4.2 Modelo conceptual

El modelo conceptual es la representación de conceptos (objetos) significativos en el dominio del proyecto. Para facilitar su lectura se ha dividido en dos partes que contienen los siguientes contenidos:

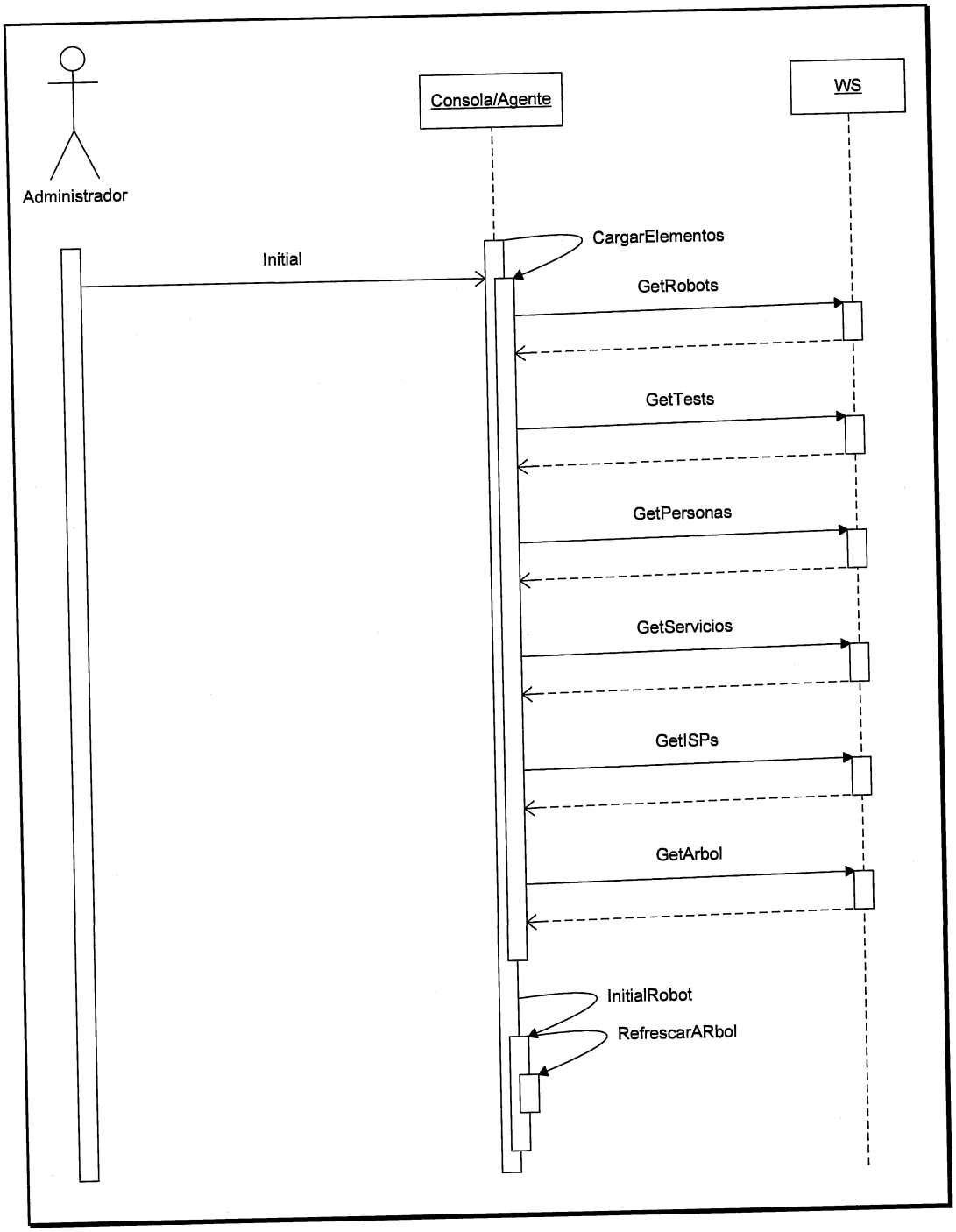
4.3 Modelo de comportamiento del sistema

4.3.1 Diagramas de Secuencias

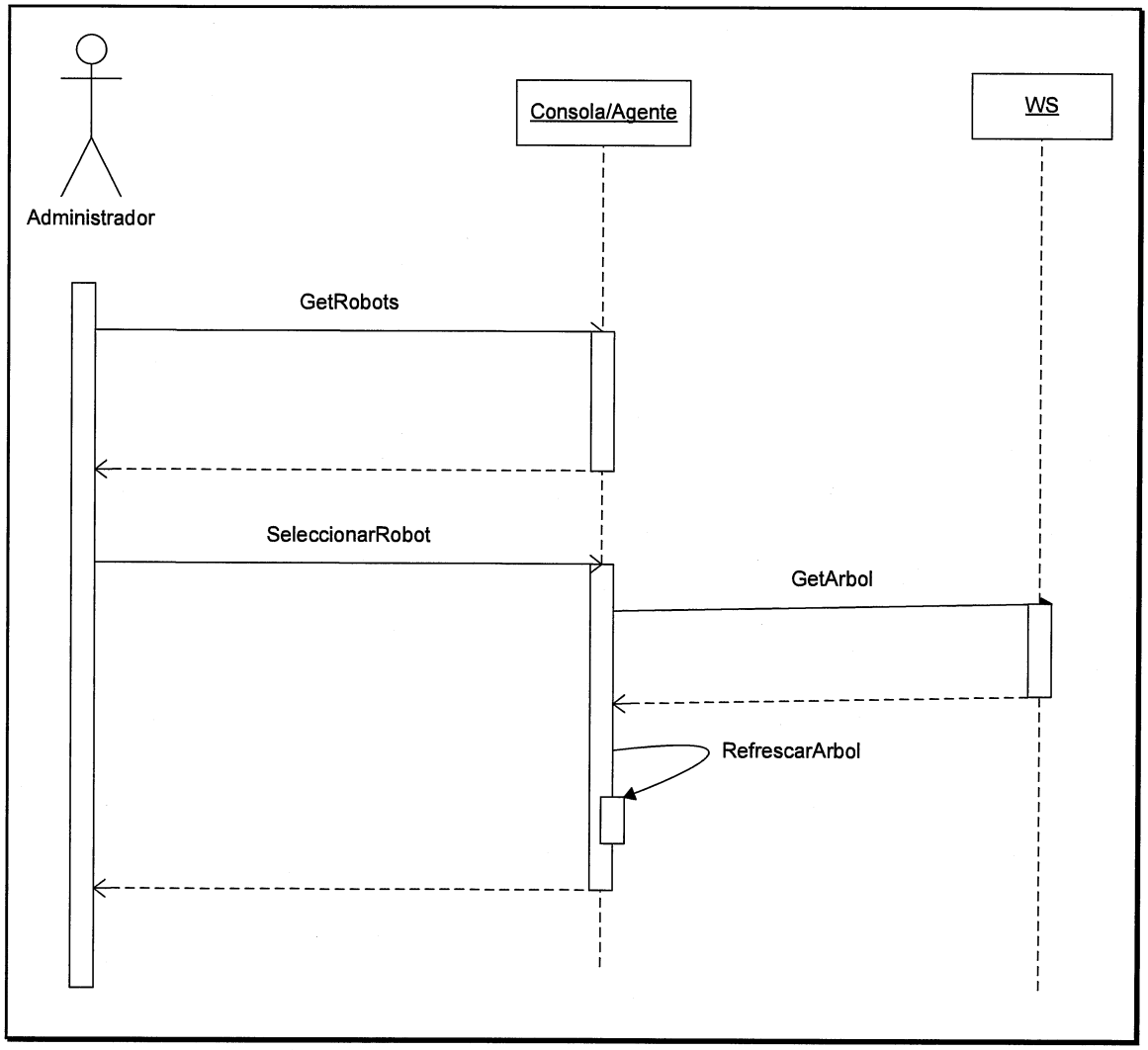
Los diagramas de secuencia tienen como objetivo mostrar la secuencia de acontecimientos entre los actores y el sistema, permitiendo identificar las operaciones de éste.

Se muestran a continuación los diagramas de secuencia de los diferentes casos de uso anteriormente indicados.

4.3.1.1 Puesta en Funcionamiento

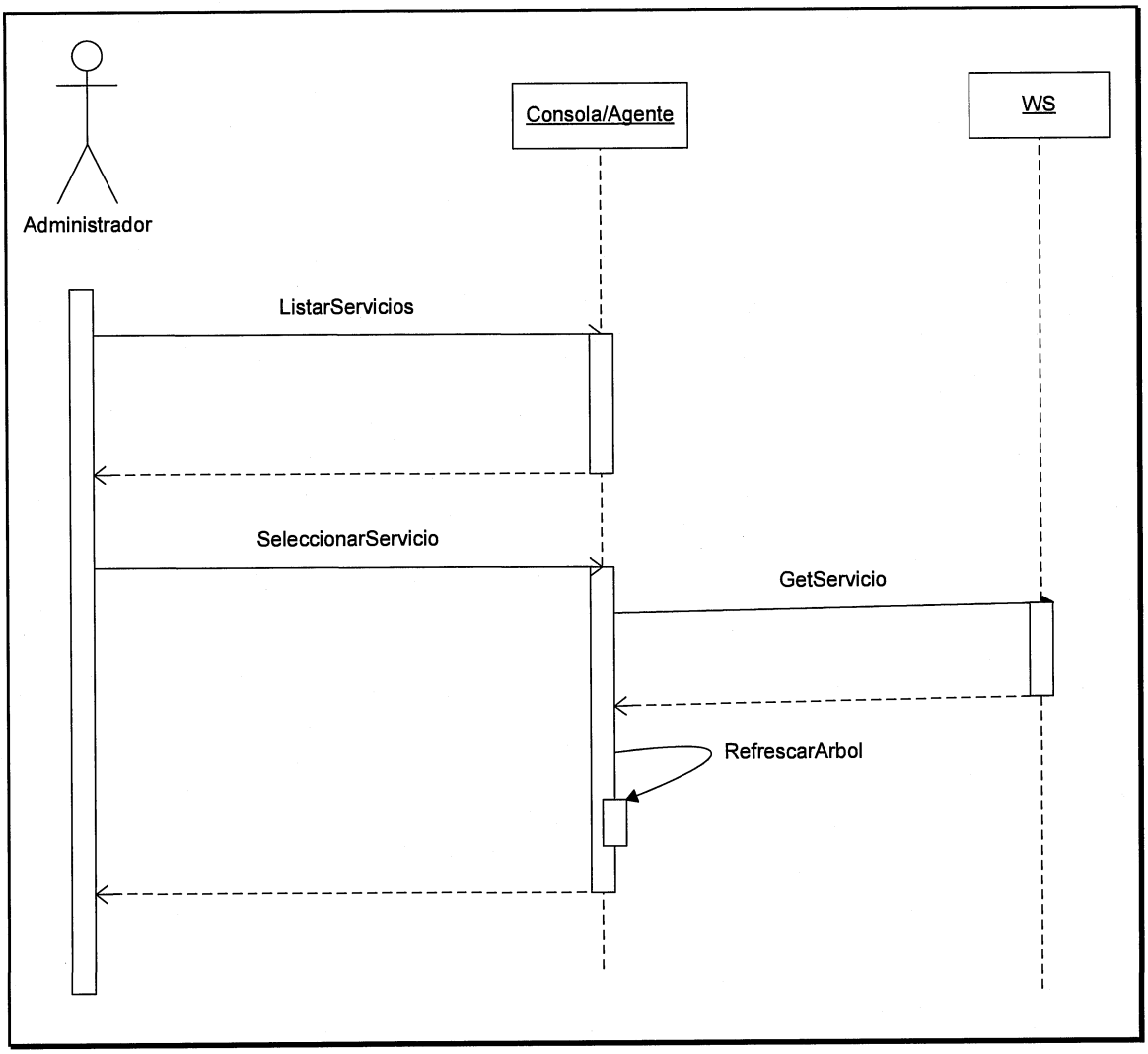


4.3.1.2 Selección Robot

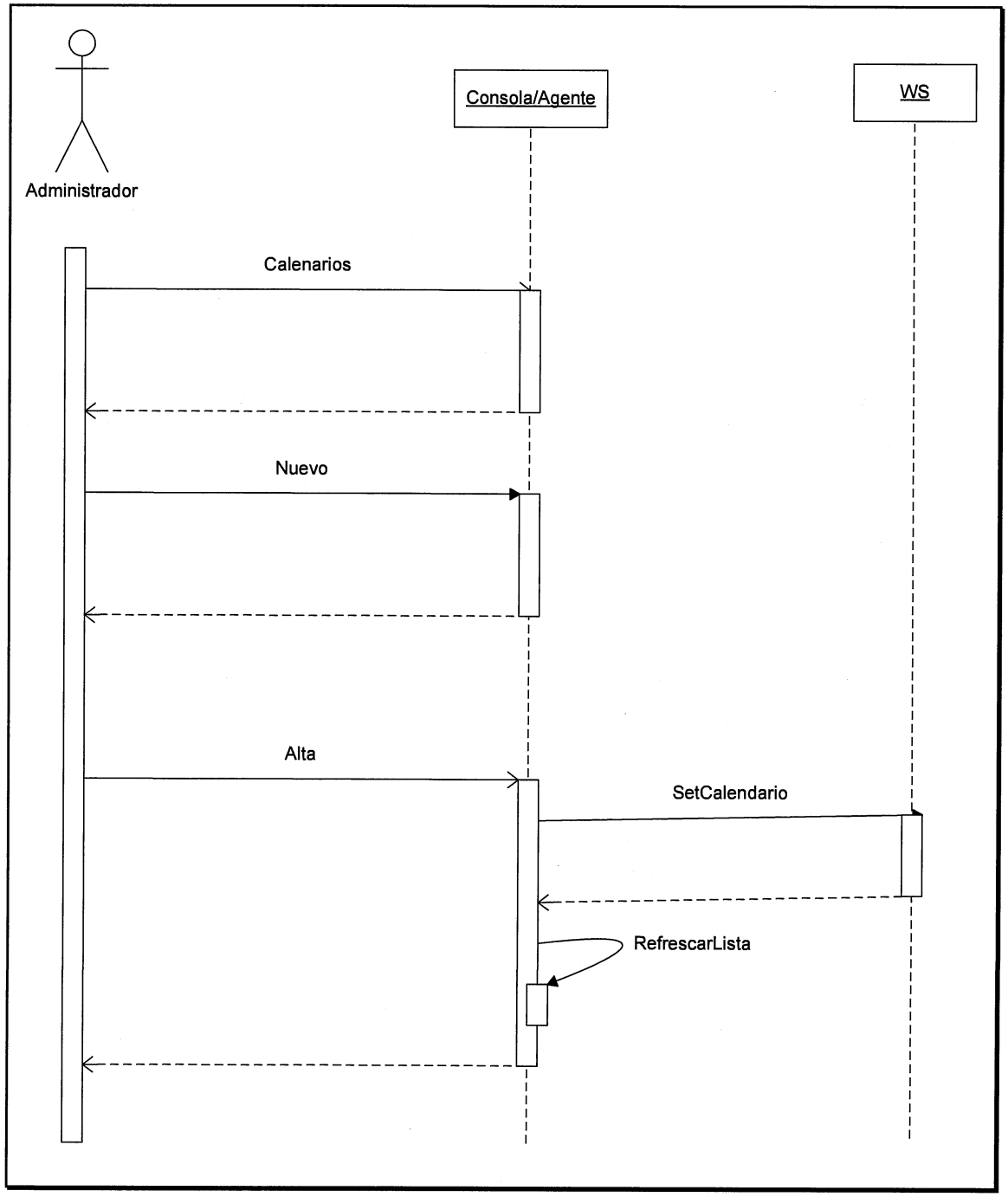


4.3.1.3 Añadir Servicio

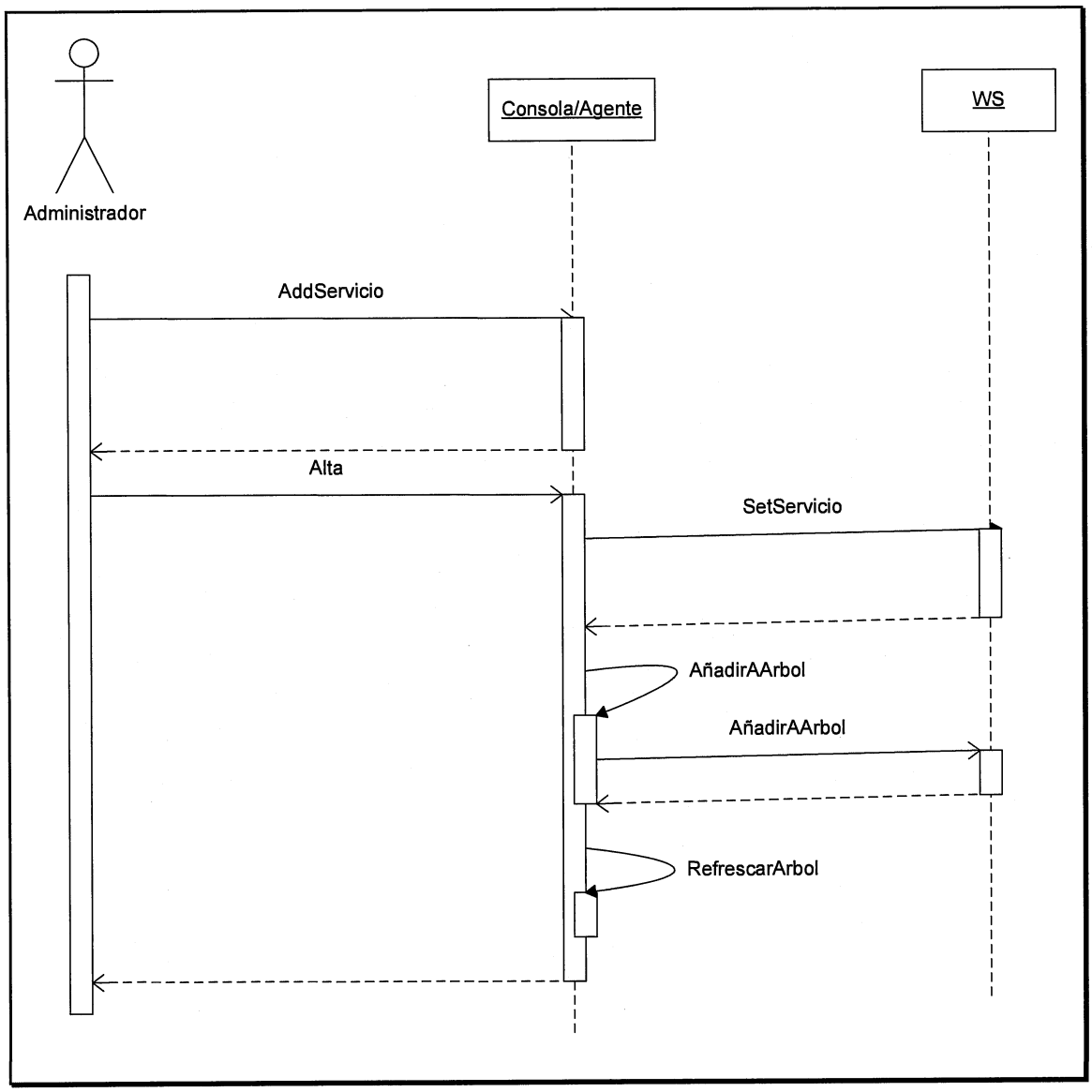
Este diagrama de secuencias, sirve para todos los diagramas de secuencia que indican añadir un elemento del árbol de navegación a otro.



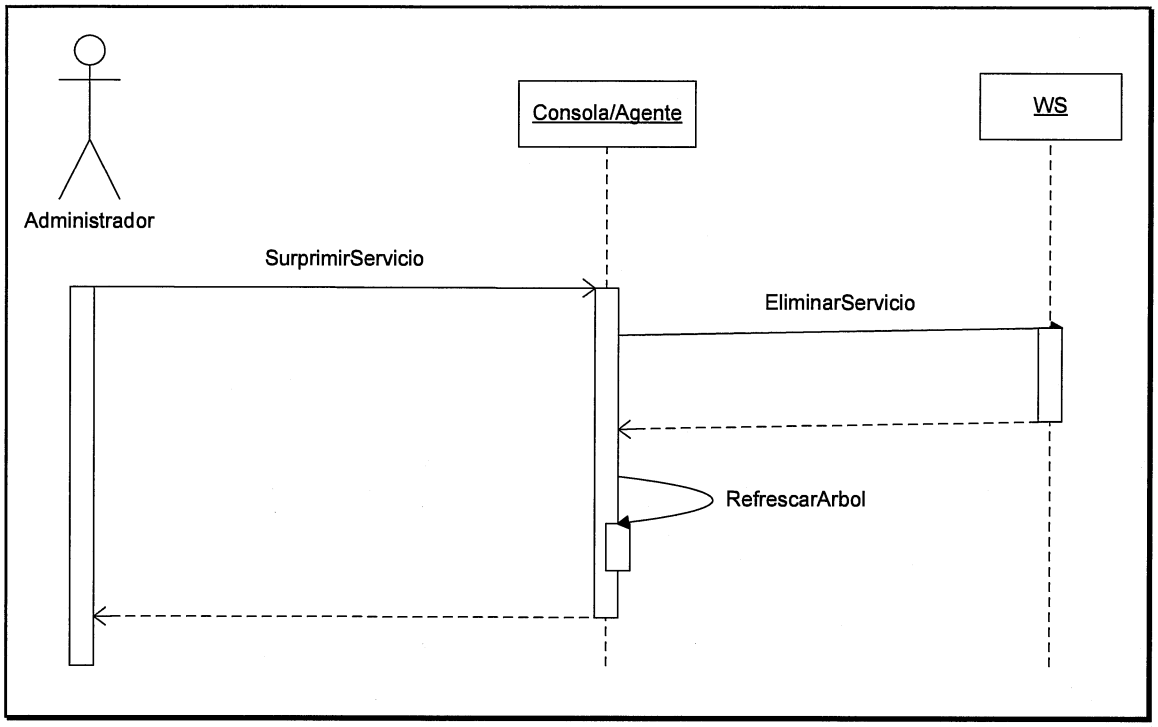
4.3.1.4 Robot – Mantenimiento Calendario (Alta)



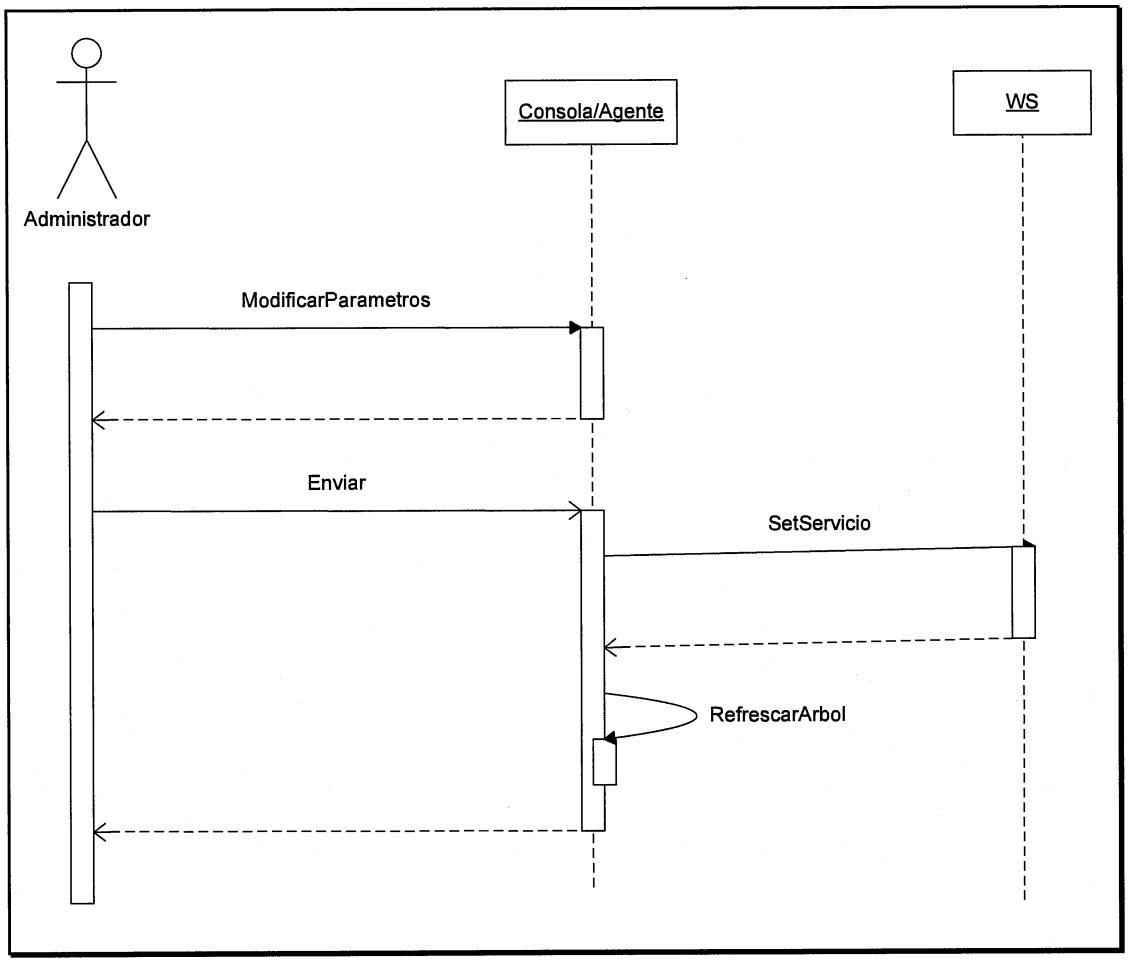
4.3.1.5 Perfil – Crear Servicio



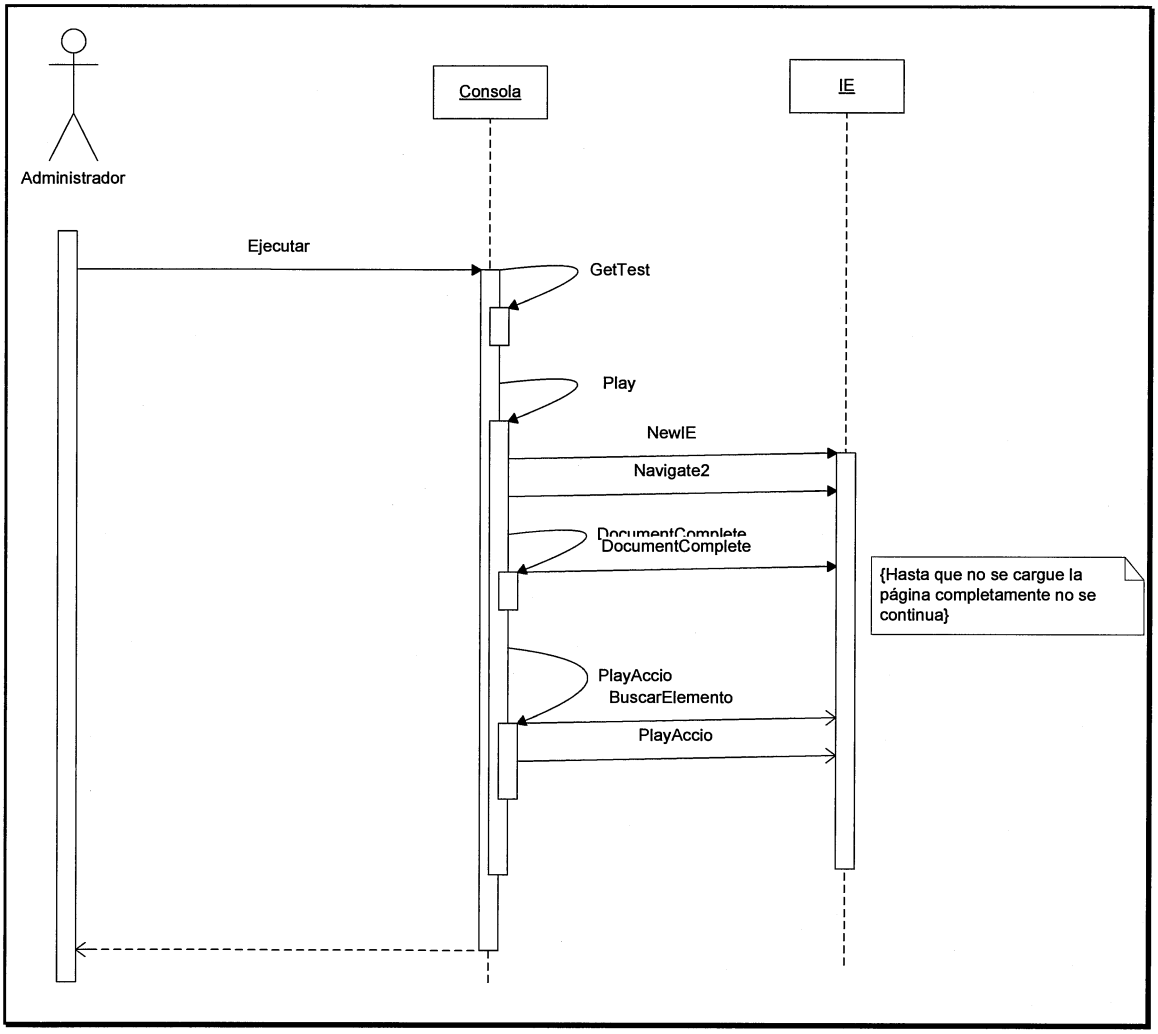
4.3.1.6 Servicio – Suprimir Servicio



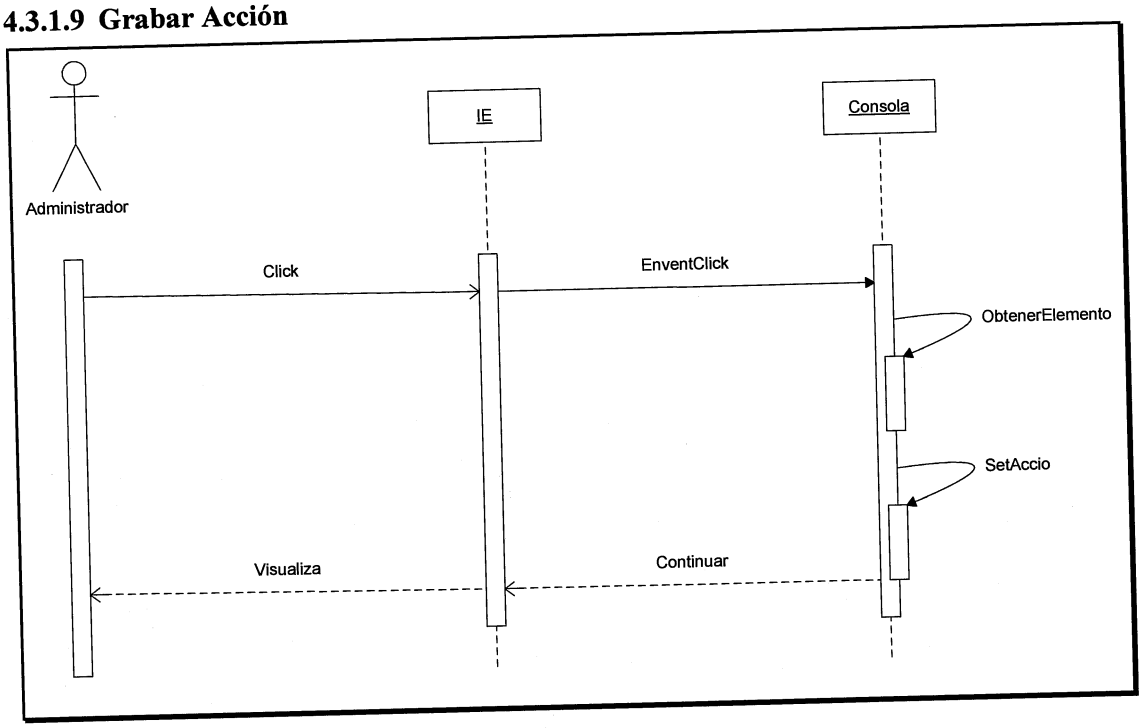
4.3.1.7 Servicio – Modificar



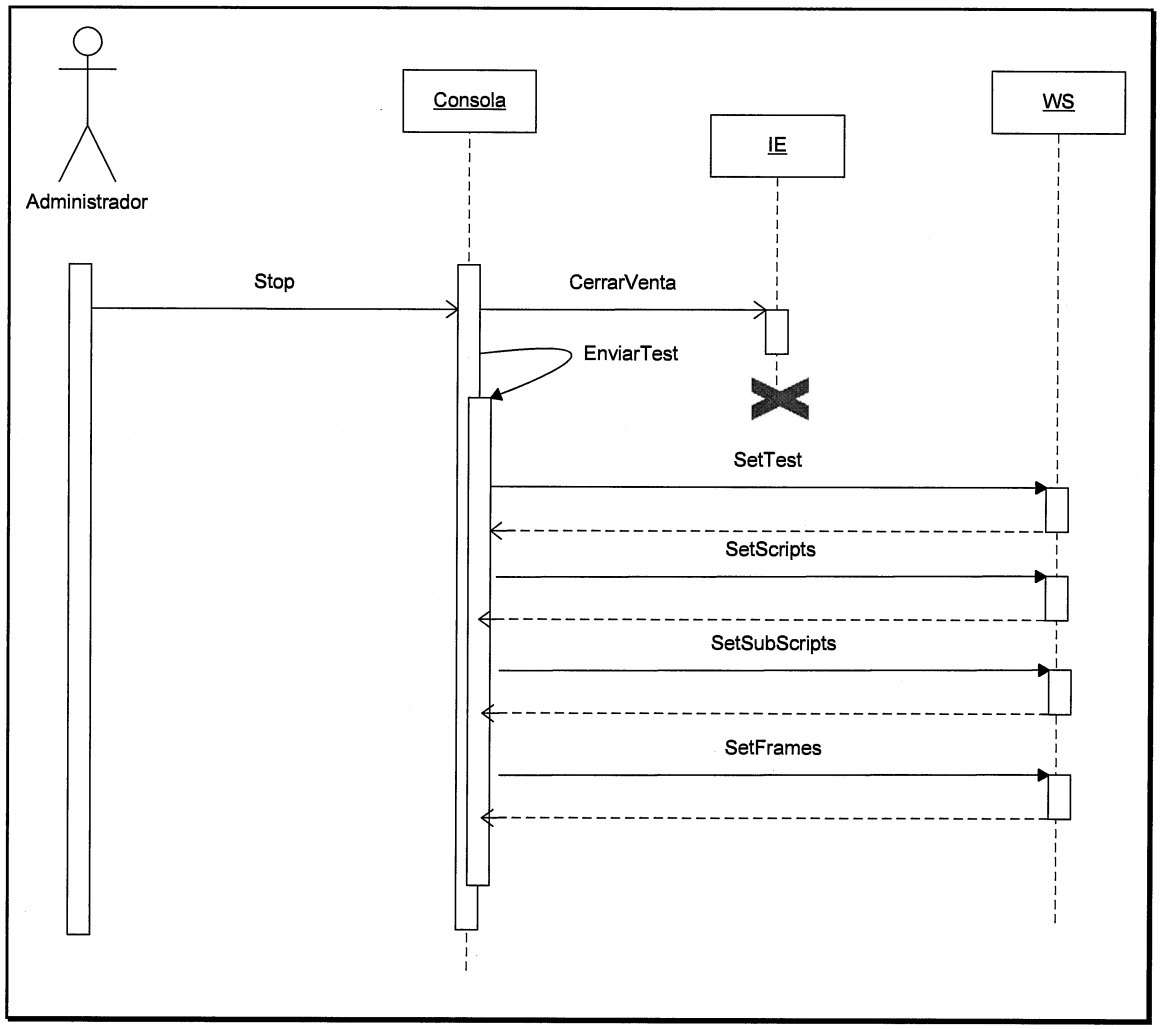
4.3.1.8 Test – Ejecutar



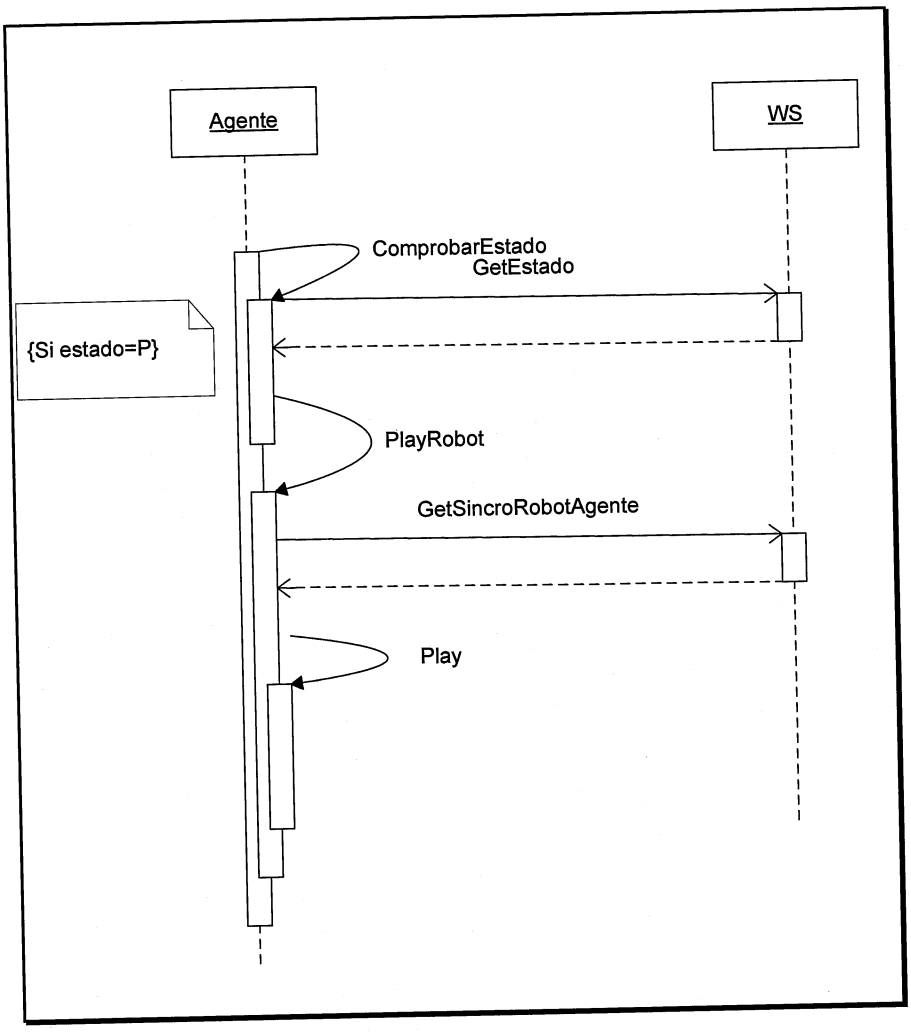
4.3.1.9 Grabar Acción



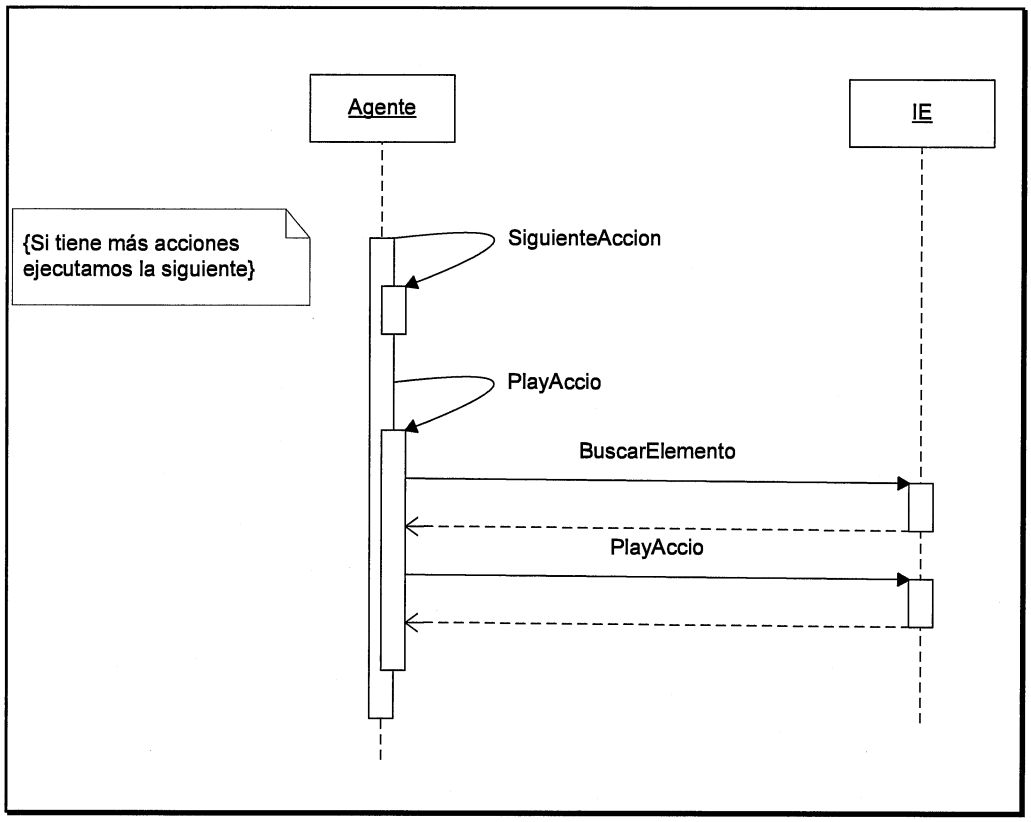
4.3.1.10 Parar Grabación



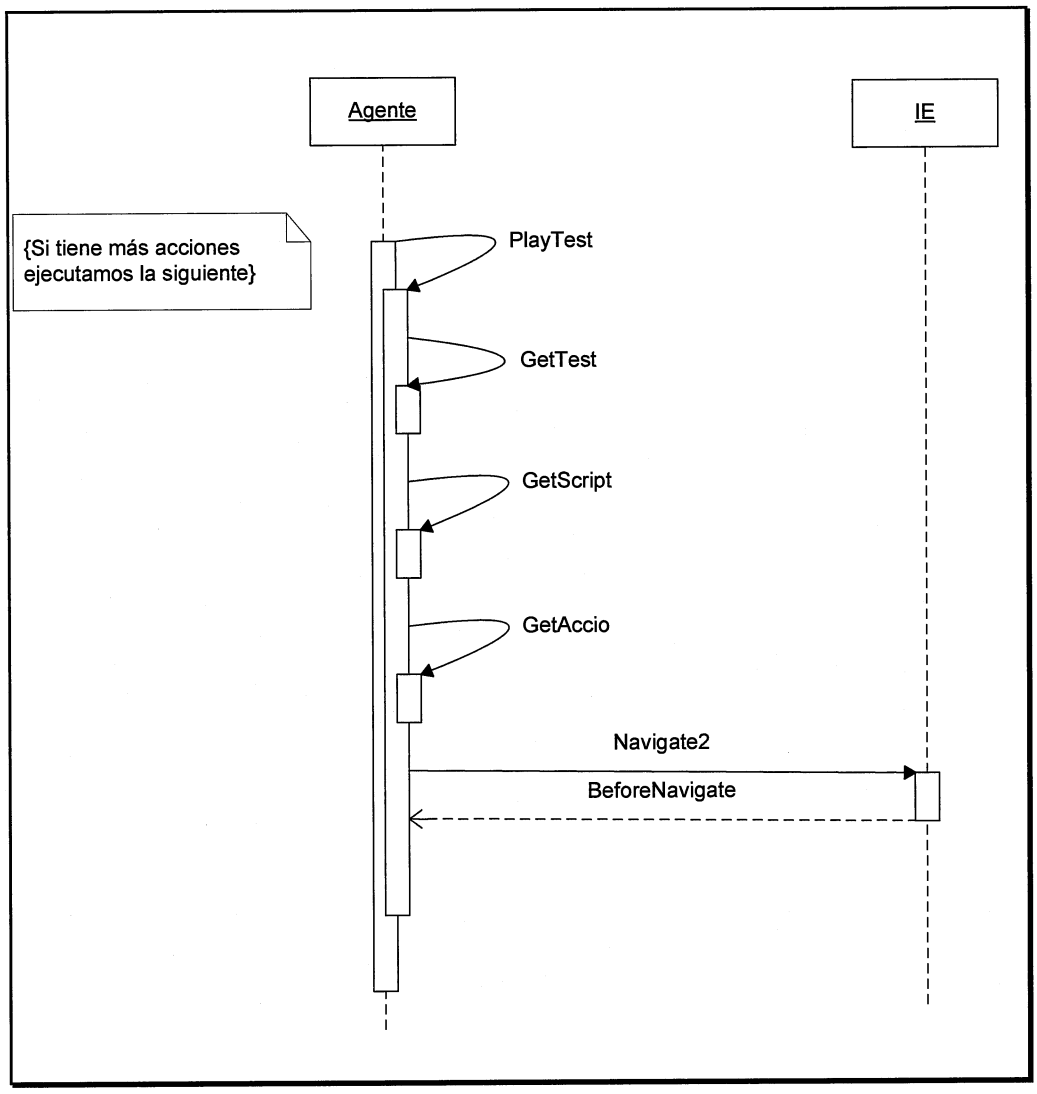
4.3.1.11 Inicio Monitorización



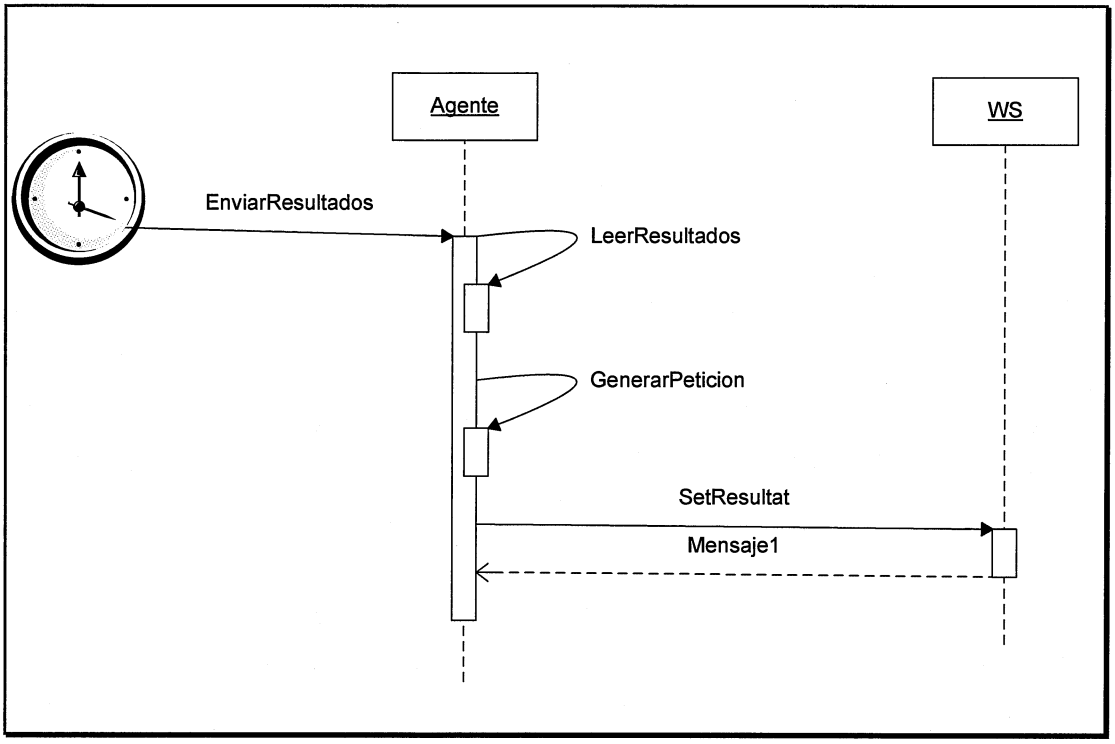
4.3.1.12 Ejecutar Acción



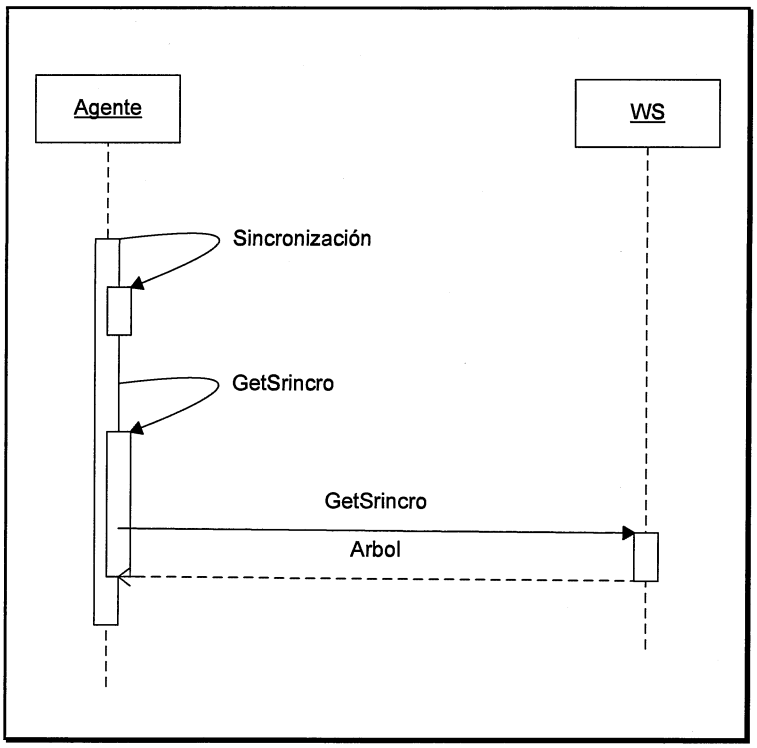
4.3.1.13 Ejecutar Test



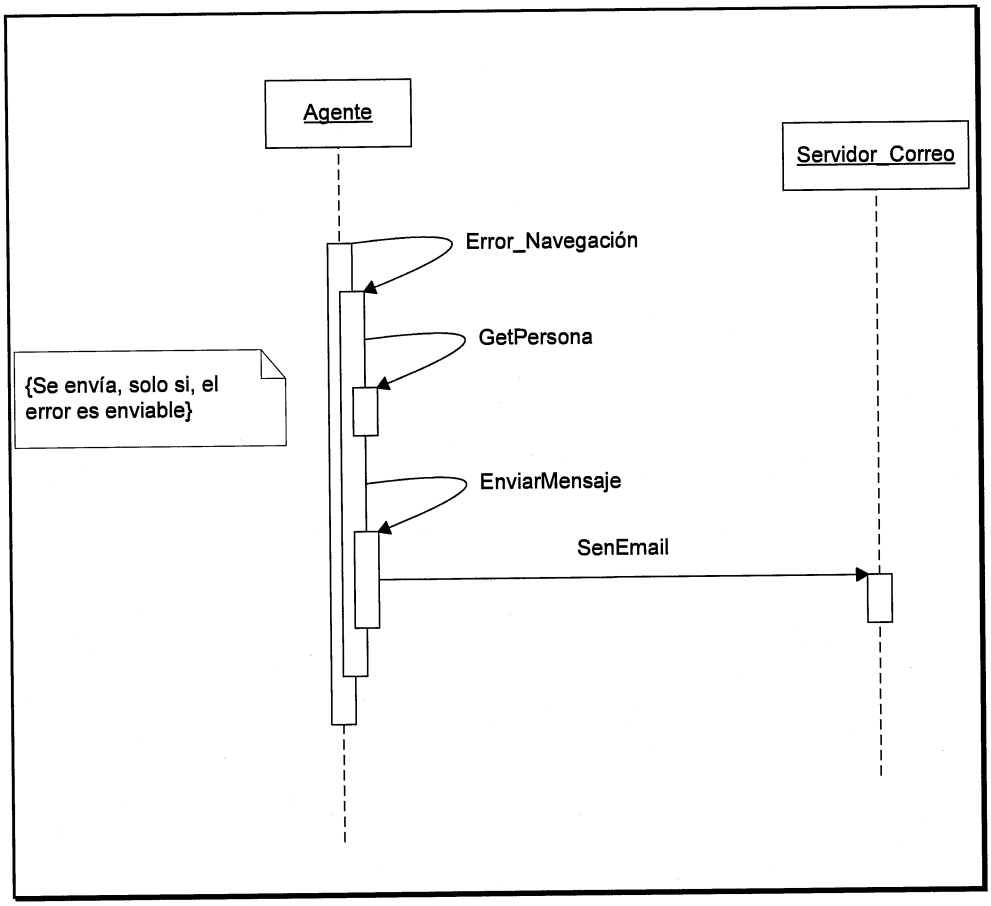
4.3.1.14 Enviar Resultados



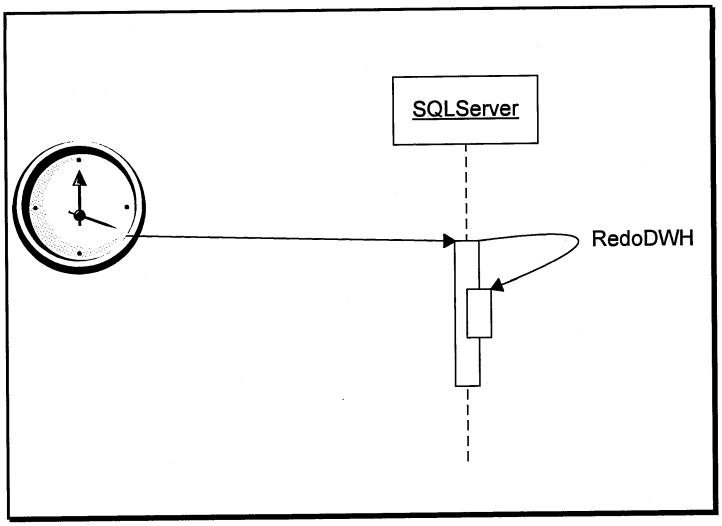
4.3.1.15 Sincronización



4.3.1.16 Enviar Alerta



4.3.1.17 Recalcular DataWareHouse



4.3.2 Contratos

Finalmente, siguiendo dentro del modelo de comportamiento del sistema, se presentan a continuación los contratos para las operaciones detectadas en los diagramas de secuencia del sistema. Éstos describen el efecto de estas operaciones sobre el sistema.

Name: PuestaenFuncionamiento
Responsabilities: Arrancar la consola o el agente.
Exceptions:

Si no existe un Web Service, base de datos local, o el fichero de configuración es incorrecto indicar error.

Preconditions:

- Debe haber conexión al WS
- Debemos tener base de datos local

Postconditions:
Salida: Visualización de la aplicación para el usuario con acceso a las funcionalidades.

Name: Seleccionar Robot
Responsabilities: Seleccionar un robot para visualizarlo en la consola y poder gestionarlo.
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:
Postconditions:
Salida: Visualización del Robot en el árbol de ejecución.

Name: Añadir Servicio, o cualquier acción de añadir un elemento a otro dentro del Árbol
Responsabilities: Añadir un elemento a otro, en este caso añadir un servicio al robot.
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:

- Seleccionar el robot
- Que exista algún servicio

Postconditions:

- Se crea la relación entre los dos elementos

Salida: Visualización del Robot modificado en el árbol de ejecución

Name: Robot- Mantenimiento Calendario (Alta), este diagrama sirve para todos los calendarios de los elementos del árbol
Responsabilities: Crear un nuevo calendario para el robot
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:
- Seleccionar el robot

Postconditions:
- Se crea el nuevo calendario
- Los agentes con este elemento se actualizarán cuando se sincronicen

Salida: Visualización en la lista de calendarios del nuevo calendario

Name: Perfil – Crear Servicio
Responsabilities: Crear un servicio y añadirlo al árbol de ejecución.
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:
- Seleccionar el Perfil

Postconditions:
- Se crea la relación entre los dos elementos

Salida: Visualización del Robot modificado en el árbol de ejecución

Name: Servicio – Suprimir Servicio, este sirve también para el resto de elementos que se quieran suprimir
Responsabilities: Suprime un servicio del árbol de ejecución, junto a todos sus elementos.
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:
- Seleccionar el Servicio

Postconditions:
- Se elimina la relación entre los dos elementos

Salida: Visualización del Robot modificado en el árbol de ejecución

Name: Servicio – Modificar Servicio, este sirve también para el resto de elementos que se quieran modificar
Responsabilities: Modifica los parámetros del servicio.
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:

- Seleccionar el Servicio

Postconditions:

- Se modifica el elemento en la base de datos

Salida: Visualización del Robot modificado en el árbol de ejecución

Name: Test – Ejecutar (hasta realzar la primera acción)
Responsabilities: Lanza la ejecución de un test.
Exceptions:

Si no disponemos de Internet Explorer, el test no tiene acciones o fallo en la navegación indicar error.

Preconditions:

- Seleccionar el Test
- El test este completo

Postconditions:
Salida: Se ejecuta el test

Name: Grabar Acción
Responsabilities: Grabar una acción en el test sobre el Internet Explorer
Exceptions:

Si no existe el Internet Explorer indicar error.

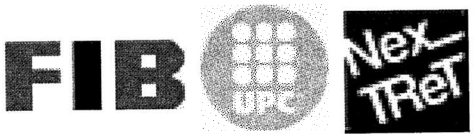
Preconditions:

- Disponer de un IE y un test ejecutándose

Postconditions:

- Se crea un nuevo subscript

Salida: Visualización del Robot modificado en el árbol de ejecución y se ejecuta la acción sobre el IE.



Name: Parar Grabación
Responsabilities: Parar la grabación del test y enviar el nuevo test al servidor central
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:
- Test ejecutándose

Postconditions:
- Se crea un nuevo test o se modifica

Salida: Visualización del Robot modificado en el árbol de ejecución.

Name: Iniciar Monitorización
Responsabilities: Arrancar la monitorización de un Robot en un Agente
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:
- Agente Arrancado
- Alguien haya iniciado desde la consola un Agente

Postconditions:
- Se arranca el Robot en el Agente

Salida: Se visualiza las navegaciones por medio del IE

Name: Ejecutar Acción
Responsabilities: Realizar la acción, previamente grabada, sobre el IE.
Exceptions:

Si no existe el IE, el elemento no existe indicar error.

Preconditions:
- Test ejecutándose
- IE arrancado

Postconditions:
Salida: Se ejecuta la acción sobre el IE

Name: Ejecutar Test
Responsabilities: Lanzar la monitorización de un test dentro del árbol de ejecución
Exceptions:

Si el test no tiene acciones indicar error.

Preconditions:
- Agente arrancado
- Test con acciones.

Postconditions:
Salida: Se ejecuta el test

Name: Enviar Resultados
Responsabilities: Enviar los resultados al servidor central
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:
Postconditions:

- Se eliminan los resultados de la tabla resultados de la base de datos global, y se mueven a la tabla auxiliar

Salida: Se envían los resultados.

Name: Sincronización
Responsabilities: Conocer los cambios en el árbol de ejecución del Robot.
Exceptions:

Si no hay conexión con Web Service o devuelve un error el servidor indicar error.

Preconditions:
Postconditions:

- Se modifica el árbol de ejecución del agente

Salida: Se monitoriza la nueva situación

Name: Enviar Alerta
Responsabilities: Dar a conocer los errores que se producen.
Exceptions:

Si no existe el servidor de correo, o devuelve algún tipo de error se indica error.

Preconditions:

- Se debe haber producido un error
- El error debe ser enviable
- El test que ha provocado el error debe tener personas asociadas

Postconditions:
Salida: Se envían las alertas

Name: Recalcular DWH
Responsabilities: Recalcular las tablas de agregación para mostrar los resultados obtenidos en el ReportingWeb
Exceptions:

Si se produce algún error de SQL indica error.

Preconditions:

Postconditions:

- Se modifican las tablas del DWH con los nuevos resultados

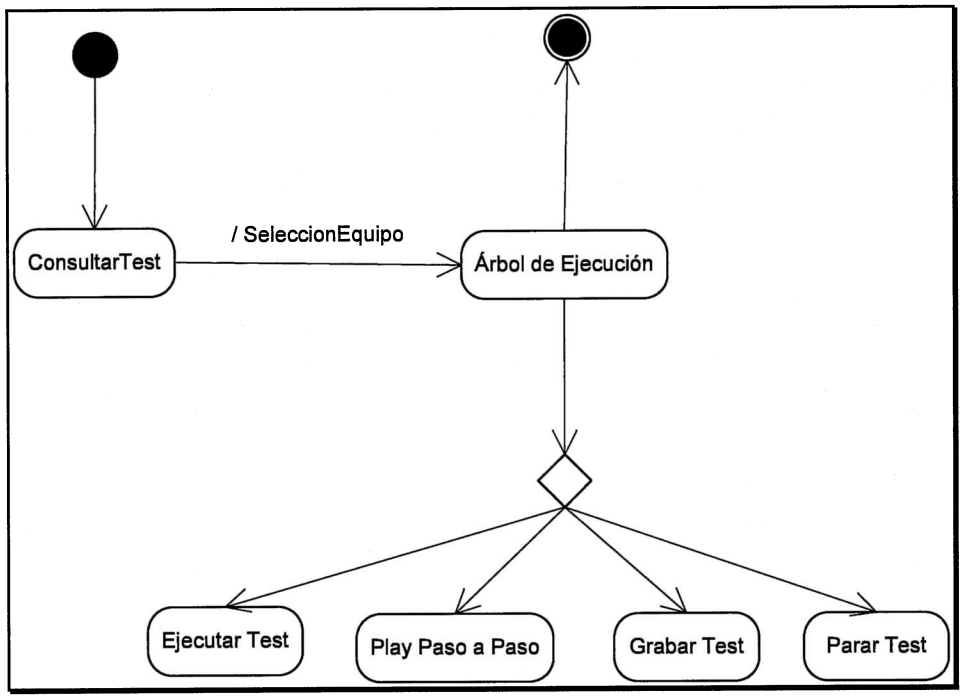
Salida:

4.4 Modelo de los estados

A continuación, se especifican claramente una serie de estados que se consideran importantes en el funcionamiento de la herramienta.

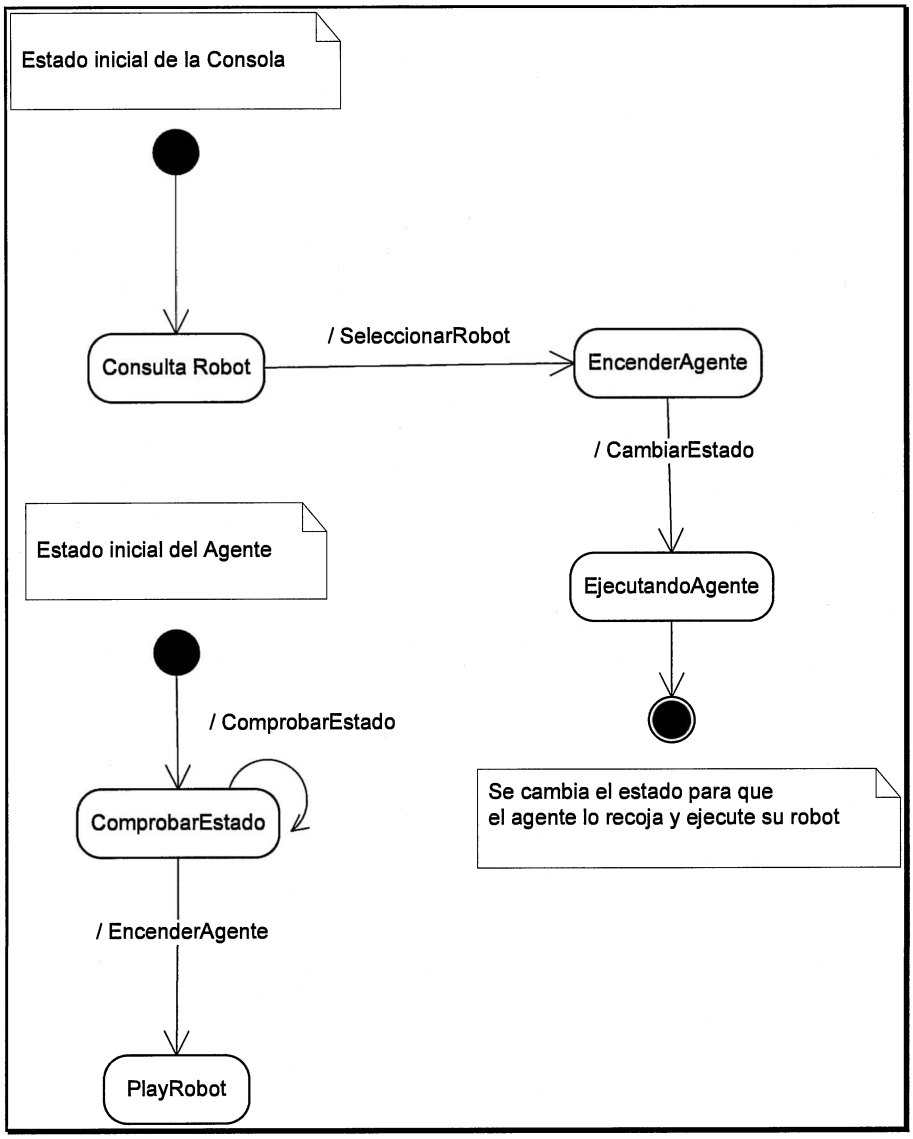
4.4.1 Generador de Scripts

En este diagrama de estados, se muestra como evoluciona el grabador de test desde que seleccionas un test hasta que realizas una de las acciones posibles.



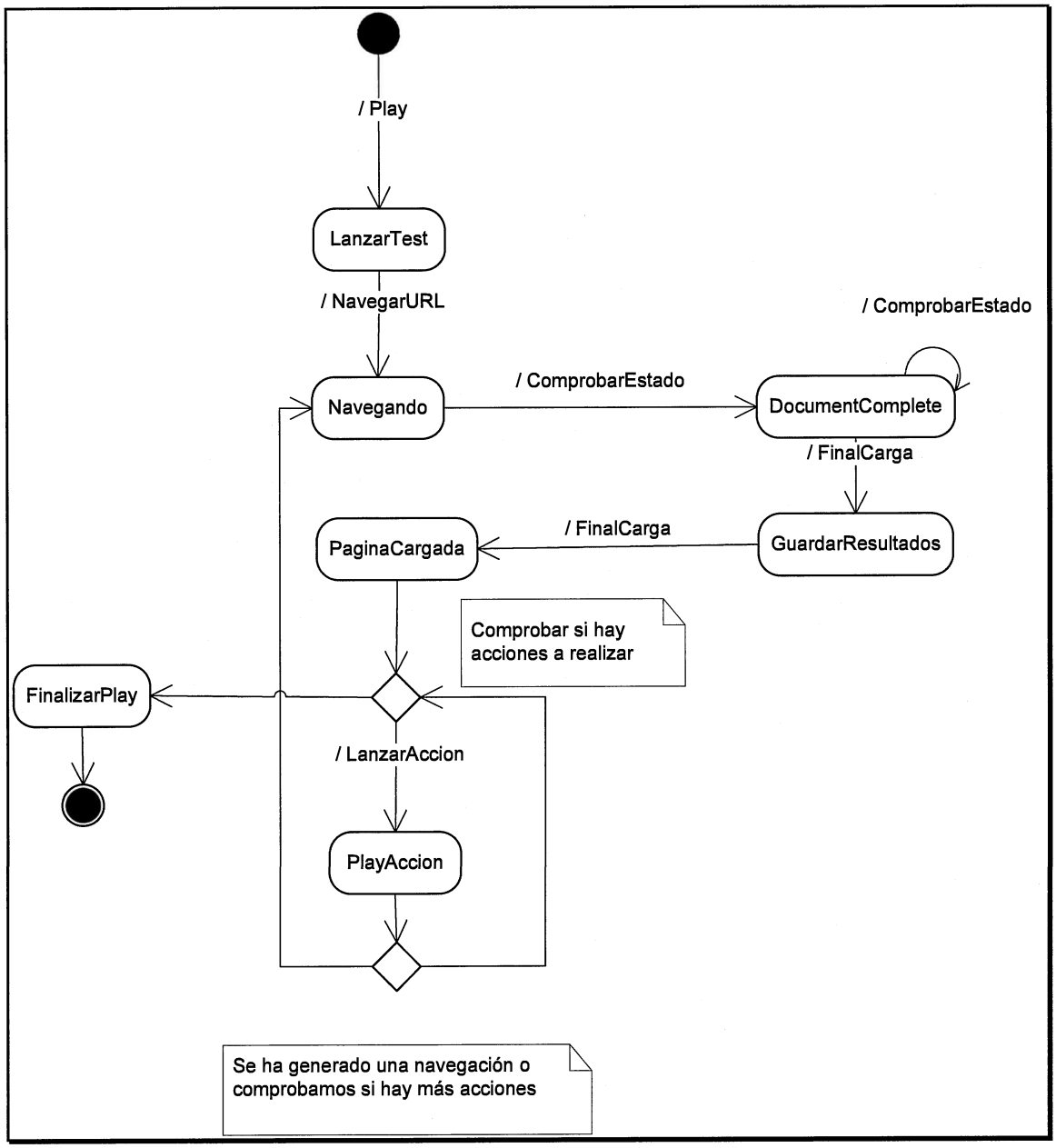
4.4.2 Encender Agente desde la consola

En este diagrama de estados, se muestran los pasos para activar la monitorización desde la consola y los pasos que hace el Agente para encenderse automáticamente.



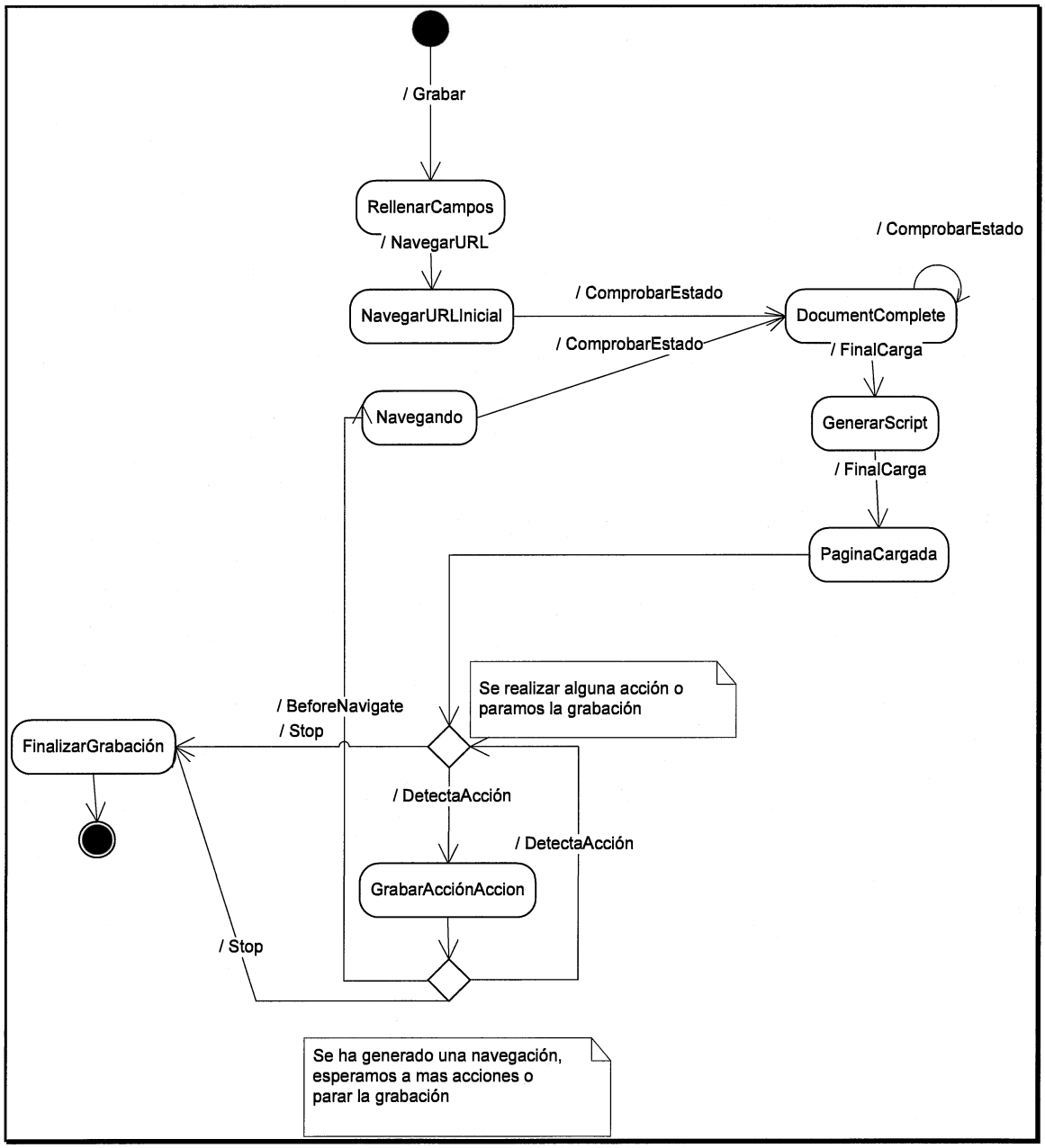
4.4.3 Estados del Navegador al reproducir

En este diagrama de estados, se muestran los pasos del navegador al reproducir una prueba.



4.4.3 Estados del Navegador al grabar

En este diagrama de estados, se muestran los pasos del navegador al reproducir una prueba.



5 *Diseño*

5.1 Introducción

A la hora de llevar a cabo el diseño de la aplicación fueron fundamentales, además de los requerimientos analizados en la primera parte de este documento, como son los requerimientos de software, de rendimiento, o de diseño, unos requerimientos económicos.
Se debe tener en cuenta que este sistema fue presentado a La Caixa, pero se pretendía que fuera un producto, por lo que la reducción de costes era fundamental, y justifica que, en algunos casos, algunas decisiones de diseño favorezcan más la reducción de costes, a la de alguno de los requerimientos previamente mencionados.
Recordamos nuevamente los requerimientos que deben tenerse en cuenta a la hora de diseñar el sistema son los siguientes:

Requerimientos de software.

- Comunicación distribuida, es necesario que las aplicaciones puedan estar en cualquier parte del mundo y se comuniquen a través de la red para poder sincronizarse.
- Diferente aplicación para grabar y para monitorizar, es necesario crear una aplicación para monitorizar (Agente) y otra para gestionar todo el sistema (Consola)
- Centralización de datos, los datos deben estar centralizados en un solo punto para poder realizar una explotación de los datos desde una única herramienta.
- Actualización en tiempo real, los resultados obtenidos se envían a la estación central en tiempo real.
- Aplicación multiusuario, es necesario que puedan existir diferentes usuarios realizando el mantenimiento del sistema.
- Acceso a datos vía Reporting Web, aplicación realizada en Java.
- Independencia de la plataforma en la parte de servidor web.
- Independencia del Gestor de bases de datos
- Utilización del Internet Explorer

Requerimientos de rendimiento:

- Reacción en tiempo real, tanto en la actualización como en la detección y envío de alarmas.
- Datos en el Reporting Web en tiempos de 10 minutos
- No interferir en el rendimiento del Internet Explorer para no distorsionar las muestras.

Requerimientos de diseño:

- Alto grado de usabilidad (Afecta al diseño de la interfaz gráfica del sistema)
- Utilización del Internet Explorer

5.2 Definición de patrón arquitectónico

Como se comentaba anteriormente, la necesidad de reducir costes en el desarrollo del aplicativo, influyen directamente en las decisiones tomadas para el diseño de la aplicación.

5.2.1 Plataforma de desarrollo

Una de las decisiones más importantes, y que afecta profundamente al diseño del aplicativo, es la elección de la plataforma de desarrollo.
En función de los diferentes requerimientos expuestos con anterioridad, se partía de los siguientes candidatos:

- Java.
- Microsoft Visual Basic.
- Visual C++

5.2.1.1 Java

Llevar a cabo el desarrollo en Java tiene las siguientes particularidades.

Ventajas:

- NexTReT dispone de un equipo experimentado en el desarrollo de aplicaciones en está tecnología, por lo que el coste de formación sería 0.

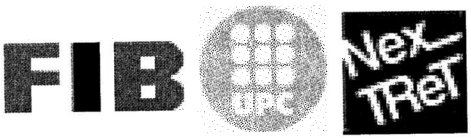
- Las licencias de Java son muy inferiores a la de otras tecnologías, por lo que permitiría llevar a cabo una propuesta económica más competitiva.

- Java, gracias a su JVM (*Java Virtual Machine*) es una tecnología multiplataforma, por lo que cumpliría uno de los requerimientos necesarios del sistema.

Inconvenientes:

- Java no se caracteriza por su rapidez, por lo que la monitorización en tiempo real de los clientes puede verse penalizada.

- El desarrollo de interfaces gráficas a través de Java, concretamente de Java swing, no se caracteriza por su facilidad de desarrollo.



- NexTReT no había desarrollado ninguna aplicación utilizando está tecnología enlazándola con un Internet Explorer, está es una de las premisas iniciales y más importantes.

5.2.1.2 Microsoft Visual Basic

Llevar a cabo el desarrollo en Visual Basic tiene las siguientes particularidades.

Ventajas:

- NexTReT dispone de un equipo experimentado en el desarrollo de aplicaciones en está tecnología, por lo que el coste de formación sería 0.
- Existen prototipos realizados con anterioridad de aplicaciones que enlazan controles ActiveX y más recientemente integración con Internet Explorer.
- Permite programar en sintaxis BASIC.
- Miles de foros alrededor de la Web le hacen el lenguaje con mayor cobertura/soporte que cualquier otro.
- La facilidad del lenguaje permite crear aplicaciones para Windows en muy poco tiempo. En otras palabras, permite un desarrollo eficaz y menor inversión en tiempo que con cualquier otro lenguaje.
- El IDE de casi todas las versiones de Visual Basic incluye un elevadísimo número de asistentes y plantillas. Todas las versiones incluyen opcionalmente directorios con material gráfico (iconos, cursores, imágenes) listos para añadir al proyecto.
- La sintaxis es flexible: se puede obligar al compilador a ignorar errores o escribir varias instrucciones en una misma línea. El IDE detecta las variables existentes, y en el caso de que al utilizarlas las escribamos de forma diferente - con mayúsculas, por ejemplo- cambia dicha variable en todo el código.
- Permite generar librerías dinámicas (DLL) ActiveX de forma nativa y Win32 (no ActiveX, sin interfaz COM) mediante una reconfiguración de su enlazador en el proceso de compilación.
- El IDE de la versión 6.0 ya permitía utilizar añadidos (Addins) que pueden programarse incluso en el propio lenguaje, lo que permite una importantísima personalización del interface.
- Permite la utilización de formularios (Forms) tanto a partir de recursos (como en otros lenguajes) como utilizando el IDE para diseñarlos. Esto permite utilizar la

interface de otros programas ya creados, tan sólo importando el archivo de recursos.

Inconvenientes:

- Microsoft Visual Basic, no es un lenguaje de programación orientado a objetos, lo que complica la programación estructurada. La escasa implementación de POO en Visual Basic 6.0 no permite sacar el máximo provecho de este modelo de programación.
- Es software propietario por parte de Microsoft, por tanto, nadie que no sea del equipo de desarrollo de está compañía decide la evolución del lenguaje.
- Es un entorno de programación que no dispone de threads
- Los ejecutables generados son relativamente lentos en Visual Basic 6.0 y anteriores al ser código pseudo-interpretado.

5.2.1.3 Microsoft Visual C++

Llevar a cabo el desarrollo en Visual C++ tiene las siguientes particularidades.

Ventajas:

- Permite realizar cualquier cosa que se proponga el programador, ya se puede acceder hasta bajo nivel.
- Existe una extensa documentación lo que hace que la ayuda sea extensa y se solucionen los problemas con rapidez.
- Soporta POO.

Inconvenientes:

- NexTReT no había desarrollado ninguna aplicación utilizando está tecnología enlazándola con un Internet Explorer, está es una de las premisas iniciales y más importantes.

- NexTReT no tiene un equipo de desarrolladores en Visual C++ con experiencia suficiente en está tecnología.

- Los programadores deben tener una experiencia elevada en el desarrollo de aplicaciones en Visual C++ para

5.2.1.3 Decisión final

Una vez expuestas y analizadas las diferentes ventajas e inconvenientes de ambos candidatos, fue igualmente clave el análisis del equipo servidor necesario para soportar, de modo fiable, un sistema como el requerido.

Es necesario un servidor de base de datos que soporte las consultas de consulta de datos y que disponga de espacio suficiente para almacenar los datos que recogen los agentes distribuidos.

Al final el servidor que se utilizó fue un servidor con Windows Server 2003 y SQL Server 2000 ya que "La Caixa" dispuso de un servidor corporativo con las características indicadas. Este servidor está incluido en todos los circuitos de control del cliente, por lo cual, dispone de alta disponibilidad y de backups programadas que permiten la recuperación total en caso de caída del sistema.
De todas formas, aunque el cliente inicial dispone de SQL Server, y como este proyecto iba a ser en un futuro un producto de mi empresa se pensó y se realizó todo el desarrollo basándonos en SQL Standard, los elementos que dependen de cada una de las plataformas se desarrollaron por separado una vez realizado el primer proyecto.

El entorno se decidió que fuera un entorno Windows ya que una de las premisas iniciales era que se debía utilizar el navegador de Microsoft Internet Explorer.

La plataforma que se eligió, para desarrollar las herramientas de Consola y Agente, fue Microsoft Visual Basic, los motivos fueron la experiencia anterior con este tipo de proyectos que tenia el equipo de desarrolladores de la empresa.
Anteriormente, se había realizado un proyecto utilizando la misma tecnología. Se utilizó Visual Basic enlazado con un Internet Explorer.

La plataforma que se eligió, para desarrollar la plataforma de consulta de datos, el Reporting Web, fue utilizar Java, HTML, JSP y JavaScript. Ya que en este caso se pretendía que fuera una herramienta Web multiplataforma.

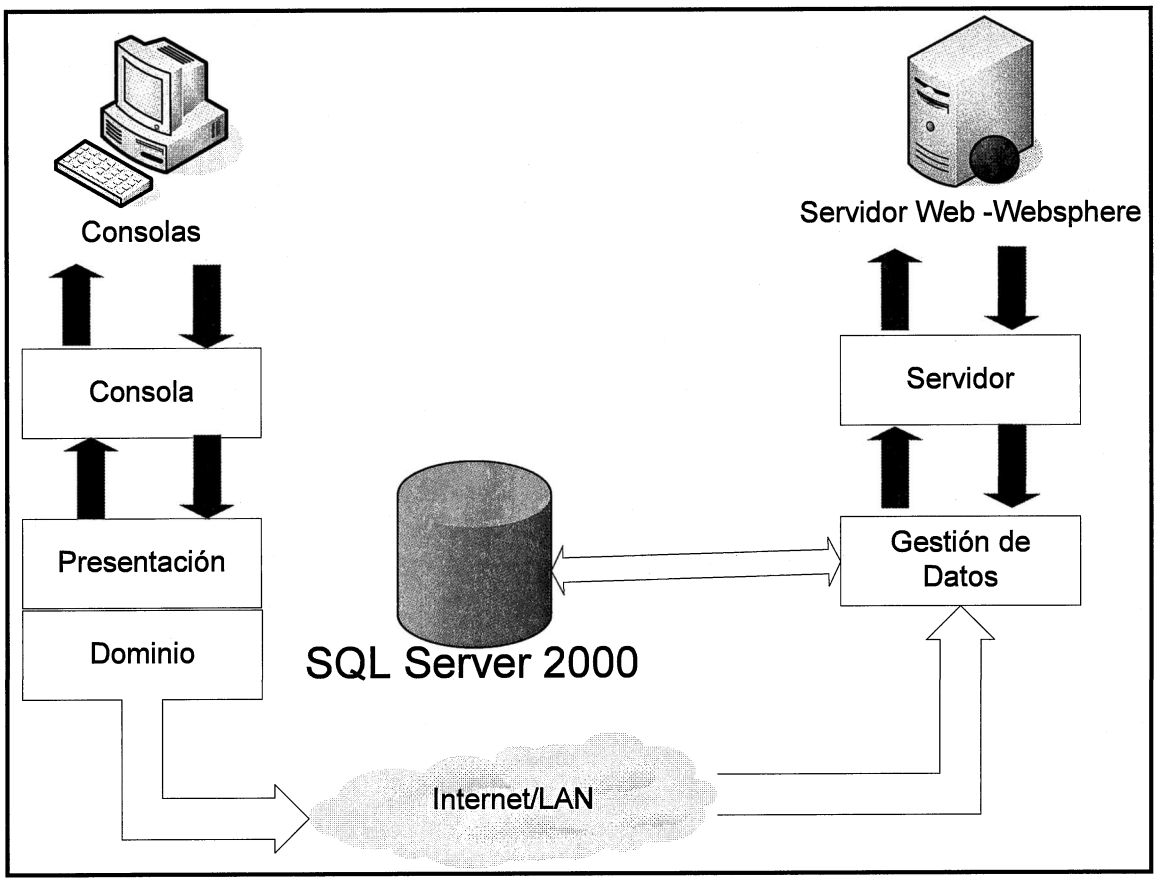
5.2.2 Patrón arquitectónico

Una vez decidida la plataforma de desarrollo, a la hora de establecer el patrón arquitectónico del sistema, debemos estudiar los siguientes escenarios:

Comunicación cliente / servidor.

5.2.2.1 Comunicación cliente/servidor

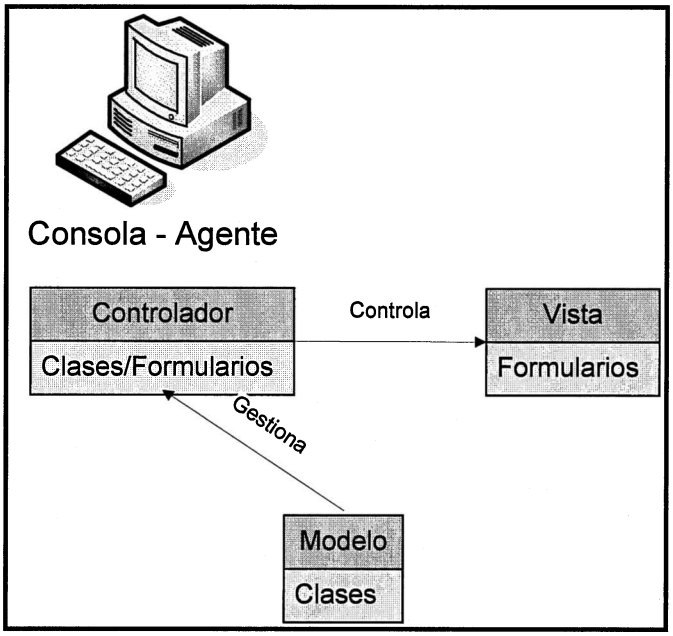
La siguiente figura representa gráficamente la comunicación que existe entre la consola, agente y el servidor Web, mediante servicios Web instalados en el servidor y este a su vez se utiliza el servidor de bases de datos SQL Server 2000:



Como puede observarse el sistema muestra una capa de presentación ubicada en el cliente, una capa de dominio, para la gestión de las operaciones.
En la parte del servidor se dispone de la capa de gestión de datos, se encarga de la conexión con la base de datos.

El sistema por tanto, es un sistema en tres capas distribuido, que utiliza servicios Web para comunicarse con el servidor. De está forma la plataforma de comunicación tiene

que ser una LAN o Internet, con esto se consigue que se pueda colocar en cualquier estación con conexión a la red y con visibilidad a través de ella del servidor.
Para continuar con el análisis del patrón arquitectónico vamos a entrar en detalle en la capa de presentación. La capa de presentación no cumple un patrón puro de Modelo-Vista-Controlar, pero se puede parecer al patrón señalado. Microsoft Visual Basic, no ayuda al diseño de aplicaciones con este patrón.



Como se ha comentado, no se dispone de un patrón puro de Modelo-Vista-Controlar. El Controlador y la vista están interconectados, ya que las particularidades del entorno de desarrollo elegidas hacen que se relacionen entre ellas y la diferenciación en el desarrollo no sea clara. Conceptualmente, es clara la diferenciación, ya que está dividido el código dentro de un objeto, pero los objetos pueden pertenecer a dos capas.

A continuación, se detallan y explican las tres capas que intervienen en el patrón indicado:

- **Controlador**. Gestión de acontecimientos sucedidos en la vista, i de las modificaciones del modelo. Traduce estos acontecimientos en peticiones o servicios. Esta capa se encarga de realizar las peticiones http al servicio Web alojado. A su vez se encarga de controlar la base de datos local para controlar los envíos al servidor.

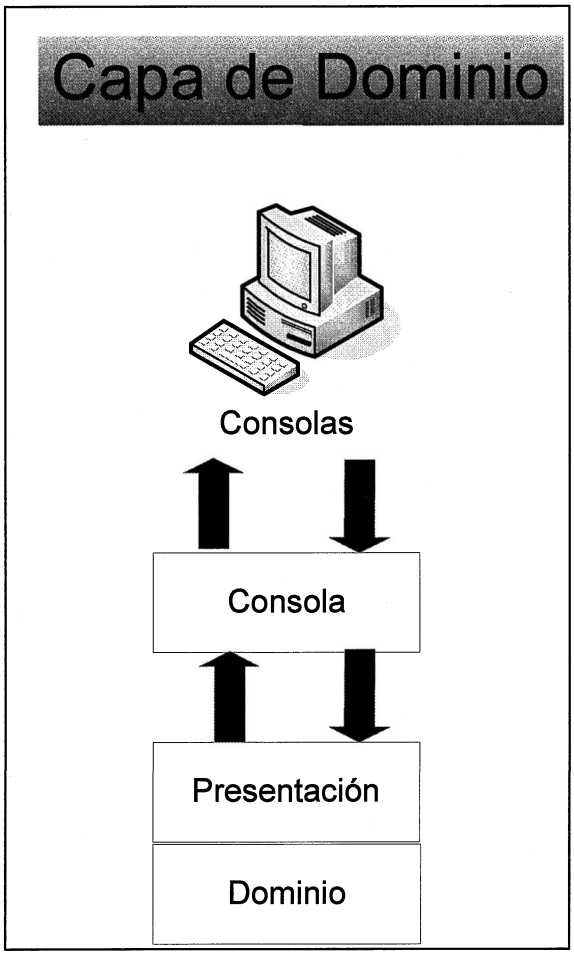
- **Vista**. Presentación de la información al usuario. Es gestionado por un controlador y proporciona operaciones al usuario. En está capa se incluye todos los formularios que intervienen en la interacción con el usuario. Aquí se incluye el navegador IE que es el elemento principal para realizar la navegación y para realizar la monitorización de las aplicaciones a testear.

- **Modelo**. Modulo gestión de la aplicación. Encapsula las funcionalidades del sistema, de forma independiente de la presentación. Proporcionando al controlador los servicios para satisfacer las peticiones del usuario.

Para continuar, se presenta la capa de **dominio**, está capara está incluida dentro de las aplicaciones cliente. Estas aplicaciones son la aplicación del Agente y de la Consola.

Dentro de está capa se encuentra toda la lógica de la aplicación, desde ella se controlan los siguientes elementos:

Consola



- Lógica de altas, bajas y modificaciones de Robots, se controla el estado de los robots y su funcionamiento.

- Lógica de altas, bajas y modificaciones de Test, desde la consola se controla los tests a monitorizar. Desde está parte se puede grabar test, reproducirlos y modificarlos.

- Lógica de altas, bajas y modificaciones de Servicios, desde la consola se controla los servicios a incluir.

- Lógica de altas, bajas y modificaciones de ISP, los ISP se usan para navegar por los tests y se pueden realizar el mismo test por diferentes ISP.

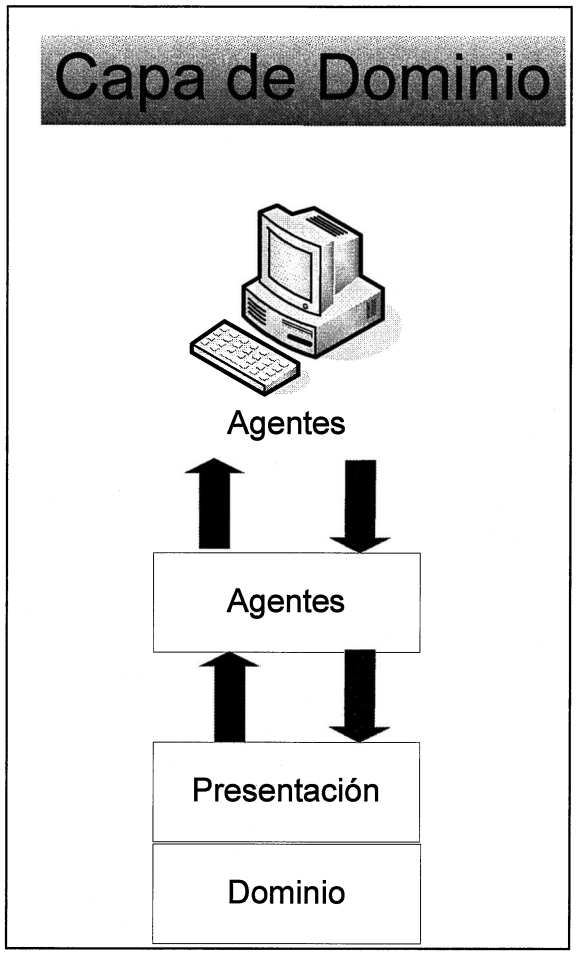
- Lógica de altas, bajas y modificaciones de Teléfonos. Desde la consola se gestiona la lógica de los teléfonos y direcciones de correo que recibirán alertas por SMS o por correo electrónico respectivamente.

- Lógica de altas, bajas y modificaciones de Personas. La gestión de los accesos al Reporting Web de ISM se gestiona desde la consola. A su vez, se gestiona la lógica de la aplicación para asignar a cada uno de las personas los tests a los cuales va a tener acceso para visualizar las estadísticas obtenidas por los Agentes de ISM.

- Lógica de altas de test en árboles de ejecución. En la consola está implementada toda la lógica para asignar los tests a monitorizar, los IPS por los cuales se monitoriza y la agrupación de test llamado servicios a incluir en cada robot.

- Lógica de control del Internet Explorer, desde la consola se controla toda la lógica que hay que implementar para gestionar todos los eventos del IE. De está forma se conoce el estado, en cada momento, de la grabación de las pruebas que posteriormente se monitorizarán por los Agentes de ISM.

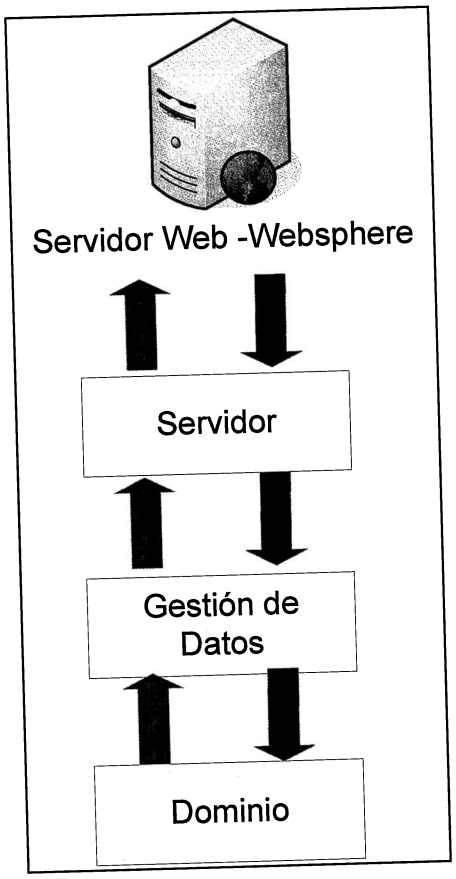
Agente



- Lógica de monitorización, desde la capa de dominio del Agente se dispone de la lógica que controla la monitorización de cada una de las pruebas que debe ejecutar el Agente y que previamente desde la Consola se han programado. Aquí se encuentra la lógica que controla los tiempos de respuesta obtenidos, los errores posibles que se encuentre durante la monitorización y las acciones a realizar dentro del Internet Explorer para continuar con la monitorización.

- Lógica de control del Internet Explorer, desde la consola se controla toda la lógica que hay que implementar para gestionar todos los eventos del IE. De está forma se conoce el estado, en cada momento, de la grabación de las pruebas que posteriormente se monitorizarán por los Agentes de ISM.

La capa de dominio está situada en los agentes y consola ya que la parte más importante, que es la monitorización, se realiza desde las aplicaciones de cliente. También existe una pequeña unidad de dominio en el servidor y en la aplicación de Web Service que contiene la lógica:



- Control de acceso a los robots desde la consola. En el servidor se gestiona los accesos a cada uno de los robots desde la consola, gestionando los accesos simultáneos a un mismo robot y los posibles bloqueos de cada uno de los agentes.

- Controla y bloquea los accesos a los tests, desde una consola se puede modificar un test y desde otra consola no se debe poder modificar. Esta lógica se encuentra en el servidor, ya que el servidor dispone de la información necesaria para tomar estas decisiones.

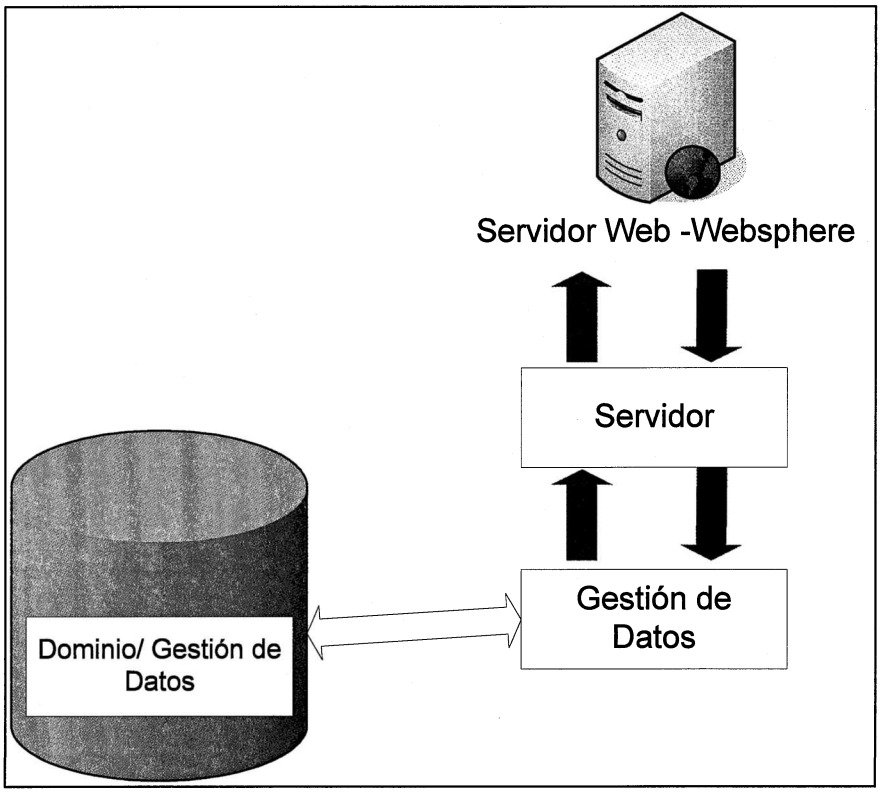
- Sincronización de Agentes y Consolas, la lógica de que partes debe actualizar cada uno de los agentes, se encuentra en el servidor. El servidor recibe los cambios de cada una de las consolas y decide a petición de los agentes que se debe actualizar para continuar con la monitorización.

- Control de inserciones de resultado. Cuando el agente realiza las monitorizaciones de los tests fijados obtiene resultados y los almacena en la base de datos local, cada x tiempo el agente envía los resultados de forma asíncrona, al servidor por medio de los Web Services, sin esperar la respuesta. Es el propio servidor el que ejecuta las inserciones y las tareas necesarias para almacenar los datos correctamente. De forma periódica, el agente pregunta que resultados se han insertado y es el propio servidor el que tiene la lógica para decidir que

resultado se ha insertado correctamente y cual no y el agente debe reenviar estos resultados

Finalmente, la capa de **gestión de datos** es la encargada de gestionar los datos y las conexiones con la base de datos. Se encarga de aportar transparencia con el SGBD, aportando transaccionalidad a las operaciones, utilizando el lenguaje SQL con las particularidades propia del SGBD utilizado, concretamente SQL Server 2000.

Se ha intentado y conseguido ser independiente del SGBD elegido en cada instalación. Por ello, muchas partes del código se han colocado directamente en la base de datos, mediante storeds procedures que permiten ejecutar la lógica de ciertas acciones.



La capa de gestión de datos está incluida en el Web Services. Esta parte está realizada en Java y para realizar las conexiones se utiliza JDBC y los drivers de conexión entre Java y cada unas de las bases de datos en la cuales puede funcionar. En el caso que se ha presentado en este PFC, se han utilizado los drivers proporcionados por Microsoft para conectarse desde Java a SQL Server 2000.

La capa situada dentro del servidor de base de datos dispone de la lógica de los siguientes puntos.

- Inserción de resultados, se encarga de insertar los resultados en la tabla inicial donde se almacenan todos los datos de cada una de las páginas monitorizadas.

- Actualización del DataWareHouse, se encarga de actualizar el DWH a petición del usuario o por ejecución desde un Jobs o tarea programada desde el propio sistema.

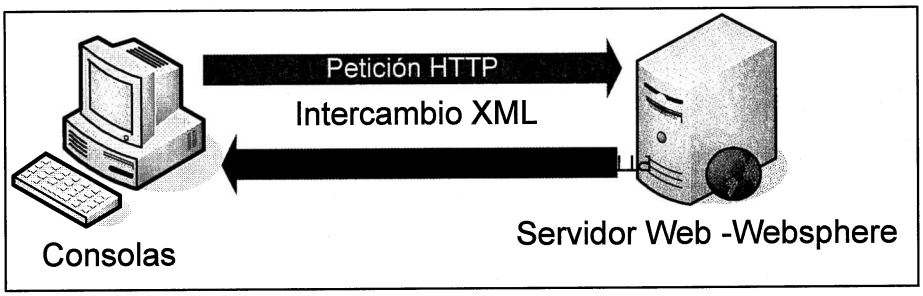
- Sincronización, se encarga de comunicar al Servicio Web si tiene o no que actualizar datos a la hora de sincronizar.

Otra lógica colocada dentro del propio SGBD es la programación de tareas mediante Jobs de SQL Server 2000.

Estos jobs realizan tareas, periódicas y configurables desde el propio SGBD, para actualizar el DWH que utiliza el Reporting Web para mostrar los datos de las monitorizaciones.

5.2.2.2 Protocolo de Comunicación cliente/servidor

Las comunicaciones entre los agentes/consola y el web service se realiza mediante la utilización de XML.



Las ventajas para la utilización de XML son las siguientes:

- Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.

- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De está manera se evitan bugs y se acelera el desarrollo de la aplicación.

- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

El formato XML que se ha utilizado cumple las siguientes características:

- Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.

- Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, sólo puede tener un elemento inicial.

- Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.

- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que el procesador XML trata de forma diferente en el marcado XML.

- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.

- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos entendibles por las personas.

El protocolo que se utiliza se base en encapsular las peticiones dentro de un tag "Peticiones" y colocar dentro de la raíz todas las acciones/peticiones que el agente o la consola demandan del servidor.

Existen dos tipos de peticiones que se pueden realizar:

- Peticiones sincronías, estas peticiones deben ser atendidas antes de responder a la petición http, el agente o la consola requieren información o alguna acción que le puede prestar el servidor.

- Peticiones asíncronas, estas peticiones no han de ser atendidas antes de responder a la petición http, el servidor responde que ha recibido la petición correctamente y responde al cliente con información de recepción correcta. El propio servidor almacena los datos para ejecutar la petición en cuanto sea posible. Este tipo de peticiones suelen requerir alguna otra petición para comprobar que el trabajo anteriormente demandado se ha realizado correctamente.

A continuación, se puede ver el esquema de una petición al servidor web por medio de XML:

```
<Peticions idRobot='x'>
<Petición1> </Petición1>
<Petición2> </Petición2>
.....
<Peticiónn> </Peticiónn>
</Peticions>
```

La respuesta del servidor se realiza por cada una de las peticiones incluidas dentro del tag "Peticions".

```
<Peticio>
<RespuestaPetición1 value="OK/KO">
</RespuestaPetición1>
<RespuestaPetición2 value="OK/KO">
</RespuestaPetición2>
.....
<RespuestaPeticiónn value="OK/KO"> >
</RespuestaPeticiónn>
</Peticio>
```

Como se observa, en cada una de las respuestas se incluye un valor que indica si la petición se ha realizado correctamente (OK) o incorrectamente (OK). Dependiendo de la petición también se incluye un identificador de error que puede dar idea que porque la petición ha fallado.

Cada petición tiene sus particularidades y puede tener más de un atributo que indiquen valores necesarios para los clientes. De la misma forma, pueden tener más ramas del XML donde se indica la información demandada por los clientes.