Object-Oriented Analysis and Design Method, IEEE Computer Society Press, 1st. IEEE International Conference on Formal Engineering Methods (ICFEM '97), Hiroshima, Japan 12-14 November 1997.

[IEEE830] Institute of Electrical and Electronics Engineers, IEEE Std. 830-1998 IEEE Recommended Practice for Software Requirements Specifications, 1998.

[HEL99] D. Torres, A. Carranza, M. Figueroa, R. Padilla y L. Ximénez, Especificación Técnica del Asistente Telefónico, 1999.

[ITUG.711] ITU-T Rec. G.711 (1993), Modulación por Impulsos Codificados (MIC) de Frecuencias Vocales.

[ITUH.323] ITU-T Rec. H.323 (1998) Sistemas de Comunicación Multimedios Basados en Paquetes.

[ITUH.231] ITU-T H.231 Rec. (07/97) Multipoint control units for audiovisual systems using digital channels up to 1920 kbit/s.

[ITUQ.1200] ITU-T Q.1200 Rec. (09/97), General series Intelligent Network Recommendation structure.

[JAC97] J. Jacobs, Deploying Microsoft TAPI 2.1, Microsoft TechNet Aug. 1997.

[JON90] C. B. Jones, Systematic Software Development Using VDM, 2nd ed. Prentice Hall International, Englewood Cliffs, NJ 1990.

[KRI99] M.S. Krishnan & M. I. Kellner, Measuring Process Consistency: Implications for Reducing Software Defects, Transactions on Software Engineering, Vol. 25, Num. 6, Nov-Dec 99.

[MCL98] S. G. McLellan, A. W. Roesler, J. T. Tempest & C.L. Spinuzzi, Building more usable APIs, IEEE Software, Vol. 15, Num. 3, May/Jun 1998.

[MAY99] K. Mayer, Why not CTI?, CTI Magazine, Nov. 1999, http://www.tmcnet.com/articles/ctimag/1199/1199edout.htm.

[MIC97A] Microsoft Corp. & Intel Corp., Telephony Application Programming Interface Programmer's Reference, 1997.

[MIC97B] Microsoft Corp., Telephony Service Provider Interface Programmer's Reference, 1997.

[MIC99A] Microsoft Corp., Media Service Provider Interface (MSPI) Reference, 1999, http://msdn.microsoft.com/library/psdk/tapi3/msp_5x9h.htm.

[MIC99B] Microsoft Corp., IP Telephony with TAPI 3.0, 1999, http://msdn.microsoft.com/library/psdk/tapi3/ip_app1_4434.htm.

[MIC99C] Microsoft Corp., Media Control Interface, 1999, http://msdn.microsoft.com/library/winresource/dnwin95/S714A.htm.

[MIR95] C. Mirho, To Learn About the Voice Modem Extensions for Windows95, Press 1 Now!, Microsoft Systems Journal, 1995.

[MIR96] C. Mirho and A. Terrise, Communications Programming for Windows 95, Microsoft Press, 1996.

[NIX96] T. Nixon, Windows Telephony (TAPI) support in Windows NT 4.0, http://www.microsoft.com/win32/dev/netwrk/tapiwp.htm, Junio 1996.

[NOV99] Novell Inc., Netware Telephony Services 2.21 Products Brief, http://www.novell.com/catalog/qr/sne24310.html.

[ORE96] O´Reilly & Associates, http://www.ora.com/reference/dictionary/terms/D/Direct_Inward_Dialing.htm, 1996.

[PAD00] R. Padilla, Análisis y Diseño en Fusion del Habilitador Telefónico basado en TAPI (HTT), 2000.

[POL99] C. Polyzois, K. Purdy, P. Yang, D. Shrader, H. Sinnreich, F. Ménard and H. Schulzrinne, From POTS to PANS: A Commentary on the Evolution to Internet Telephony, IEEE Internet Computing, May-Jun 1999.

[RIZ99] D. Rizzetto and C. Catania, A Voice over IP Service Architecture for Integrated Communications, IEEE Internet Computing, May-Jun 1999.

[RUM88] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, Object-Oriented Modeling and Design, Prentice Hall International, Englewood Cliffs, NJ. 1988.

[SHA96] M. Shaw & D. Garlan, Software Architecture, Perspectives on an Emerging Discipline, Prentice Hall, 1996.

[TEC99A] TechTarget.com Inc., www.whatis.com/acd.htm, 1999.

[TEC99B] TechTarget.com Inc., www.whatis.com/did.htm, 1999.

[UNI00] Referencia sobre UNICODE, http://unicode.org.

[VAR99] D. Varney, In Migration: Circuit-Switched Intelligence Takes a Ride on a Packet, CTI Magazine, Dec. 1999, http://www.tmcnet.com/articles/ctimag/1299/1299lucent.htm.

[WAL97] R. Walters, CTI in Action, Ed. Wiley & Sons 1997.

[XIM00] L. Ximénez, Especificación Formal de Facilidades Telefónicas, Tesis en Desarrollo.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: "Análisis, diseño e implementación de un habilitador CTI basado en TAPI" de la Lic. Rosa Gabriela Padilla Gutiérrez el día 16 de Junio de 2000.

Dr. Deni Librado Torres Román
Investigador Cinvestav 2 C
CINVESTAV DEL IPN
Guadalajara.

Dr. Félix Francisco Ramos Corchado
Investigador Cinvestav 2 A
CINVESTAV DEL IPN
Guadalajara.

Dr. Ricardo Raúl Jacinto Montes
Director
Diseño y Desarrollo Tecnológico S.C.
Guadalajara, Jal.