UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2 "TELECOMUNICACIONES Y SEGURIDAD INFORMÁTICA"

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Plataforma de Gestión de Servicios Telemáticos en GNU\Linux**.** Sistema de Inventario de Hardware y Software. Módulo Obtención de Información.

Autores:

Camilo Denis González.

Alié Castillo Ruiz.

Tutor: Ing. Ramón Alexander Anglada Martínez.

Ciudad de La Habana, Mayo del 2010.

"Año 52 de la Revolución"

Declaración de Autoría

Declaramos que _____ y _____somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) y a la Facultad (2) para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de julio del 2009.

_____ _____

Firma del Autor Firma del Autor

<Autor 1><Autor 2>

_____

Firma del Tutor

<Tutor>

"Si me lo explicas, lo olvidaré;
si me lo muestras, lo recordaré;
si me dejas hacerlo, lo aprenderé."

-- Lao Tzu.

AGRADECIMIENTOS

Alie Castillo Ruíz

- A mis padres Rosa Ruíz Rodríguez e Idel Castillo Chaveco, que han sido mis guías en la vida, y han estado día a día a mi lado celebrando mis éxitos como estudiante y acompañándome en las preocupaciones.
- A mi familia, y en especial a mi hermana Laritza Castillo Ruíz, mis primos Reinier Ruíz y Marbelis Portales Ruíz por el apoyo incondicional a lo largo de estos 5 años de estudiante universitario.
- A Yoerlis Pico, Francisco Izquierdo, Jenny de la Rosa Pasteur, compañeros del apartamento y de mi grupo, a mis amigos de la universidad y de la zona.
- A la gente del proyecto Servicios Telemáticos y en especial a las plagas.
- A todos los que de una forma u otra han hecho de mi un hombre de bien.

Camilo Denis González

- A mi familia y en especial a mis padres Agustina González Pluma y Camilo Denis Duanys por haberme enseñado no solo por medio de teorías, sino también a través de ejemplos a querer lo que uno hace por insignificante que parezca.
- A mi hermana por ser mi amiga.
- A ese gran grupo de desarrolladores del proyecto Servicios Telemáticos, por haberme enseñado la importancia de un import.
- A las plagas, alias "Aliuska_Yumara" porque sin ellas este trabajo quedaría inconcluso y sin uso alguno.
- A la gente de mi grupo y en especial a la gente de mi apartamento.
- Al primo y a Andrés.
- A todas las amistades que de una forma u otra me han soportado a lo largo de estos 5 años.

Al tutor Ramón Alexander Anglada Martínez por habernos educado y guiado durante el proceso de desarrollo del trabajo. Por ser más que un tutor, un amigo.

DEDICATORIA

A mi familia y en especial a mis padres.

A mi hermana.

RESUMEN

Hacia la creación de un sistema de obtención de información de hardware y software en una red de computadoras está orientada la elaboración del presente trabajo de diploma. Dividido en 4 capítulos que van desde la investigación relacionada a las principales herramientas existentes en el mundo con igual propósito, hasta la implementación de la solución propuesta. El sistema propuesto se divide lógicamente en dos partes fundamentales, un servidor y un cliente o agente. El agente está destinado a la recolección de información de hardware (motherboard, bios, memoria, disco duro, procesador, tarjeta de video, tarjeta de red, tarjeta de sonido, Lector de discos DVD-CD-ROM), software (nombre de paquetes instalados, tamaño) y configuración de la PC agentes (dirección ip, nombre de dominio, nombre de máquina, usuarios, usuario logueado, etc); una PC agente es aquella en la que está instalado el agente o cliente del Sistema de Inventario de Hardware y Software. El sistema servidor recibe los inventarios enviados por los clientes, almacena la información recibida en una base de datos, analiza si el inventario es una incidencia o no, de ser incidencia verifica si existe periodo de cambio para la incidencia, si no se ha definido periodo de cambio para la incidencia, ejecuta alertas según la configuración establecida por el administrador de red responsable del control sobre los recursos informáticos. Ambos sistemas son compatibles con sistemas operativos GNU\Linux y Windows.

ÍNDICE

INTRODUCCIÓN 10
CAPÍTULO 1 "Fundamentación Teórica" 12
1.1 Introducción 12
1.2 Concepto de Inventario 12
1.3 Arquitectura Cliente – Servidor 12
1.4 Comunicación Cliente – Servidor 13
1.5 Sistemas de obtención inventarios de Hardware y Software 13
1.5.1 OCS Inventory NG 14
1.5.2 Cacic 14
1.5.3 NetSupport DNA 14
1.6 Lenguaje de Modelado 15
1.7 Metodología de desarrollo 16
1.8 Modelamiento con IDEF0 17
1.9 Herramienta CASE 18
1.9.1 Visual Paradigm 18
1.10 Lenguaje de Programación 18
1.10.1 La elección de un lenguaje 18
1.10.2 Python 19
1.10.3 Pyro 20
1.11 Herramienta para el almacenamiento de datos 20
1.12 Herramienta de Desarrollo (IDE) 21
1.12.1 IDE 21
1.12.2 EasyEclipse 22
1.13 Conclusiones 22
CAPÍTULO 2 "Características del Sistema" 23
2.1 Introducción 23
2.2 Problema y Situación problémica 23
2.3 Modelo de Negocio. 24
2.2.1 Descripción de los Procesos del Negocio empleando "to-be" 27
2.4 Objetos de automatización identificados 28
2.5 Especificación de los requisitos de software 29

2.5.1 Requerimientos funcionales de software 29
2.5.2 Requerimientos no funcionales de software 30
2.6 Modelo de Caso de Uso del Sistema 33
2.6.1 Actores del sistema 33
2.6.2 Casos de Uso del Sistema (CUS) 33
2.6.3 Diagrama de paquetes 39
2.6.4 Diagrama de Caso de Uso del Sistema (DCUS) 40
2.7 Conclusiones 42
CAPÍTULO 3 “Diseño del Sistema” 43
3.1 Introducción 43
3.2 Modelo de Diseño 43
3.2.1 Estructura general del Sistema 44
3.2.2 Estructura detallada del Sistema Agente 45
3.2.3 Estructura detallada del Sistema Servidor 46
3.2.4 Diagramas de Clases del Diseño Sistema Agente 47
3.2.5 Diagramas de Clases del Diseño del Sistema Servidor 53
3.3 Conclusiones 57
CAPÍTULO 4 “Implementación del Sistema” 58
4.1 Introducción 58
4.2 Diagrama de componentes 58
4.2.1 Diagrama de Componentes General del Sistema Agente 59
4.2.2 Diagrama de Componentes General del Sistema Servidor 67
4.3 Diagrama de despliegue 71
4.4 Conclusiones 73
5 Conclusiones 74
6 Recomendaciones 75
Referencias Bibliográficas 76

INTRODUCCIÓN

Las Tecnologías de la Informática y las Comunicaciones (TIC) evolucionan de manera que el desarrollo social en el planeta no está a la altura del actual desarrollo científico en el ámbito de la informática y las comunicaciones; todo esto impulsado por la necesidad de científicos, estudiantes y hasta el más simple obrero por resolver sus necesidades más básicas de la forma más sencilla y con el menor costo posible.

Necesidades que han convertido al mundo en una gran red de dispositivos conectados. En nuestros días es casi imposible predecir hasta qué punto pueda influir en el futuro las tecnologías informáticas. Aunque se vive en un mundo que es controlado por redes de computadoras, una amplia gama de recursos que se encuentran en las mismas no son controlados y hasta desconocidos por sus respectivos propietarios o administradores de redes, aquí surge la necesidad de herramientas que proporcionen información de hardware y software de las redes de computadoras, herramientas que informen al administrador de red sobre los cambios que son realizados sobre la red de computadoras que administra.

Son necesarias este tipo de herramientas por el tamaño desproporcionado de las redes actuales, porque los inventarios se realizan de forma manual y no engloban la parte del software, porque son pocos frecuentes y la información se registra en papeles, además de estar completamente descentralizada. Producto al poco control el hurto y las violaciones de privilegios y políticas de seguridad son muy habituales y con perspectivas de crecimiento, esto en gran medida corresponde a que la seguridad de los recursos tangibles de una computadora es atribuida a componentes de seguridad físicos; en Cuba por ejemplo, se agregan sellos para el control de acceso no autorizado a los recursos tangibles en las redes de computadoras, un sello que depende del comportamiento y los valores éticos del personal beneficiado, que no en pocos casos son los correctos. El robo o cambio de un activo tanto tangible como intangible presente en la red de computadoras puede ocasionar además del las pérdidas monetarias, de información almacenado en el mismo según el recurso, el inhabilitamiento de los servicios que preste.

Existen variedades de herramientas que realizan inventarios de hardware y software, algunas privativas y otras de software libre; de las de software libre muchas carecen de sistemas de alertas en caso de cambio de hardware o software; sin embargo los sistemas privativos son funcionalmente más completos integrando inventarios de hardware, software y sistema de gestión de alertas. Cuba como país que opta por el Software Libre, y la Universidad de las Ciencias Informática como institución dentro de Cuba que posee potencial y capacidad humana, necesita del desarrollo de una herramienta de Software Libre que realice inventarios de hardware y software y disponga de al menos un sistema de alerta, para proporcionarles a los administradores de redes cubanos una aplicación competitiva con las privadas existentes con igual propósito.

Es importante conocer que los sistemas que automatizan los procesos de inventario de hardware y software de una red de computadoras, presentan dos divisiones lógicas totalmente distintas, pero dependientes una de la otra. La primera división lógica es una aplicación que procede a la obtención del inventario de los agentes o clientes del sistema, luego gestiona todo el proceso de alertas y guarda la información recibida de los agentes. Los procesos de obtener y almacenar los inventarios de los agentes son basados en la arquitectura cliente servidor, un agente o cliente que es una aplicación instalada en la computadoras de la red, dicho agente enviará de forma periódica al servidor la información obtenida, el servidor por su parte decide que debe de realizar según la información recibida de los agentes, como por ejemplo, iniciar sistema de alertas, almacenar cambios o registrar un nuevo inventario. La segunda división lógica es la aplicación de administración que gestiona la información obtenida de los clientes asociados al sistema y configura los procesos de alertas.

Por todo lo antes expuesto se plantea como **Problema Científico** a resolver: ¿Cómo mejorar los procesos de obtención de información de hardware y software de una red de computadoras?. Definiendo como **Objeto de estudio** para esta investigación los procesos de inventario de hardware y software en una red de computadoras. Como **Campo de acción** los procesos para la obtención de información de hardware y software en la red de computadoras de la Universidad de las Ciencias Informáticas.

El **Objetivo General**: Desarrollar un sistema cliente-servidor que automatice los procesos de obtención de información de hardware y software en redes de computadoras.

En correspondencia a la problemática y el objetivo planteado se han enfocado las **tareas de investigación a:**

- Análisis del proceso actual de obtención de inventarios de hardware y software en redes de computadoras a partir de las herramientas más importantes existentes con este propósito para lograr una comprensión más exacta de la lógica del negocio que implementan estos sistemas.
- Identificación de los requisitos del sistema que cumplan con las exigencias de los administradores de redes específicamente para sistemas de inventario de hardware y software.
- Selección de las herramientas que más se adapten para el desarrollo de la aplicación.
- Identificación de los protocolos de comunicación para adaptarlos a la arquitectura cliente-servidor de sistemas de inventario de hardware y software.
- Desarrollo del sistema de obtención de información de hardware y software.

CAPÍTULO 1 “Fundamentación Teórica”

1.1 Introducción

El desarrollo de un software siempre requiere de un análisis profundo que justifique las razones por la cual es necesaria la implementación del mismo; razones como los aportes sociales o económicos que implican su existencia. La realización de un software que no presente ventajas sobre otros de igual propósito no sería otra cosa que un cúmulo de tiempo, esfuerzo y capital perdido sin sentido. Por otra parte factores como una mala selección de herramientas o tecnologías puede transformarse al final del proceso de desarrollo de software en pérdidas en la calidad del producto. Para evitar dichos errores se realiza un estudio de herramientas y tecnologías relacionadas al entorno de los sistemas de Inventario de Hardware y Software.

1.2 Concepto de Inventario

Inventario es la verificación periódica de las existencias de materiales, equipo, muebles e inmuebles con que cuenta una dependencia o entidad, a efecto de comprobar el grado de eficacia en los sistemas de control administrativo, el manejo de los materiales, el método de almacenaje y el aprovechamiento de espacio en el almacén. Por ello, los inventarios representan uno de los renglones más importantes de las empresas. Es el proceso de identificación y categorización de los recursos de información de una forma sistemática. De esta forma, se proporciona una fotografía de lo que la organización posee en términos de recursos de información en un momento determinado.

1.3 Arquitectura Cliente – Servidor

La arquitectura cliente – servidor representa una división lógica, donde la capacidad de proceso se reparte entre ambos, esto trae como ventaja un diseño mucho más claro del sistema. Esta arquitectura consiste básicamente en la comunicación entre dos nodos, un nodo cliente que realiza peticiones al nodo servidor, y este a su vez es capaz de recibir y procesar dicha información. El hecho de que la división cliente - servidor sea lógica, trae como resultado distintas características dentro de los sistemas cliente – servidor, variando en dependencia de los servicios que se quieran brindar con dicho sistema, pero hay que tener presente que la arquitectura básica es siempre la misma.

1.4 Comunicación Cliente – Servidor

La interacción entre un cliente y el servidor ocurre mediante Socket. Socket es un punto final en una conexión, los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces nombrado interfaz de programación de aplicación de sockets. (1) En la actualidad existen implementaciones de tecnologías que brindan servicios orientado a la arquitectura cliente – servidor en lenguajes como C, C++, Python o Java, pero la creciente utilización de este tipo de arquitectura ha dado como resultado el surgimiento de estándares y especificaciones que no han hecho más que duplicarse en funcionalidades y dificultar con esto la interoperabilidad, ignorando en gran medida algunos de los problemas ya resueltos, como es el ejemplo de XML-RPC (Extensible Markup Language – Remote Procedure Call) y de SOAP (Simple Object Access Protocol) que aunque hay que reconocer que han ganado popularidad en la Implementación de servicios web y sistemas distribuidos, también hay que señalar que duplican ciertas funcionalidades de RPC (Remote Procedure Call) y CORBA (Common Object Request Brooker Architecture), pero no tratan transacciones y otro aspectos más difíciles de resolver. Como resultado hay múltiples tecnologías que intentan resolver el mismo problema. Entonces ¿qué tecnología emplear?, para dar respuesta a esta pregunta debemos enfocarnos en 3 factores importantes en arquitecturas cliente – servidor, la **interoperabilidad** si lo que deseamos es ofrecer servicios desde cualquier ordenador, la **seguridad** porque es inconcebible un sistema que no asegure en todo momento la integridad de los datos que se mueven por la red y la **escalabilidad** por que los sistemas de grandes prestaciones deben ser capaz de soportar el incremento de hardware y de software en casos de desborde de servicios. Como resultado se ha elegido a Pyro (Python remote object) que no es más que un avanzado y potente sistema de tecnología de objetos distribuidos muy similar al RMI (Remote Method Invocation), este ofrece grandes posibilidades en la Implementación de objetos distribuidos, así como de conexiones seguras, garantizando con esto la seguridad en el tráfico de información.

1.5 Sistemas de obtención inventarios de Hardware y Software

Esta etapa tiene como principal objetivo el análisis de sistemas cliente–servidor recolectores de información, específicamente el hardware y software de los activos informáticos. El funcionamiento de un sistema de obtención de inventario con arquitectura cliente - servidor básicamente radica en la obtención periódica de información por el cliente en una computadora así como el procesamiento de dicho inventario por parte del servidor, inventario que es almacenado en una base de datos.

1.5.1 OCS Inventory NG

OCS Inventory es una herramienta que nos permite realizar inventario de los equipos de una red, permitiéndonos así recolectar información diariamente del hardware y el software instalado en los ordenadores, y llevar un seguimiento del mismo, OCS Inventory NG utiliza un agente que ejecuta el inventario de los equipos cliente y un servidor de administración que centraliza los resultados del inventario, permite ver los resultados del inventario y la creación de paquetes de implementación. Las comunicaciones entre los agentes y el servidor de gestión se realiza mediante los protocolos HTTP / HTTPS. Todos los datos son formateados en XML y comprimido con Zlib para reducir el tráfico en la red. También es capaz de detectar todos los dispositivos activos de la red, tales como router, impresora de red y almacenar direcciones MAC e IP, soportando casi todas las plataformas disponibles en el mercado, tales como GNU\Linux, Windows, Mac os, Sun, IBM, AIX, entre otros, su licencia es GNU General Public License, versión (GNU GPLv2). (2)

1.5.2 Cacic

El Controlador automático y colector de informaciones computacionales, Cacic, desarrollado por el gobierno de Brasil, tiene como objetivo principal realizar el inventario de hardware y software de activos informáticos interconectados en redes estructuradas, por medio de tecnologías basadas en agentes inteligentes, presenta un sistema disparador de alerta cuando se identifican cambios en componentes de hardware de cada computadora, así como la identificación de las carpetas e impresoras compartidas. Tiene soporte para las principales plataformas, como GNU\Linux (Debian, Ubuntu, RedHat), Windows y se encuentra disponible bajo la licencia GPL. (3)

1.5.3 NetSupport DNA

NetSupport DNA (Dynamic Network Administration) es una completa solución modular que ofrece inventario de hardware y de software y gestión de licencias. Presenta avisos detallados y plenamente personalizables, medición/control de uso de aplicaciones y de Internet, y actualización automática y por consulta de distribución de software, a través de una LAN (Local Area Network) o una WAN (Wide Area Network). NetSupport DNA proporciona una puerta de enlace de comunicaciones integrada que le permite interactuar con sus activos con toda seguridad, por Internet, en cualquier lugar, todo ello sin necesidad de VPN (Virtual Private Network) ni cambios en su red existente o en la configuración de cortafuegos. Con plena integración con el servicio de directorio de Microsoft Active Directory, el recurso de información y asistencia Helpdesk basado en ITIL (Biblioteca de Infraestructuras de Tecnologías de Información) opcional y el mejor control remoto del mercado. (4)

Herramientas	Agente	Sistema Operativo	Alerta	SWL	Sistema Operativo del Agente
OCS Inventory NG	si	Windows, Unix, GNU	no	si	Windows, Unix, GNU
Cacic	si	Debian, Ubuntu	si	si	Windows, GNU
NetSupport DNA	si	Windows	si	no	Windows, Red Hat

En la tabla se muestran las principales herramientas existentes que realizan inventario de hardware y software, atendiendo a 5 de los criterios más importantes a tener en cuenta a la hora de elegir un sistema recolector de información. Como se muestra en la tabla, una de las desventajas presentes en OCS Inventory NG y del CACIC es la ausencia de un sistema de alertas que sea capaz de mantener informado a los administradores de cualquier cambio ocurrido en los ordenadores clientes, además de que el inventario se hace de forma periódica y no en el momento en que tiene lugar la incidencia. En el caso de NetSupport el servidor solo se encuentra disponible para plataformas Windows y el agente está disponible solo para un pequeño grupo de plataformas, pero la mayor de las desventajas radica en que NetSupport DNA es una herramienta privativa y Cuba ha optado por el software libre como la única vía posible para su desarrollo informático, el software privativo fuerza a tomar decisiones que solo le corresponde al usuario, atenta contra las comunidades de desarrollo, impide el desarrollo local y favorece solo a unos pocos, ya que "*el software privativo como mecanismo conduce inevitablemente al monopolio*", mientras que el Software Libre fomenta el desarrollo local y se nutre de igual forma del desarrollo global, sentando las bases para un desarrollo sólido.

"La libertad sólo puede engendrar más libertad, más independencia, más soberanía.

En todos los planos: en el tecnológico, en potenciales de desarrollo, en economía, pero fundamentalmente en el desarrollo humano y profesional". (5)

1.6 Lenguaje de Modelado

UML (Lenguaje Unificado de Modelado) nace en el 1994 como fusión de los métodos y las notaciones de G. Booch, J. Rumbaugh y I. Jacobson2. Este fue propuesto como estándar en el 97, ante el OMG (Object Management Group), una asociación que se dedica a hacer propuestas de estándares para la industria. Este se ha ido imponiendo como el estándar usado por defecto para la notación no solo relacionada con el modelamiento de software o de negocios, sino también para variedades de documentos y publicaciones. Es definido nada más y nada menos que como un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas de software, así como también el modelado de negocios. Se encuentra definido por varios niveles, adecuado para los modelos orientados a objetos. Entre sus mayores atracciones esta la notación gráfica.

UML también presenta una cantidad variada de herramientas CASE empleadas para la notación, notación que cuenta con 3 grupos de diagramas:

Los Diagramas de Estructura, que muestran los elementos que deben existir en un sistema de modelado, los de Comportamiento nos muestran lo que debe ocurrir en el sistema de modelado, y los de Interacción que no son mas que una extensión de los Diagramas de Comportamiento, muestran el flujo de control y dato entre los elementos de un sistema de modelado.

UML nos da la destreza necesaria para la creación de sistemas de software que se encuentren bien diseñados, que sean robustos y de fácil mantenimiento, ya que nos da la habilidad para analizar y diseñar un sistema desde la perspectiva de los objetos.

1.7 Metodología de desarrollo

En los últimos tiempos el desarrollo de la informática ha estado emparejado a una rápida evolución de la ingeniería de software, provocada en parte por el creciente desarrollo de sistemas complejos, esto se debe en gran medida al principio conocido como “divide y vencerás”, es decir la descomposición del sistema en partes más pequeñas llamadas mini proyectos de forma tal que sea mucho más fácil de manejar. Puesto que no existe una metodología de software universal, solo metodologías ajustables de acuerdo a las características existentes de un determinado proyecto. La elección de un proceso que sea capaz de guiar de forma efectiva el desarrollo de un software está dada por la capacidad del proceso de ajustarse a las características del sistema propuesto. Razón por la que se decide seleccionar a RUP como metodología de desarrollo, por ser ideal en proyectos de gran magnitud y capaz de indicar de forma eficiente “como” construir un software que sea económico, fiable y que funcione eficientemente.

La versión estandarizada de RUP (Proceso Unificado de Rational) se da a conocer en 1998, fue el resultado de años de desarrollo y el uso práctico de técnicas de desarrollo empleando UML.

RUP es un proceso que define como principales elementos:

Quien (“***Trabajadores***”) hace que (“***Artefactos***”), como (“***Actividades***”) y cuando (“***Flujo de Actividades***”) lo hace.

RUP no solo está organizado en 9 grupos lógicos llamados flujos de trabajo donde los 6 primeros son conocidos como de Ingeniería y los 3 últimos son de apoyo, sino que también presenta 4 fases conocidas como Inicio, Elaboración, Construcción y Transición, fases por las que recorren cada uno de los flujos de

trabajos, esto ocurre de forma iterativa e incrementar, por lo que una iteración involucra actividades de todas las fases de trabajos.

Los flujos de trabajo conocidos como de ingeniería son Modelamiento del negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba y Despliegue; las distintas iteraciones por cada fase de cada uno de estos flujos traen como resultado un release del producto listo para ser entregado a los usuarios finales. El ciclo de vida de RUP se caracteriza por ser **Dirigido por casos de uso**, ya que son los que reflejan las necesidades de los usuarios, necesidades que guían el proceso de desarrollo una vez representados a través de los requerimientos, **Centrado en la arquitectura**, porque esta nos brinda los cimientos del sistema que son necesarios para un desarrollo económico, empezando por los CU más relevantes, e **Iterativo e Incremental**, cada fase de RUP es desarrollada en iteraciones o mini proyectos que involucran actividades de los flujos de trabajo, estas iteraciones dan como resultado el crecimiento del producto.

1.8 Modelamiento con IDEF0

IDEF es una serie de técnicas concebidas en los años 70 como resultado del programa llamado ICAM desarrollado por el ejército del aire de los EEUU, éste programa identificaba las necesidades de una mejora en las técnicas y el análisis de la comunicación para personal involucrado en la producción. Esta serie de técnicas se dividen en:

IDEF0: Es utilizado para la representación de las actividades o procesos.

IDEF1: Es utilizado como modelo de representación y estructuración de la información.

IDEF2: Es utilizado para representar modelos que varían con el tiempo.

En el 1983, el ejército del aire de los EEUU programó un sistema integrado de ayuda de la información que estaba basado en IDEF1 creando así el IDEF1X (IDEF1 ampliado). Actualmente IDEF0 y IDEF1X son utilizadas de forma generalizada en los sectores del gobierno, de la industria y del comercio.

Para que utilizar IDEF0:

- Como medio para comunicar las reglas y procesos de negocios.
- Para facilitar el análisis a la hora de identificar puntos de mejoras.
- Obtener una vista estratégica de un proceso.
- Para modelar una variedad amplia de sistemas automatizados y no automatizados.

- Se puede emplear para definir primero los requisitos y especificar las funciones de nuevos sistemas, y luego diseñar una puesta en práctica que se capaz de resolver los requisitos y realizar las funciones.
- En sistemas ya existentes, se puede emplear para el análisis de funciones realizadas por el sistema y registrar los mecanismos por los cuales estos son hechos.

IDEF0 ha sido elegido para el desarrollo del sistema propuesto por ser capaz de representar gráficamente una amplia variedad de negocios, así como otros tipos de operaciones de la empresa a cualquier nivel de detalle, por ser expresivo, coherente, comprensivo y simple, previendo así cualquier expresión rigurosa y promoviendo como resultado la consistencia del uso y de la interpretación.

1.9 Herramienta CASE

1.9.1 Visual Paradigm

Visual Paradigm para UML, es una herramienta de diseño que soporta todos los diagramas UML, diagramas de SysML y el diagrama entidad-relación. Visual Paradigm para UML ofrece amplias características de modelado de casos de uso incluyendo la función completa de Diagrama de casos de uso, editor de flujo de eventos, de casos de uso y la generación de diagrama de actividad. Visual Paradigm para UML produce la documentación del sistema en formato PDF, HTML y MS Word. Los desarrolladores pueden diseñar la documentación del sistema con plantillas de diseño. El analista de sistemas puede estimar las consecuencias de los cambios con los diagramas de análisis de impacto, como la matriz y el diagrama de análisis. Visual Paradigm para UML genera código Java, Python, C entre otros. Entre las principales características presente en Visual Paradigm para UML está el diseño de Diagramas de Procesos de Negocio, el Modelado colaborativo con CVS y Subversión, la interoperabilidad con modelos UML 2, la Ingeniería de ida y vuelta, la Ingeniería Inversa tanto de código a modelo como de código a diagrama y la Generación de código de modelo a código y de diagrama a código. (6)

1.10 Lenguaje de Programación

1.10.1 La elección de un lenguaje

La elección de un lenguaje de programación que sea capaz de enfrentar de forma eficiente las metas propuestas para el desarrollo de un determinado producto no siempre es sencillo, muchas veces prima la inseguridad en cuanto a sí un determinado lenguaje es mejor que otro; cada lenguaje de programación cumple un determinado propósito y algunos lenguaje lo hacen mejores que otros; no existe un lenguaje

perfecto, por lo que nuestra elección se debe limitar a aquel lenguaje que dada sus características sea el que más se adapte al desarrollo de nuestro sistema.

A pesar de toda esa búsqueda se enfoca en el análisis de dos grupos, lenguajes de bajo nivel y de alto nivel. La programación en lenguaje de bajo nivel como el lenguaje máquina o lenguaje simbólico tiene ciertas ventajas:

- Mayor adaptación al equipo.
- Posibilidad de obtener velocidades máximas con uso mínimo de memoria.

Pero como es de suponer también presenta importantes inconvenientes:

- Imposibilidad de escribir código independiente de la máquina.
- Mayor dificultad en la programación y la comprensión de los programas.

Es por esta razón que surge un nuevo tipo de lenguaje llamado de “tercera generación” o de “alto nivel” que evitaba los inconvenientes antes mencionados, a costa de ceder un poco en las ventajas. Estos lenguajes se caracterizan por el uso de macro-instrucciones, las cuales en el momento de ser ejecutadas por el computador tienen que ser traducidas a lenguaje de máquina, ya sea por medio de un compilador o intérprete.

Entre los lenguajes de alto nivel existente, han resaltado por su facilidad para el manejo de ficheros y el trabajo con sistemas distribuidos C, C++, Perl, Ruby y Python, siendo este último la elección para el sistema propuesto. Python permite dividir el programa en módulos, es un lenguaje de programación con una variada colección de módulos estándar que se pueden emplear en el desarrollo de aplicaciones, para el trabajo de E/S de ficheros, llamadas al sistema, trabajo con sockets, servicios web, objetos remotos, sistemas distribuidos, y una variedad de interfaces GUI, ahorrando con esto una cantidad considerable de tiempo en el desarrollo de programas ya que no es necesario compilar ni enlazar; presentando a diferencia de otros muchos lenguajes una sintaxis limpia y elegante.

1.10.2 Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90, nombre inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible, y es por las características que se mostraran a continuación que ha sido elegido para el desarrollo de la aplicación propuesta.

- Lenguaje Interpretado o de Script:

Python es un lenguaje Interpretado, es decir su código se ejecuta utilizando un programa intermedio que es llamado interprete, además de esto el código fuente también se traduce a un pseudo-código máquina intermedio llamado bytecode. Los lenguajes interpretados tienen como principal ventaja la flexibilidad y portabilidad.

- Tipado Dinámico:

  Se refiere a que el tipo de dato de una variable puede ser definido en tiempo de ejecución según el tipo de dato que le sea asignado a la variable, y esta puede cambiar si se le asigna otro tipo de variable.

- Lenguaje Orientado a Objeto:

  Python soporta el paradigma orientado a objeto, donde la ejecución de los programas no es más que la interacción entre objetos compuestos generalmente por clases.

  Además de esto también soporta programación funcional, imperativa y orientada a aspectos.

- Multiplataforma:

  El intérprete de Python se encuentra disponible en muchas plataformas, asegurando así su portabilidad; por lo que nuestro programa puede correr en varios sistemas sin hacer grandes cambios.

1.10.3 Pyro

Pyro fue la tecnología seleccionada para la comunicación entre cliente y servidor. Pyro es un avanzado y potente sistema de tecnología de objetos distribuidos completamente escrito en Python, que está diseñado para ser muy fácil de usar. Es muy similar a la implementación de Java Remote Method Invocation (RMI). Aunque no es muy similar a CORBA - que es un sistema independiente del lenguaje y la Tecnología de Objetos Distribuidos y tiene mucho más que ofrecer que Pyro o RMI, pero Pyro es pequeño, sencillo y esta liberado bajo la licencia MIT (*Massachusetts Institute of Technology*), por lo que permite reutilizar el software licenciado bajo MIT, tanto para crear software libre como para software no libre, permitiendo con esto no liberar los cambios que se puedan hacer sobre el software original. (7)

1.11 Herramienta para el almacenamiento de datos

Como gestor de base de datos se ha seleccionado a PostgreSQL, por ser compatible con almacenamiento de objetos binarios de gran tamaño, incluyendo imagines, música y videos. Presenta interfaces de

programación nativa para C / C + +, Java, .NET, Perl, Python, Ruby, Tcl, ODBC, entre otros, además de una magnifica documentación.

PostgreSQL es una base de datos de clase empresarial, que cuenta con sofisticadas funciones como la versión multi-Control de concurrencia (MVCC), respaldo incremental, espacio de tablas, replicación asincrónica, transacciones anidadas (puntos de retorno), backups en caliente y un planificador de consultas sofisticadas / optimizador. Es compatible con conjuntos de caracteres internacionales, codificación de caracteres multibyte, Unicode, y es consciente de la configuración regional para la clasificación y el formato. Además de que PostgreSQL es un gestor objeto-relacional de código abierto sumamente potente e ideal para sistemas grandes y de un mayor nivel que MySQL, comparable en algunos aspectos a Oracle, Sybase o Interbase. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar como en el número de usuarios concurrentes que puede acomodar. Aunque el tamaño límite de una base de datos de PostgreSQL es ilimitado, no existe un sistema activo de PostgreSQL en entornos de producción que manejen un exceso de 4 terabytes de datos. Algunos de los límites generales de PostgreSQL se incluyen en la tabla mostrada abajo. (8)

Limite	Valor
Tamaño máximo de la base de datos	Ilimitado
Tamaño máximo de la Tabla	32 TB
Tamaño máximo de Fila	1,6 TB
Tamaño máximo de Campo	1 GB
Máximo de Filas por Tabla	Ilimitado
Máximo de Columnas por Tabla	250 - 1600 dependiendo de los tipos de columna
Índices máximo por Tabla	Ilimitado

“El código fuente de PostgreSQL se encuentra disponible bajo la licencia de código abierto: BSD. Esta licencia le da la libertad para usar, modificar y distribuir PostgreSQL, en la forma que más te guste. Puedes realizar cualquier modificación, mejora o cambio”. (9)

1.12 Herramienta de Desarrollo (IDE)

1.12.1 IDE

Un entorno de desarrollo integrado o IDE (Integrated Development Environment), es un conjunto de herramientas de programación integradas en un único entorno, entre las que destacan el editor de código, un compilador, un depurador, una consola y en algunos caso un constructor de interfaz gráfica. Este puede servir a un único lenguaje de programación o varios, como ocurre con EasyEclipse, una plataforma

que mediante el uso de plugins se le añaden nuevas funcionalidades o soportes para lenguajes de programación como Java, Python, Perl, C, C++, entre otros. (10)

1.12.2 EasyEclipse

El proyecto EasyEclipse está considerado por su robustez uno de los mejores IDE, esta plataforma tiene un variado número de plugins o extensiones útiles para programar en lenguajes como (Python, C, Java, LAMP, PHP o Ruby on Rails). Como es el caso de Pydev, una extensión que aprovecha las funcionalidades de EasyEclipse para explotarlas con Python, este nos brinda ayuda con el código ya que posee un auto completamiento bastante potente, Análisis de sintaxis y del código, Depurador remoto y una consola interactiva. Pydev se encuentra bajo la licencia EPL (Eclipse Public License). (11)

1.13 Conclusiones

A partir del análisis realizado durante el desarrollo del capítulo fueron demostradas algunas deficiencias en los sistemas estudiados como Casic, OCS Inventory NG y NetSupport DNA que justifican la creación de una nueva herramienta que permita la obtención de información de hardware y software, sin estas deficiencias que reflejan desde carencia de funcionalidades hasta problemas de soporte e incluso económicos. La elección de UML como lenguaje de modelado, RUP como metodología de desarrollo, IDEF0 para el modelamiento del negocio así como Visual Paradigm como herramienta CASE y Python como lenguaje de programación soportados por un IDE como EasyEclipse deben transformarse en calidad, reducción de tiempo y costos al finalizar el proceso de desarrollo.

CAPÍTULO 2 “Características del Sistema”

2.1 Introducción

Una visión común entre actores del negocio y desarrolladores del futuro sistema, es parte importante del éxito de una aplicación o un software que cuya materialización este fundamentada en la solución de un problema real; un problema que afecta o interactúa con la sociedad. Con objetivo de establecer una visión general de lo que el sistema debe de hacer según las prioridades de los actores del negocio, se establece la necesidad del desarrollo del presente capítulo, que guíe a la determinación de los objetivos del sistema; basándose en la descripción detallada de los procesos presentes en el negocio.

2.2 Problema y Situación problémica

Es difícil el control de los recursos disponibles en una red de computadoras cuando la magnitud de las mismas crece diariamente. Poco control de estos recursos ocasionan perdidas de los mismos, que implica pérdidas económicas así como de los servicios que prestan o información vigente en estos. La manera más económica y eficiente de establecer un control sobre los dispositivos asociados a una red de computadoras es a través de un sistema automatizado que esté constantemente vigilando los medios informáticos de la red. La existencia de estas herramientas de administración, centralizan toda la información de hardware así como de software de una red de computadoras.

Actualmente en una red de computadoras donde se realiza el inventario y el proceso no esté automatizado como en la Universidad de las Ciencias Informáticas; un grupo de trabajadores o un trabajador según la magnitud de la red, se presenta(an) en el local donde están físicamente los recursos, realizan el inventario de hardware específicamente, no de software, se escribe en papeles que luego son archivados en una local dispuesto para esto. La seguridad de los recursos se le asigna a objetos materiales como alarmas instaladas en los locales donde se encuentran o aplicando un sello se seguridad. Realizar un inventario manualmente es un problema para el personal que administra una red por las pérdidas económicas así como el retardo en la detección de incidencia a menos que lo hurtado preste algún servicio importante y detectable con facilidad, como por ejemplo servicios de correo electrónico.

Se conoce que existen múltiples herramientas explicadas en el anterior capítulo que automatizan los procesos de inventario de hardware y software como Cacic, OCS Inventory o NetSupport DNA, unas privativas otras de software libre. Cuba es un país que no dispone de capital financiero suficiente como

para invertir en la compra o el pago de licencias para software privativo, por lo que a los administradores de redes cubanos se le hace difícil disponer de herramientas privativas que realicen los procesos de inventario. Por lo que la apuesta por el software libre es una necesidad cubana, que es evidente en los últimos años. La carencia de funcionalidades en las herramientas de software libre como sistemas de alerta en caso de detección de cambios en el hardware o software, impulsa el desarrollo de una herramienta destinada al control de hardware y software en una red de computadoras por parte de los administradores cubanos.

2.3 Modelo de Negocio.

Seguidamente se realiza un modelado detallado de que debe de hacer el sistema de inventario de hardware y software a partir del estudio de los software tanto privativos como libres, enfocado en las necesidades de los administradores de redes cubanos; modelado con la notación IDEF0 efectiva en el modelado de procesos del negocio, en este caso los procesos identificados para realizar un inventario de hardware y software en una red de computadoras.

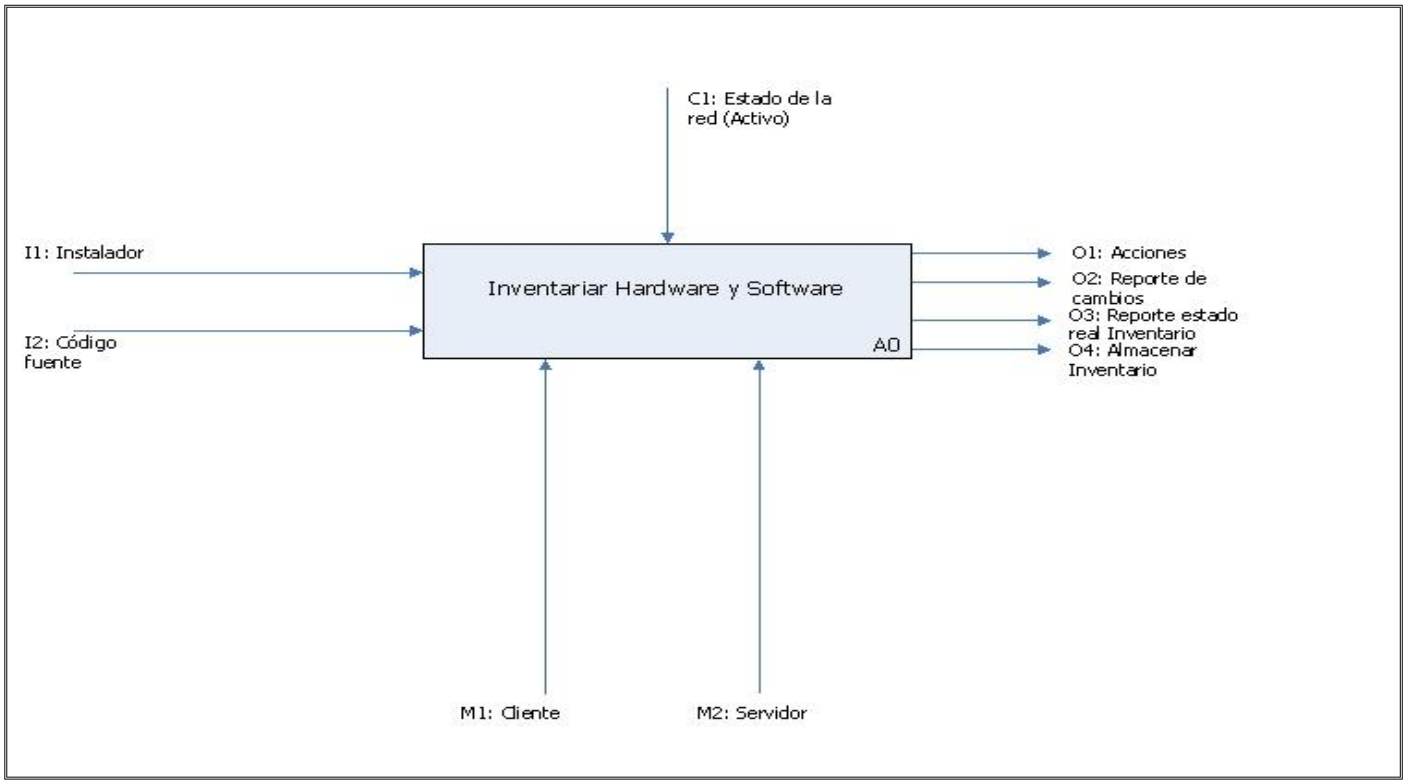

Fig 1: Proceso base "***Inventariar Hardware y Software***" modelado empleando **to-be**

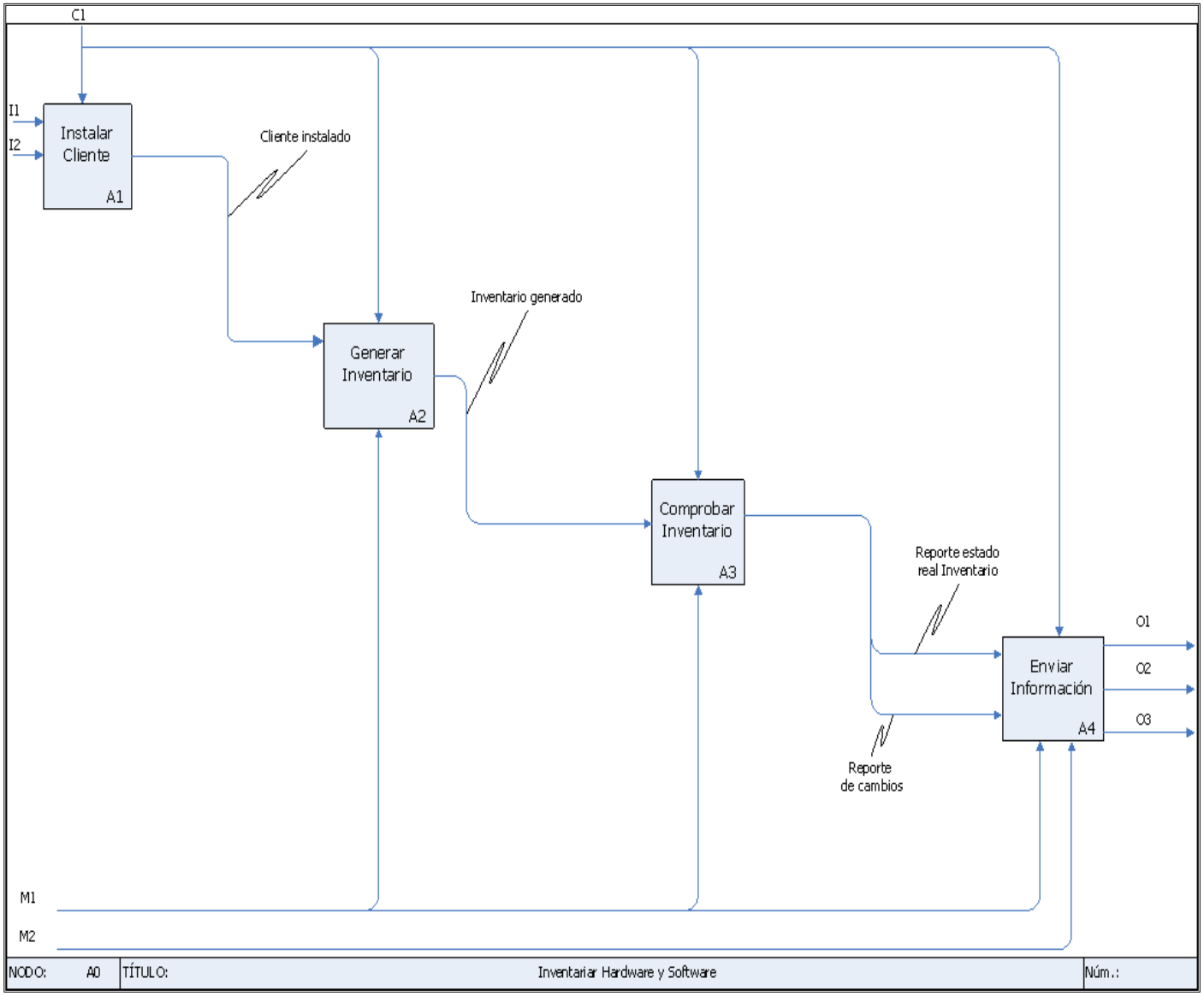

Fig 2: Modelado detallada del proceso base "***Inventariar Hardware y Software***" empleando **to-be**

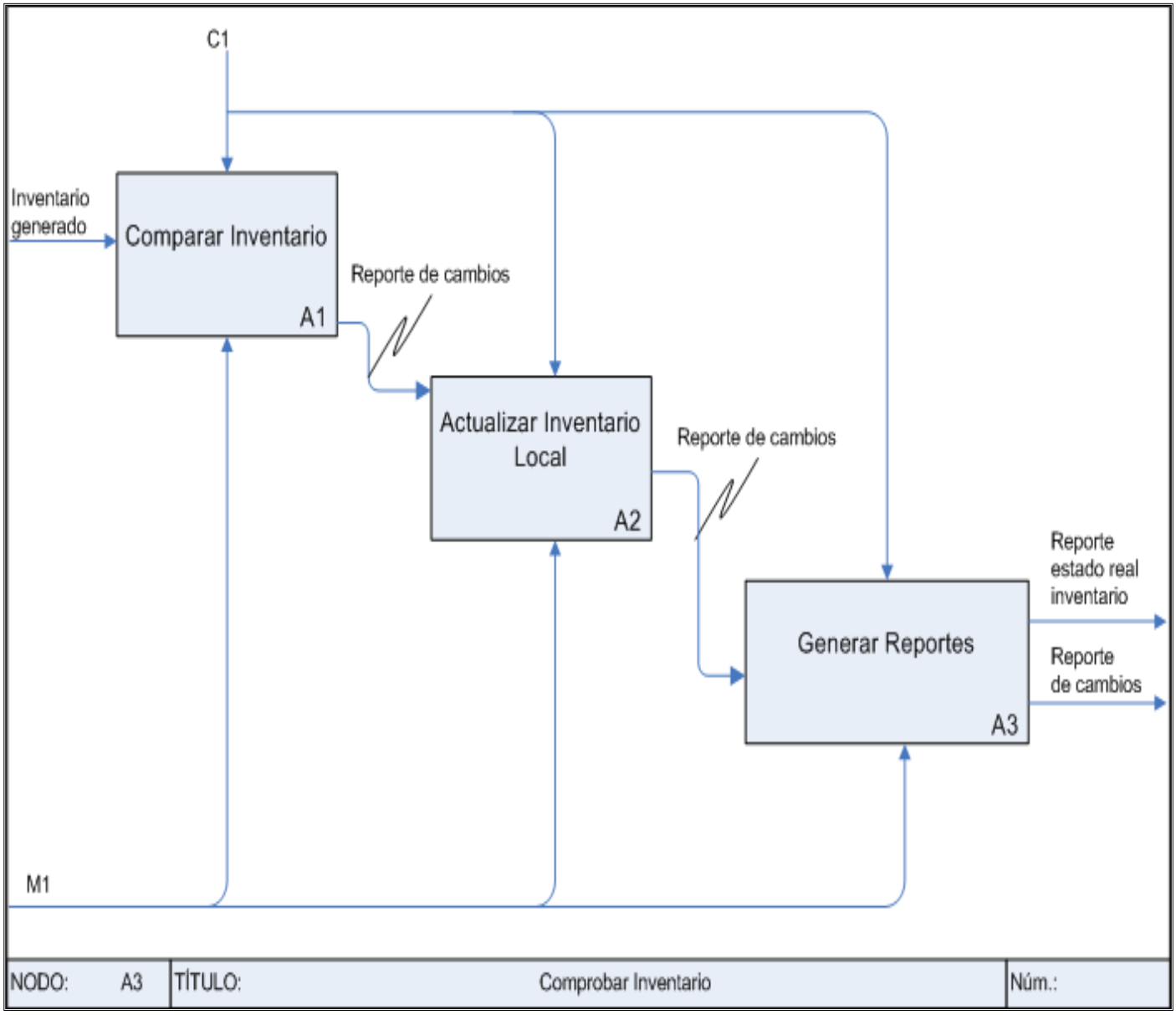

Fig 3: Modelado detallada del proceso "***Comprobar Inventario***" empleando **to-be**

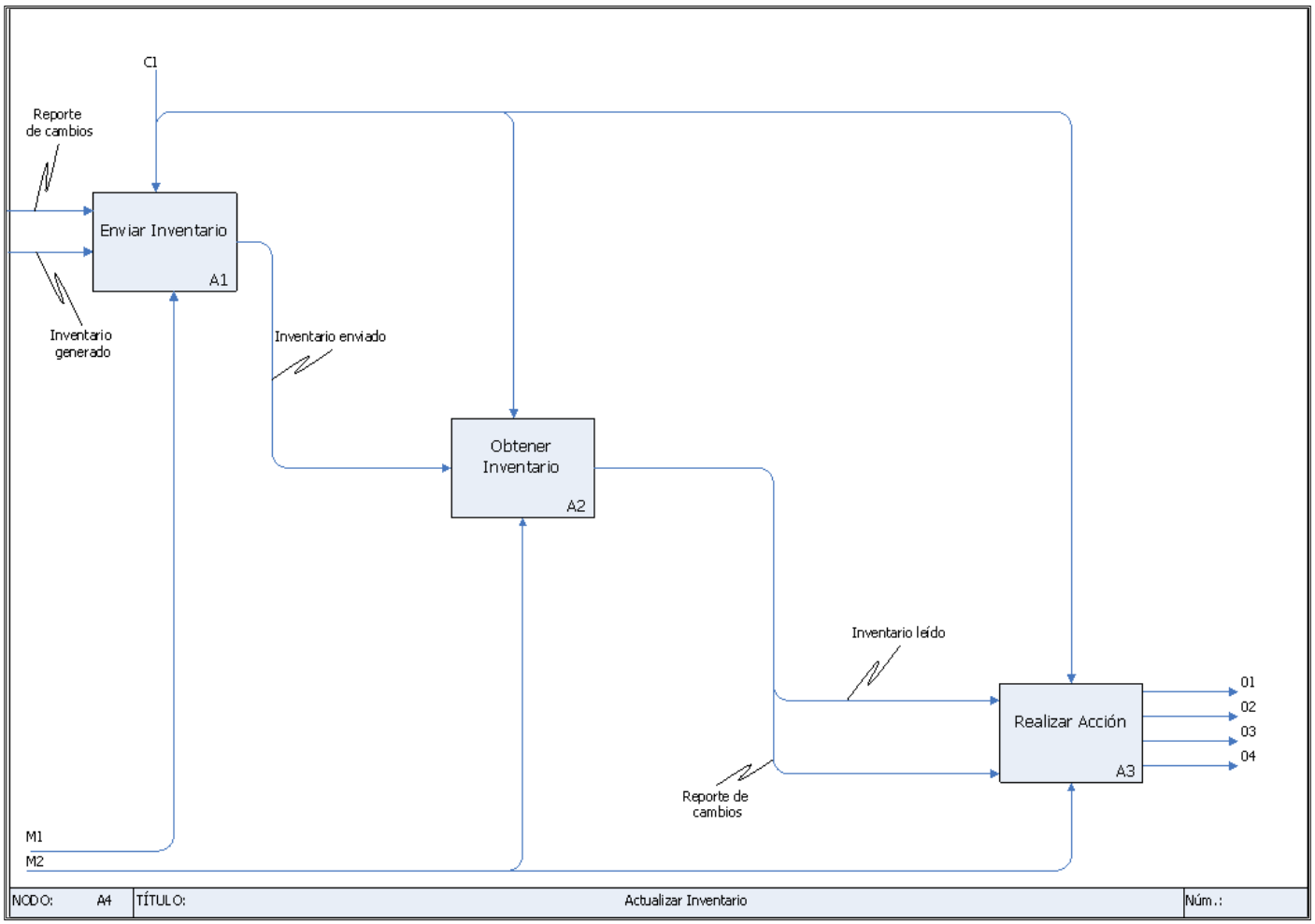

Fig 4: Modelado detallada del proceso **"*Actualizar Inventario Local*"** empleando**to-be**

2.2.1 Descripción de los Procesos del Negocio empleando "to-be"

2.2.1.1 Descripción del proceso base Inventariar Hardware y Software

- El proceso base **Inventariar Hardware y Software** empleando to-be no es más que la intervención de dos aplicaciones, una cliente y otra servidor, donde con la primera instalada en cualquier ordenador se puede conocer los componentes del hardware y los software que tienen instalados, permitiendo identificarlos como puede ser, por ejemplo, su número de serie; que se pueda conocer los cambios ocurridos a través de reportes; tener un sistema de alarma que informe las violaciones o robos a través de correo electrónico así como la información centralizada y actualizada del software y hardware existente en una red LAN.
- **Instalar cliente**: El proceso consiste en instalar una aplicación cliente con el objetivo de que ésta obtenga toda la información del ordenador donde se instala para enviarla luego a la aplicación servidor.

- **Generar inventario:** El cliente genera un inventario con toda la información de la máquina, el cual es enviado posteriormente a una etapa de comprobación al sistema servidor.
- **Comprobar inventario:** El cliente obtiene el inventario generado y procede a comprobarlo con el viejo inventario guardado en la cache, en caso de no coincidir se genera un reporte con los cambios.
- **Enviar Información:** El cliente establece conexión con el servidor enviando el Inventario o el Reporte de los cambios, una vez que el servidor obtiene el Inventario o Reporte de los cambios procede a almacenar en la base de datos y realizar una determinada acción en correspondencia de los datos obtenidos.

2.2.1.2 Descripción del proceso Comprobar Inventario

- **Comparar Inventario:** Luego de que el Cliente genere el inventario local, es comparado con el reporte viejo almacenado en la máquina Cliente, en caso de encontrar cambios este genera un Reporte de cambios.
- **Actualizar Inventario Local:** Este proceso se ejecuta en caso de que se encuentren cambios, procediendo entonces a actualizar el inventario local.
- **Generar Reportes:** Después de obtener el Inventario general o el Reporte con los cambios, el Cliente generar un inventario de cambios que es considerado como un reporte, que luego es enviado al servidor.

2.2.1.3 Descripción del proceso Actualizar Inventario

- **Enviar Inventario:** Luego de que el Cliente genere el Reporte, este lo envía hacia la Servidor.
- **Obtener Inventario:** El Servidor obtiene el Reporte enviado por el Cliente.
- **Realizar Acción:** Después de obtener el Inventario general o el Reporte con los cambios, el Servidor Realiza una determinada acción, en correspondencia del Reporte obtenido, como el almacenamiento en una Base de Datos.

2.4 Objetos de automatización identificados

Luego de realizar el modelado con la notación IDEF0 empleando to-be, es decir lo que debe de hacer el sistema, se identificaron los procesos a ser automatizados. El proceso base Inventariar hardware y software requiere de la automatización de todos los subprocesos que incluye en su realización.

La ejecución integra del proceso base desenlaza un número de subprocesos ya expuestos que concluyen con la creación de un inventario de hardware y software o alerta de correo electrónico ocasionadas por la detección de incidencia sobre un objetivo identificado dentro de un inventario. Los procesos mostrados

durante el modelado están orientados a la creación de un sistema que soluciones las necesidades de los administradores de redes cubanos. Dichos procesos son:

1. **Inventariar Hardware y Software:**
   - Instalar cliente
   - Generar inventario
   - Comparar Inventario
   - Enviar Información
2. **Comprobar Inventario:**
   - Comparar Inventario
   - Actualizar Inventario Local
   - Generar Reportes
3. **Actualizar Inventario:**
   - Generar Reportes
   - Actualizar Inventario
   - Realizar Acción

2.5 Especificación de los requisitos de software

Modelar el negocio permite comprender la estructura y dinámica del entorno donde será aplicado el sistema que se desea desarrollar, de manera que defina los objetivos en los cuales se guía el desarrollo del sistema. Crea una visualización concreta y única del negocio entre los clientes o trabajadores del negocio y los futuros desarrolladores de la aplicación. Conocer los problemas que presenta y las futuras mejoras que podrían aplicarse. En referencia al sistema de inventario de Hardware y Software que no presenta un negocio como normalmente es encontrado, sino que es un sistema que es adaptable a necesidades de múltiples clientes en este caso a administradores de redes. Después de modelar el negocio con la notación de modelado IDEF0 empleando “to-be” se encontraron un conjunto de procesos que para darle cumplimiento se derivan un conjunto de requisitos funcionales y no funcionales.

2.5.1 Requerimientos funcionales de software

Los requisitos funcionales no son más que condiciones o capacidades funcionales que el sistema debe de tener. Definen actividades internas del software, como manipulación de datos, flujos de información en dependencias de situaciones predefinidas por eventos lógicos. A continuación se muestran los requisitos funcionales del módulo “Obtención de Información del Sistema de Inventario de Hardware y Software”.

2.5.1.1 Módulo Obtención de Información

El Módulo Obtención de Información consiste en una aplicación cliente servidor que su objetivo principal es centralizar en el servidor la información del hardware y el software instalados en las estaciones clientes.

2.5.1.2 Requerimientos funcionales módulo Obtención de Información "Sistema Agente"

EL objetivo principal de la aplicación cliente es obtener la información de hardware y software instalados, enviarla al servidor y determinar si hubo cambios en el hardware o el software a partir del inventario obtenido anteriormente.

- RF1. Generar inventario de hardware y software.
- RF2. Guardar inventario generado
- RF3. Determinar incidencia
- RF4: Enviar inventario o incidencia.
- RF5. Instalar aplicación agente.

2.5.1.3 Requerimientos funcionales módulo Obtención de Información "Sistema Servidor"

Su objetivo principal es el de almacenar en un gestor de base de datos determinado (PostgreSQL, MySQL, Oracle, etc), toda la información de los inventarios e incidencias enviadas por los clientes, en caso de que exista algún cambio en el inventario, denominado incidencia, se ejecutará una determinada acción, pudiendo ser un correo electrónico, alertando al administrador de los cambios realizados.

- RF1. Recibir inventario o incidencia.
- RF2. Almacenar información de cada inventario o incidencia recibidos.
- RF3. Tomar una determinada acción por tipo de incidencia.
- RF5. Instalar aplicación servidor.

2.5.2 Requerimientos no funcionales de software

Los requisitos no funcionales de software son propiedades o cualidades que el producto debe de tener. Pueden ser característica que hacen al producto atractivo, usable rápido o confiable, complementan en ocasiones requisitos funcionales para concretar casos de uso, o incluso regulan funciones de requisitos funcionales. Pueden definir el éxito final de un sistema.

2.5.2.1 Requerimientos no funcionales del módulo Obtención de Información

Requerimientos de software:

- Las estaciones de trabajo donde se instalará cualquiera de los sistemas tanto agente como servidor deben tener asociada algún sistema operativo de la familia GNU\Linux o Windows.
- El proceso de instalación de servidores requiere que los ordenadores que van a hospedar el servidor contengan un compilador de paquetes apropiado.

Requerimientos de hardware:

Los requisitos de hardware del sistema servidor están asociados fundamentalmente al tamaño de la red y a la cantidad de sistemas agentes que recibirán el servicio. Es importante acotar que los servidores debido a su funcionalidad específica de atender las solicitudes de un grupo de agentes necesitan de una estructura de hardware superior que les permita dar cumplimiento a su labor. Si se tratase de una pequeña red, con servicio para algunas decenas de usuarios, podríamos emplear:

- Microprocesador Pentium IV.
- 512 MB de memoria RAM.
- 2 GB de espacio libre en el disco duro.

Incluso pudiera operar correctamente en condiciones inferiores; sin embargo, si el tamaño de la red aumentara y se tratase ya de algunos miles de usuarios, el cambio más significativo estaría en aumentar la capacidad de almacenamiento empleando discos duros como una de las mejores alternativas para las grandes empresas, además debe emplearse un procesador con capacidad de multiprocesamiento. Por otra parte se hace necesario aumentar la capacidad de memoria RAM hasta un rango de 1 a 4 GB.

En caso del sistema agente no requiere de muchos recursos para su ejecución bastaría con:

- Microprocesador Pentium III.
- 256 MB de memoria RAM.
- 20 MB de espacio libre en el disco duro.

Restricciones en el diseño y la implementación:

Este tipo de requerimiento especifica o restringe la codificación o construcción de nuestro sistema, constituyen restricciones que deben ser cumplidas estrictamente.

- Será empleado python 2.5 como lenguaje de programación para la construcción del sistema.
- La codificación estará regida por la guía de estilo propuesta por Guido van Rossum creador del lenguaje de programación python.
- Las herramientas utilizadas en la codificación aunque python permite extender su codificación a los más simples editores de texto, se ha decidido emplear Eclipse con Pydev. Pydev es un plug-in o complemento para el entorno de desarrollo Eclipse que permite la programación en python; estos complementos constituyen una forma de expandir programas de forma modular, de manera que se puedan añadir funcionalidades sin afectar a las ya existentes.

- Como arquitectura general del sistema se ha concluido la utilización de una arquitectura en capas debido a las ventajas que ello proporciona en términos de claridad, facilidad en la corrección de errores producidos durante el desarrollo, rápido desarrollo al permitir el trabajo paralelo en cada una de las capas definidas, entre otros factores.
- Sería realmente ineficiente si en cada ocasión que los desarrolladores de sistemas informáticos se proponen una nueva meta deban comenzar la elaboración de su producto desde la base, sobretodo en casos donde se pueden reutilizar muchas funcionalidades. Con el objetivo de resolver la problemática anterior han sido creadas bibliotecas de clases o marcos de trabajos que faciliten el desarrollo y acorten los tiempos de producción, permitiendo a los desarrolladores concentrarse en aspectos de mayor nivel. El sistema propuesto se ha apoyado en un marco de trabajo desarrollado para resolver problemas puntuales asociados a la gestión de servidores.

Requerimientos de seguridad:

Los requerimientos asociados a la seguridad es uno de los cuales provocaría los mayores riesgos si no se manejan correctamente. Nuestro sistema se ha centrado en:

- Evitar que la información del inventario almacenado en las computadoras agentes y sistema servidor no sea contaminada con información o perdida de información. Esta información será almacenada de forma serializada.
- Las aplicaciones deben ser instaladas con privilegios de administración.
- Para el envío del inventario desde los agentes al servidor se empleó el protocolo Pyro al cual se le asocia un mecanismo de encriptación o cifrado, en este caso SSL(Secure Sockets Layer).

Requerimientos de usabilidad:

Este tipo de requerimientos describen los niveles apropiados de usabilidad, dados los usuarios finales del producto, para ello deben revisarse las especificaciones de los perfiles de usuarios y sus niveles de experiencia; en nuestro caso los administradores de redes. Debe conocerse además que tipo de producto necesitan y qué tipo de requisito haría el producto adecuado para ellos. Por tanto, se han definido de la siguiente manera:

- Los sistemas deben de estar bien documentados, con el fin de lograr un mejor uso de los servicios que estos ofrecen, para ello se realizará un documento de ayuda que explicará las funcionalidades del sistema así como donde entran a interactuar los clientes o administradores de redes con el sistema aclarando que los sistemas tanto agentes como servidor serán aplicaciones instaladas para que corran como demonios o .procesos en las computadoras sin interactuar directamente con el cliente.

Requerimientos de soporte:

Los requerimientos de soporte están orientados a todas las acciones a tener en cuenta una vez que se ha terminado, desarrollado el software con motivos de asistir a los clientes, lograr mejoramientos progresivos y evolutivos en el tiempo. Pueden incluir: Pruebas, Extensibilidad, Adaptabilidad, Mantenimiento, Compatibilidad, Configuración, Servicios, Instalación, Internacionalización y Requisitos de Portabilidad

- Los sistemas tendrán un periodo de pruebas donde se identificará los posibles errores durante el proceso de desarrollo del producto integrado, es decir sistema agente y sistema servidor, antes de prestar servicios.
- Se realizara la documentación por separado para sistema agente y sistema servidor adecuada que facilite a los administradores de redes los procesos de configuración del sistema

Requerimientos legales:

- El sistema agente así como el sistema servidor propiedad exclusiva de la Universidad de las Ciencias Informáticas (UCI).

2.6 Modelo de Caso de Uso del Sistema

2.6.1 Actores del sistema

Actor	Descripción
PC Agente	El sistema agente representa a una PC en la cual debe de estar instalada la aplicación agente, es la que inicia el proceso de inventario de hardware y software sobre dicha PC en dependencia de un evento esperado como puede ser el inicio de la misma o desconexión o conexión de un dispositivo de entrada por USB Eje: Teclado o ratón.
PC Servidor	El sistema servidor representa a una PC servidora en la cual presta servicios el servidor correspondiente al sistema agente-servidor del sistema de Inventario de Hardware y Software. El sistema servidor inicializa las actividades en el servidor una vez iniciada la PC servidora.

2.6.2 Casos de Uso del Sistema (CUS)

Encontrar los casos de uso correctos y definir su nivel de prioridad son pasos elementales para efectuar de forma iterativa e incremental un buen desarrollo de software que merite en calidad del producto reducción de tiempo y costo. Los casos de uso deben de responder a requisitos funcionales del futuro producto.

A partir del proceso de modelado del negocio y la especificación de requisitos funcionales fueron identificados 13 casos de uso reflejados a continuación:

2.6.2.1 Casos de Uso Módulo Obtención de Información

Sistema Agente:

- Buscar Información de Hardware
- Buscar Información de Software
- Buscar Información de PC
- Realizar Inventario
- Determinar Incidencia
- Gestionar Cache
- Enviar Inventario al Servidor
- Obtener Configuración
- Procesar Inventario

Sistema Servidor:

- Buscar Inventario
- Recibir Información de Agentes
- Guardar Inventario
- Guardar Incidencia
- Ejecutar Acción

2.6.2.1 Descripción detallada de casos de Uso Módulo Obtención de Información

Sistema Agente:

Descripción detallada de Casos de Uso del Sistema Agente	
CU1	**Buscar Información de Hardware**
Actor	
Descripción	El caso de uso Buscar Información de Hardware consiste en la recolección de la información de hardware de la PC Agente la cual incluye información de: • Motherboard • Memoria • Disco Duro • Controladores

	• Monitor • Teclado • Ratón • Tarjeta de Video • Tarjeta de Sonido • Tarjeta de Red • CD-ROM • Procesador • Bios El caso de Uso es invocado cuando se realiza un inventario de Hardware y Software (Incluido de Realizar Inventario).
Referencia	**RF1**
CU-2	**Buscar Información de Software**
Actor	
Descripción	El caso de uso Buscar Información de Software consiste en la recolección de información asociada a los paquetes instalados en la PC Agente, esta incluye: • Tamaño del paquete • Nombre del paquete Así como información referente al sistema operativo en el que corre la aplicación: • UUID • Número de Serie • Versión • Nombre de Producto • Wake up Type El caso de Uso es invocado cuando se realiza un inventario de Hardware y Software (Incluido de Realizar Inventario).
Referencia	**RF1**
CU-3	**Buscar Información de PC**
Actor	

Descripción	El caso de uso Buscar Información de PC consiste en la recolección de información relacionada con la PC Agente como : • Dirección IP • Dirección de Puerta de Enlace • Máscara de Sub-Red • Dirección Mac • Nombre de la PC Agente • Usuarios de PC Agente El caso de Uso es invocado cuando se realiza un inventario de Hardware y Software (Incluido de Realizar Inventario).
Referencia	**RF1**
CU-4	**Realizar Inventario**
Actor	
Descripción	El caso de Uso realizar Inventario consiste en la creación de un inventario. Un inventario es la unión de la información recolectada por los casos de uso Buscar Información de Hardware, Buscar Información de Software y Buscar Información de PC. Es un incluido de (Procesar Inventario)
Referencia	**RF1**
CU-5	**Determinar Incidencia**
Actor	
Descripción	El caso de Uso Determinar Incidencia consiste en comprobar si ha cambiado la información de hardware o software en la PC Agente. El caso de uso es realizado a partir del nuevo inventario realizado y el inventario almacenado en la cache, realiza una comparación de datos y determina que ha cambiado, procede a crear un nuevo inventario con los cambios para enviarlo al sistema servidor. El caso de uso está condicionado por la existencia de un inventario en la cache del Agente.

Referencia	**RF3**
CU-6	**Gestionar Caché**
Actor	
Descripción	El caso de uso Gestionar Caché consiste en el almacenamiento y obtención del inventario según el caso. El almacenamiento no es más que la serialización del inventario previamente realizado en la cache del Agente. La obtención consiste en la lectura del inventario para posteriormente realizar una comparación en busca de incidencia por el caso de uso Determinar Incidencia.
Referencia	**RF2**
CU-7	**Obtener Configuración**
Actor	
Descripción	El caso de uso Obtener Configuración es responsable de la lectura de la información relacionada a las configuraciones para la conexión con el sistema servidor. La configuración está formada por los datos: • Puerto • Host • Usuario • Contraseña El caso de uso es uso es un incluido de (Enviar Inventario al Servidor)
Referencia	**RF4**
CU-8	**Enviar Inventario al Servidor**
Actor	
Descripción	El caso de uso Enviar Inventario al Servidor consiste en la conexión al sistema servidor a partir de la lectura de la configuración, y posterior envío del inventario al servidor. El sistema puede enviar dos tipos de inventario, uno completo u otro en caso de incidencia. La obtención de la configuración de conexión es responsabilidad del caso de uso Obtener

	Configuración. El caso de uso es extendido de (Procesar Inventario)
Referencia	**RF4**
CU-9	**Procesar Inventario**
Actor	**PC Agente**
Descripción	El caso de uso Procesar Inventario es el responsable de controlar e iniciar los procesos en el sistema servidor. Identifica que flujo de eventos debe de realizar el sistema ante algún tipo de evento. El caso de uso primeramente procede a la realización del inventario de hardware y software, luego verifica si existe un inventario almacenado en caché, de haberlo, obtiene el inventario guardado ("Gestionar Caché") e inicia el caso de uso "Determinar Incidencia", si existe incidencia procede al envío del inventario al sistema servidor (caso de uso "Enviar Inventario al Servidor") de no existir inventario almacenado en caché envía el inventario realizado al sistema servidor.
Referencia	**RF1, RF2, RF3, RF4, RF5**

Sistema Servidor:

Descripción detallada de Casos de Uso del Sistema Servidor	
CU-1	**Buscar Inventario**
Actor	
Descripción	El caso de uso Buscar Inventario consiste en la búsqueda de inventario en la base de datos del sistema. La búsqueda es realizada a partir de un identificador de activo (dirección mac de la tarjeta de red en la PC agente)
Referencia	**RF2, RF3**
CU-2	**Recibir Información de Agentes**
Actor	**Sistema Servidor**
Descripción	El caso de uso Recibir Información de Agentes consiste en que luego que el sistema agente envía un inventario el sistema servidor almacena en caché el inventario recibido.

Referencia	**RF1**
CU-3	**Guardar Inventario**
Actor	**Sistema Servidor**
Descripción	El caso de uso Guardar Inventario consiste en la verificación de existencia u existencias de inventario o inventarios almacenado en caché de la PC Servidor, de existir establece conexión con la base de datos del sistema y procede a la inserción de la información.
Referencia	**RF2**
CU-4	**Guardar Incidencia**
Actor	**Sistema Servidor**
Descripción	El caso de uso Guardar Incidencia consiste en determinar la incidencia almacenada en la cache, para luego procesarla y almacenarla en la base de datos, en caso de no estar especificada dentro de un periodo de cambio, se ejecuta una acción determina.
Referencia	**RF3,RF4**
CU-5	**Ejecutar Acción**
Actor	**Sistema Servidor**
Descripción	El caso de uso es iniciado cuando se recibe una incidencia desde un agente. El sistema establece conexión con la base de datos e identifica que tipo de incidencia están presentes en el inventario recibido. Comprueba si en la base de datos hay período de cambio para tipo de incidencia presentes en el inventario, de no existir período de cambio obtiene información de alerta para la incidencia y procede al envío de correo electrónico a partir de la información obtenida de la alerta. Si para el tipo de incidencia identificado en el inventario existe período de cambio no se efectúa acción de alerta.
Referencia	**RF4**

2.6.3 Diagrama de paquetes

Lo diagramas de paquetes agrupan lógicamente las divisiones de un sistema, representando las dependencias entre sus agrupaciones. Es una forma de organizar funcionalmente los casos de uso del

sistema. El siguiente diagrama muestra las divisiones lógicas el Sistema de Inventario de Hardware y Software.

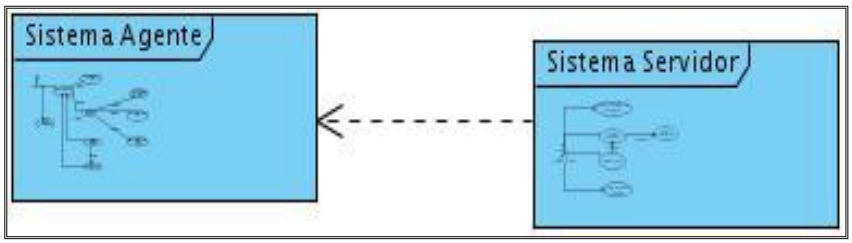

Fig 5. Diagrama de paquetes del sistema.

2.6.4 Diagrama de Caso de Uso del Sistema (DCUS)

2.6.4.1 Diagrama de Caso de Uso del Sistema Servidor

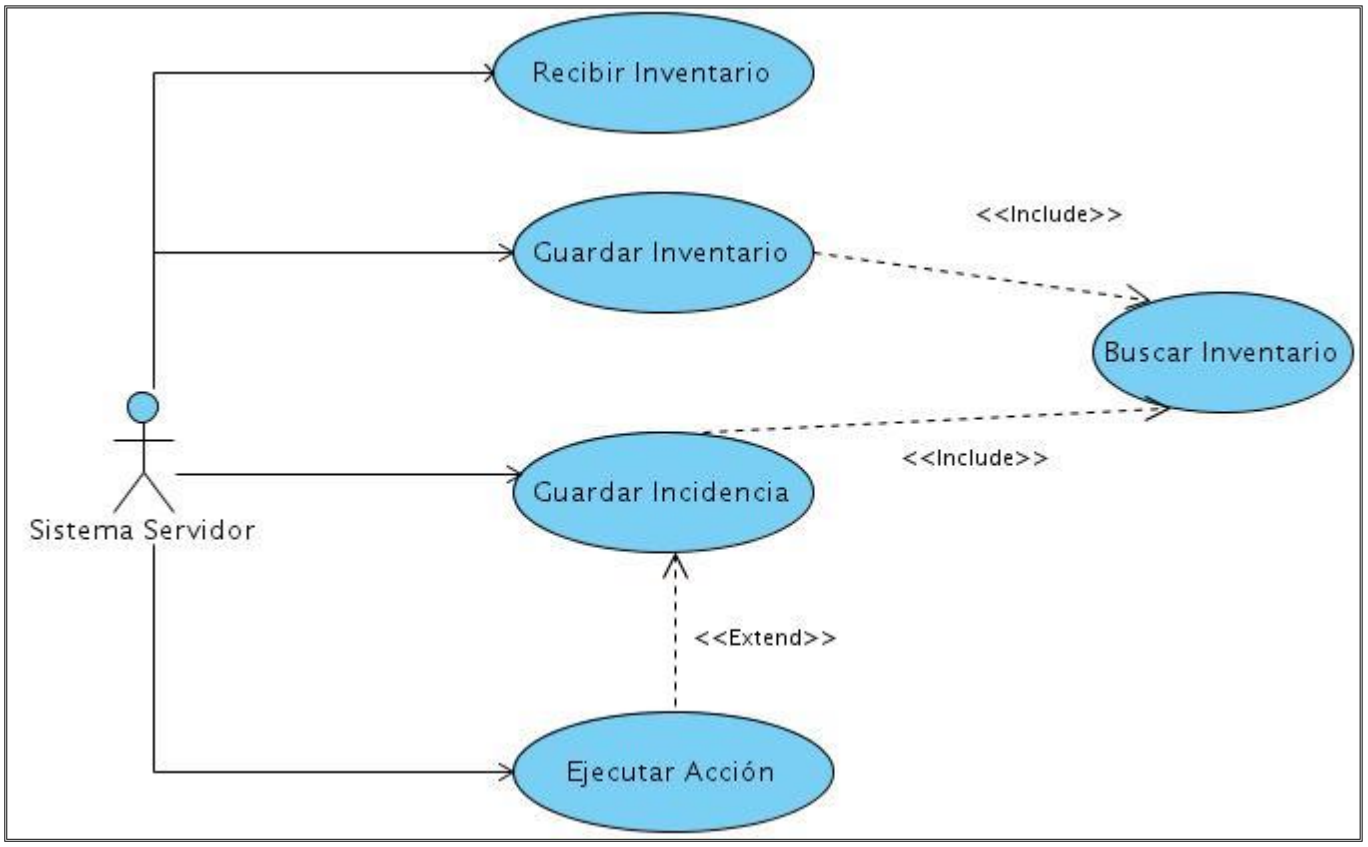

Fig 6. Diagrama de Caso de Uso del Sistema Servidor

2.6.4.2 Diagrama de Caso de Uso del Sistema Agente

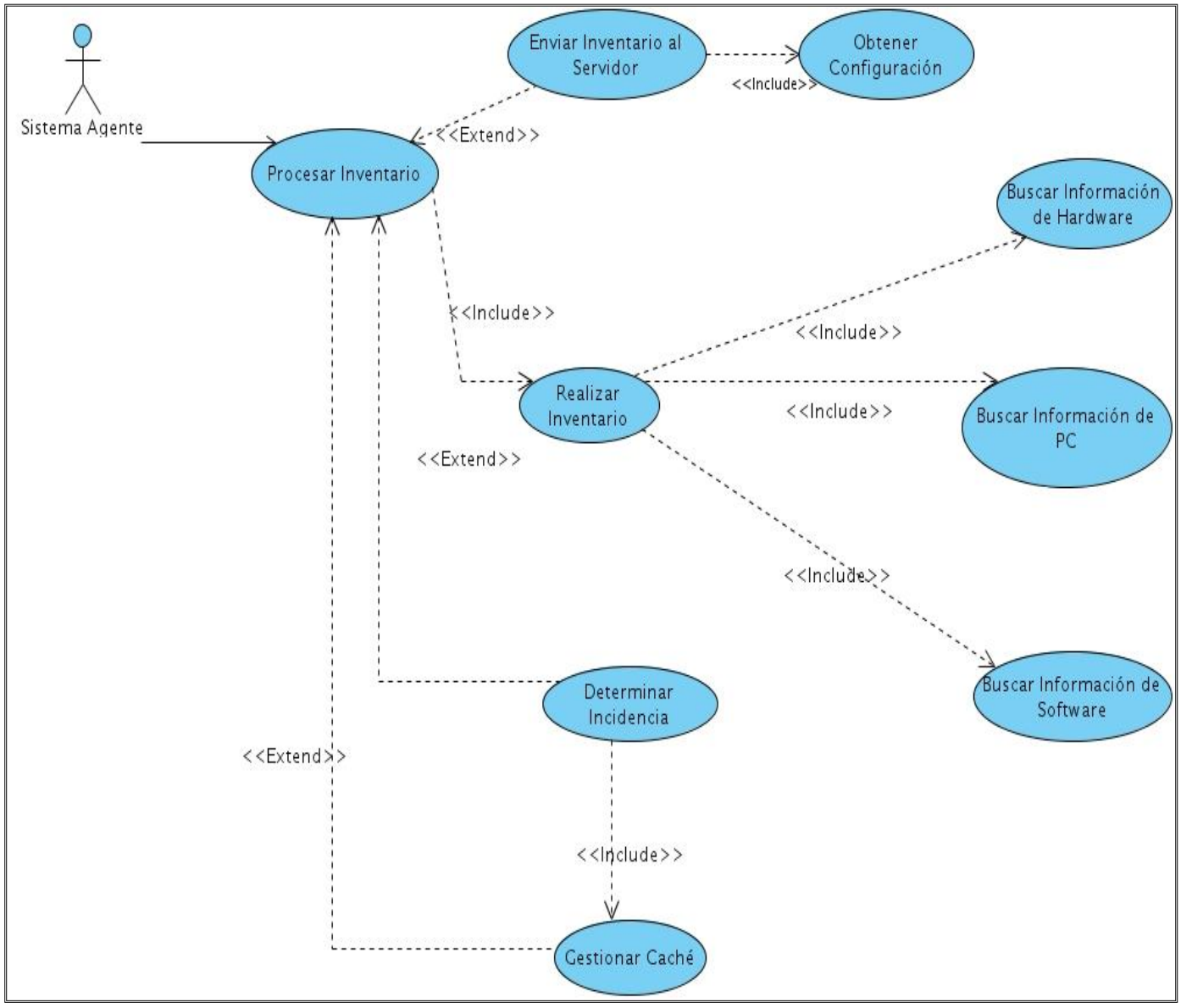

Fig 7. Diagrama de Caso de Uso del Sistema Agente

2.7 Conclusiones

Durante el desarrollo del capítulo se demostró el alcance de un sistema de inventario de hardware y software mediante el modelamiento del negocio, siguiendo los métodos y flujos de actividades que define un proceso de desarrollo; la identificación de los procesos del negocio, requisitos funcionales, no funcionales, actores del sistema, casos de uso del sistema así como el desglose detallado de los mismos, siendo cruciales para el análisis y diseño del sistema. Se obtuvieron conocimientos del negocio que avalan lo razonable de invertir en la creación del mismo.

CAPÍTULO 3 “Diseño del Sistema”

3.1 Introducción

Determinados los requerimientos funcionales y casos de uso del sistema, entrada fundamental para el análisis y diseño, se procede a analizar si es posible dar una solución que satisfaga los requisitos significativos de la arquitectura. El análisis y diseño contribuyen a una arquitectura sólida y estable que soporte las funcionalidades del sistema correlacionadas a requerimientos funcionales. Crean un punto de partida para las actividades de implementación

3.2 Modelo de Diseño

Un modelo de diseño es un modelo físico o plano específico para una implementación, cuyo propósito es comprender los aspectos relacionados a los requisitos no funcionales, restricciones de lenguaje de programación, tecnología de distribución, sistema operativo entre otros. Identifica estilos arquitectónicos, patrones arquitectónicos que expresan esquemas estructurales fundamentales para sistemas de software y patrones de diseño que dan solución a problemas comunes encontrados en la programación.

Según el nivel de abstracción, escalabilidad, flexibilidad que se quiere obtener en el sistema, se guía el desarrollo por un estilo arquitectónico de software. Teniendo en cuenta que el sistema de inventario de hardware y software es un sistema complejo, inestable por la evolución del hardware y el software, un estilo arquitectónico de llamada y retorno, como arquitectura en capas, garantiza flexibilidad, escalabilidad, aplicaciones robustas debido al encapsulamiento, desarrollo paralelo, mantenimiento y soporte más sencillo.

La arquitectura de software podría decirse que ha evolucionado de la siguiente manera (12):

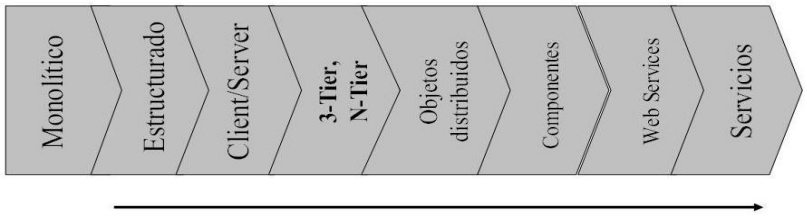

Fig 8. Evolución de la Arquitectura.

3.2.1 Estructura general del Sistema

El siguiente diagrama representa la distribución de los componentes en capas del Sistema de Inventario de Hardware y software.

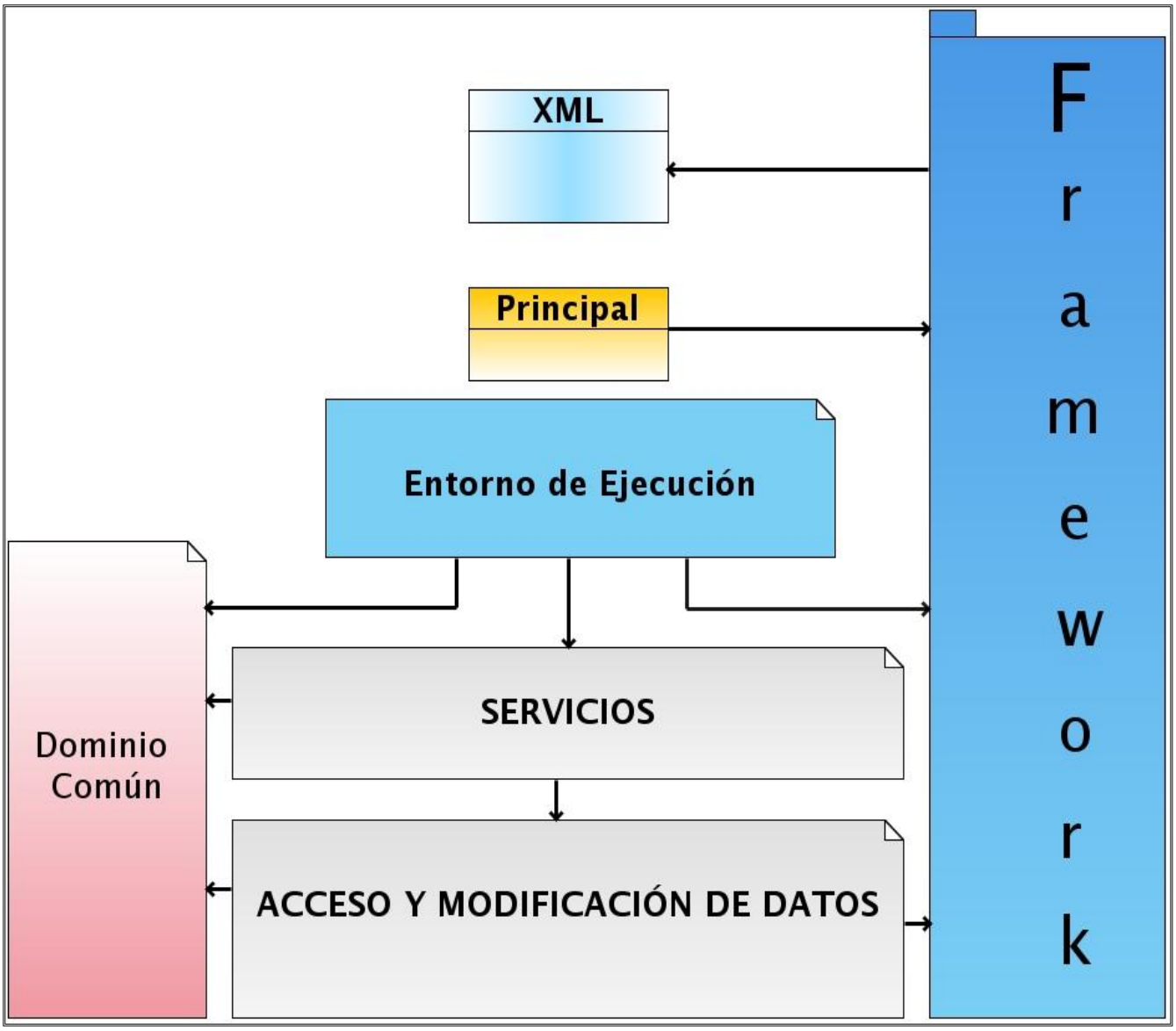

Fig 9. Estructura general del sistema

3.2.2 Estructura detallada del Sistema Agente

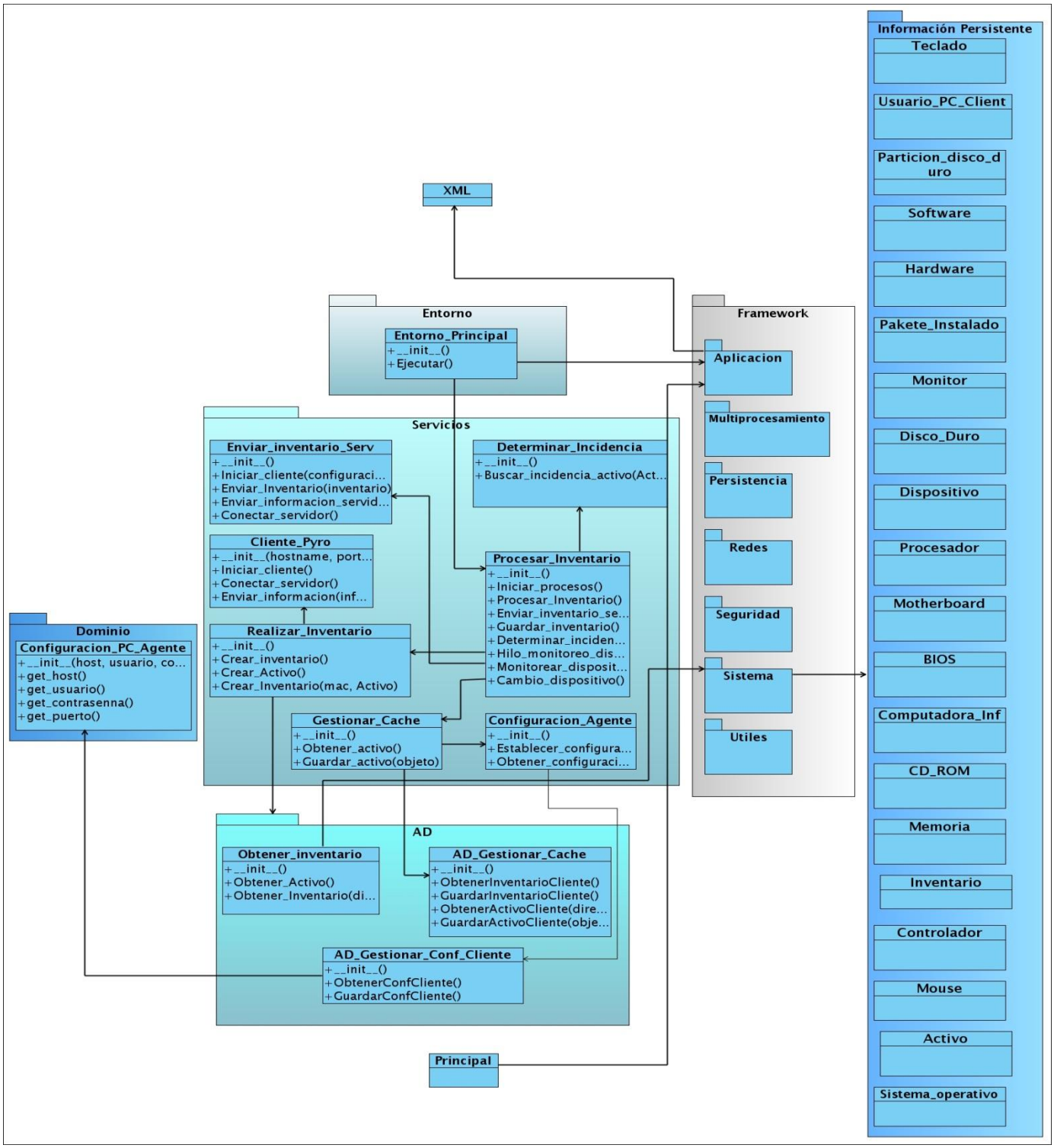

Fig 10. Estructura detallada del Sistema Agente.

3.2.3 Estructura detallada del Sistema Servidor

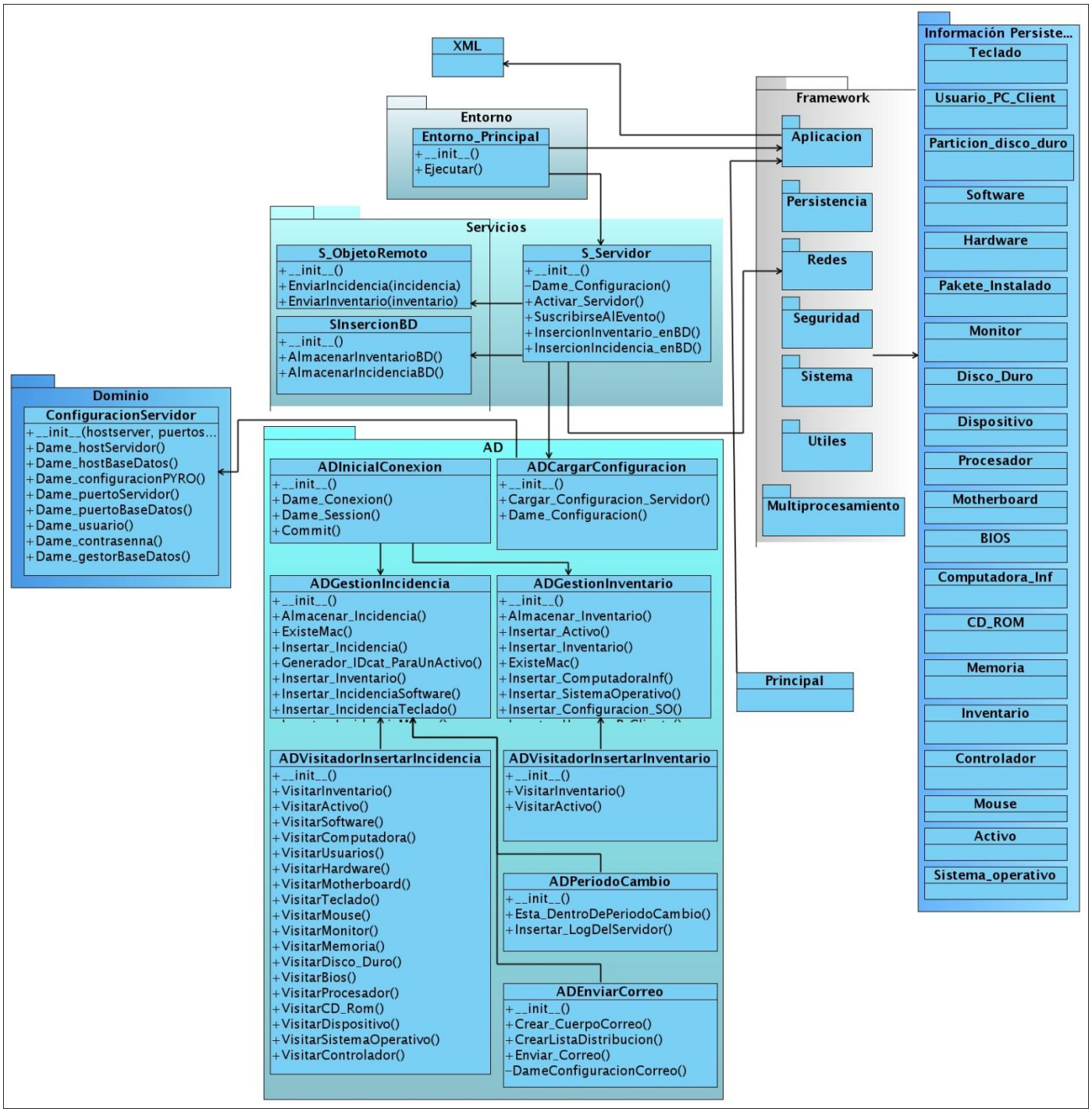

Fig 11. Estructura detallada del Sistema Servidor.

3.2.4 Diagramas de Clases del Diseño Sistema Agente

3.2.4.1 Diagrama de Clase del Diseño CU Buscar Información de Hardware

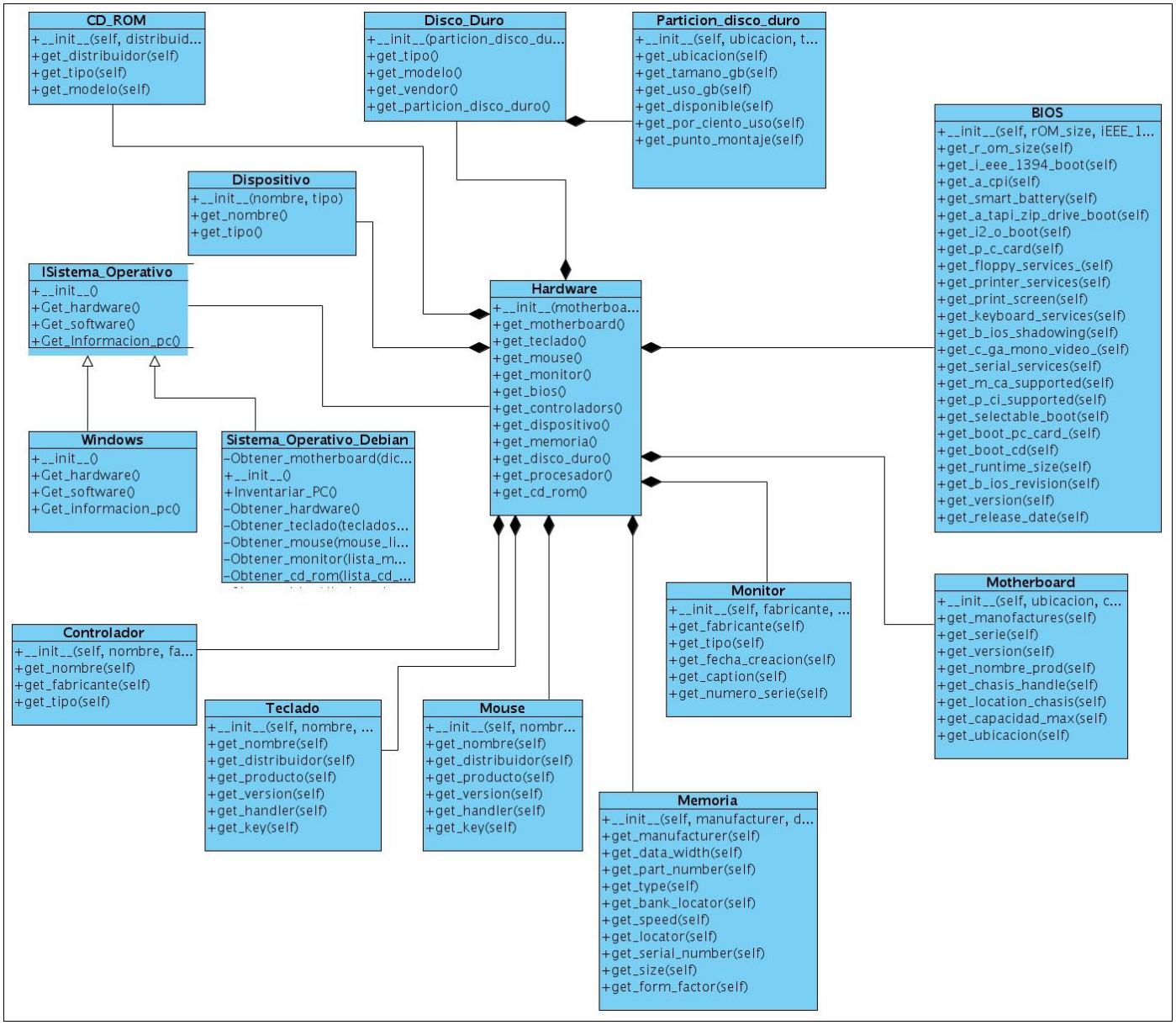

Fig 12. Diagrama de Clase del Diseño: CUBuscar Información de Hardware.

3.2.4.2 Diagrama de Clase del Diseño CU Buscar Información de Software

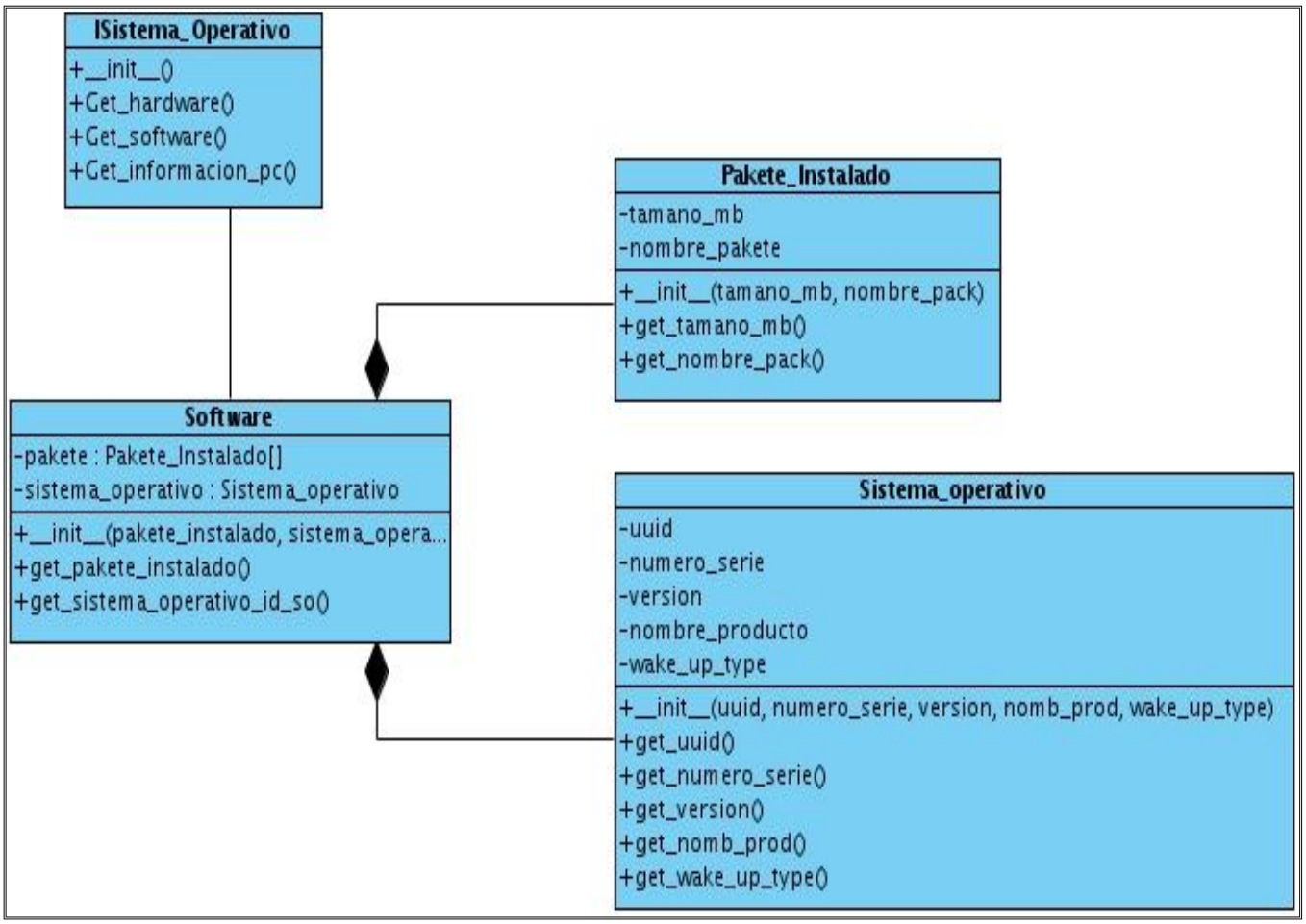

Fig 13. Diagrama de Clase del Diseño: CU Buscar Información de Software

3.2.4.3 Diagrama de Clase del Diseño CU Buscar Información de PC

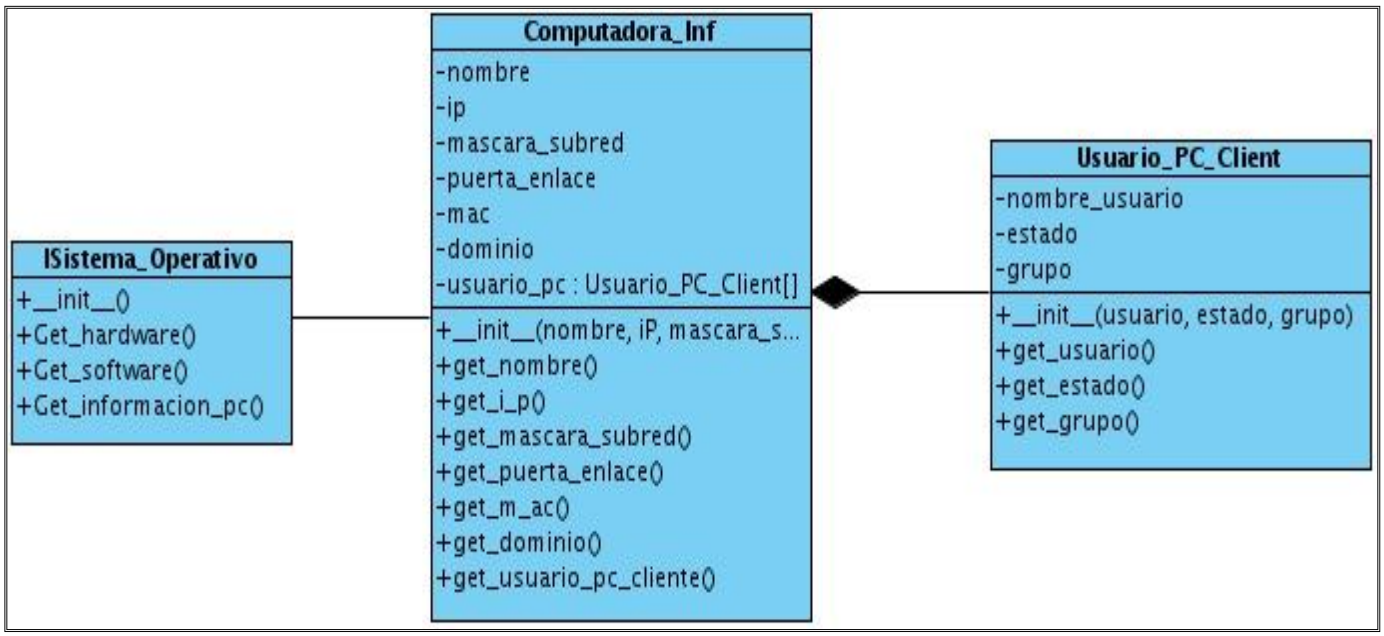

Fig 14. Diagrama de Clase del Diseño: CU Buscar Información de PC

3.2.4.4 Diagrama de Clase del Diseño CU Realizar Inventario

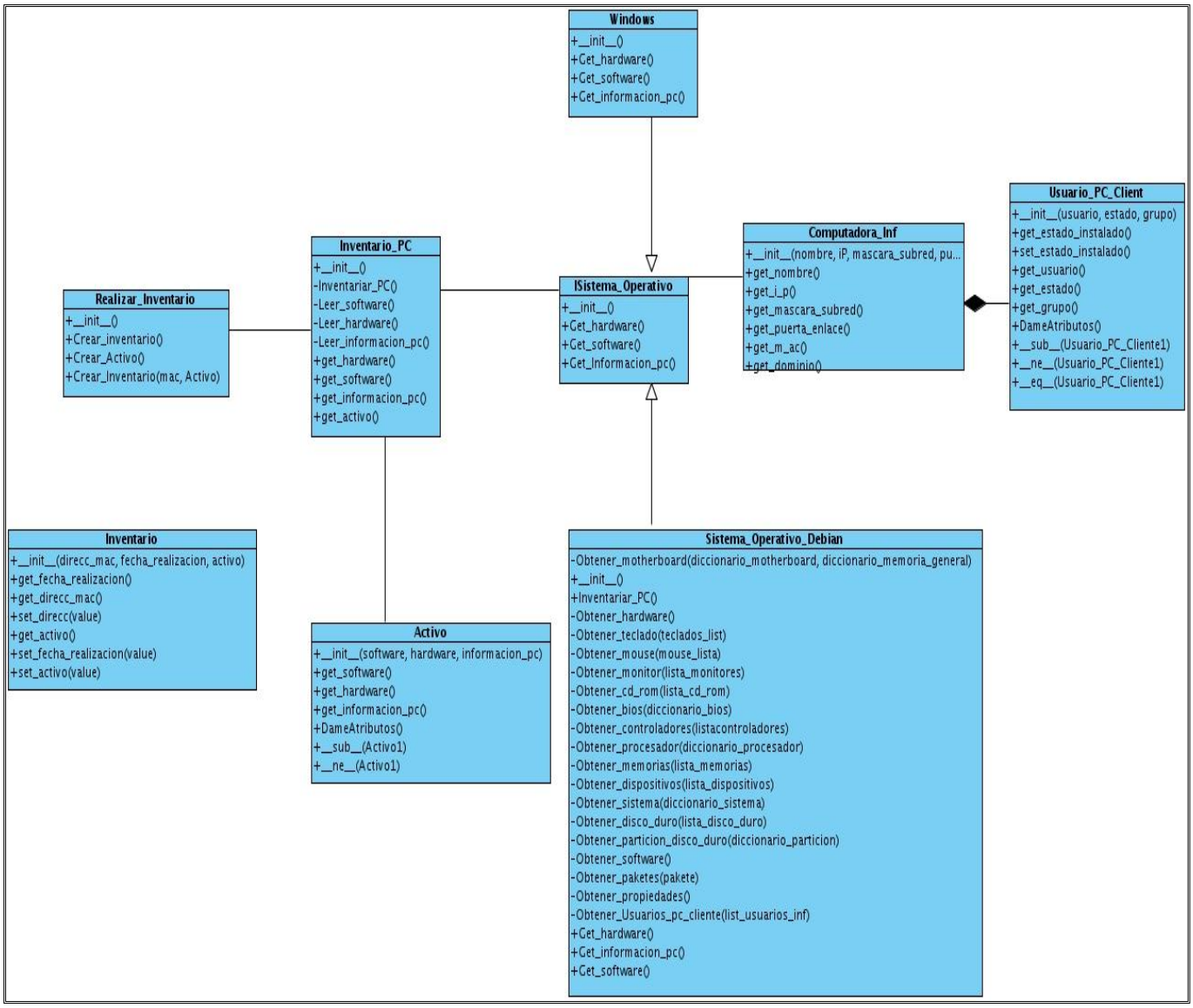

Fig 15. Diagrama de Clase del Diseño: CU Realizar Inventario

3.2.4.5 Diagrama de Clase del Diseño CU Determinar Incidencia

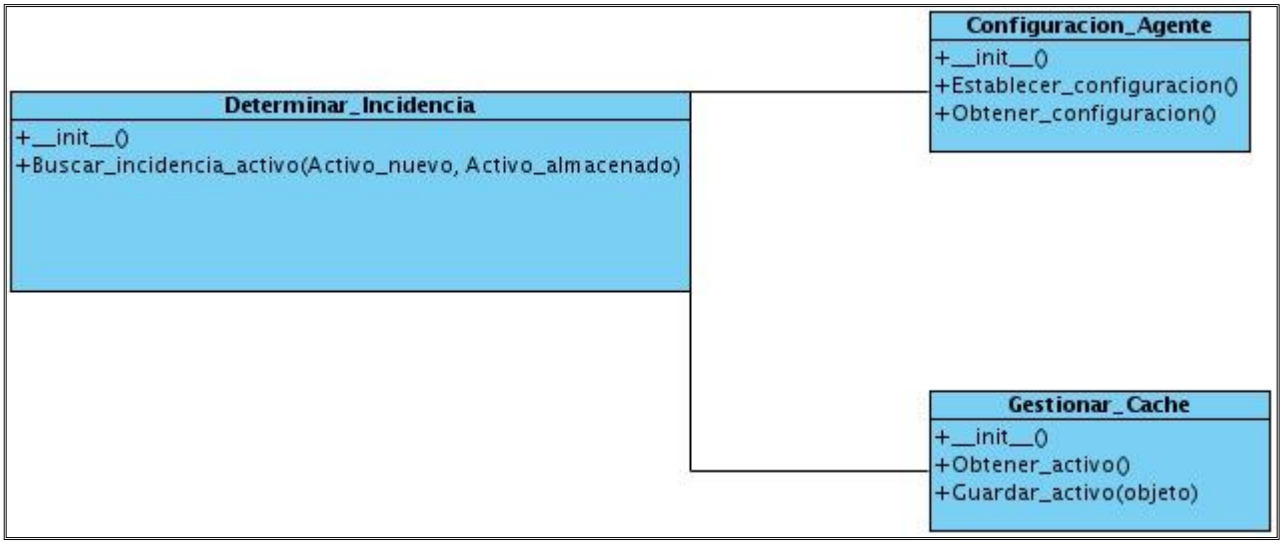

Fig 16. Diagrama de Clase del Diseño: CU Realizar Inventario

3.2.4.5 Diagrama de Clase del Diseño CU Gestionar Caché

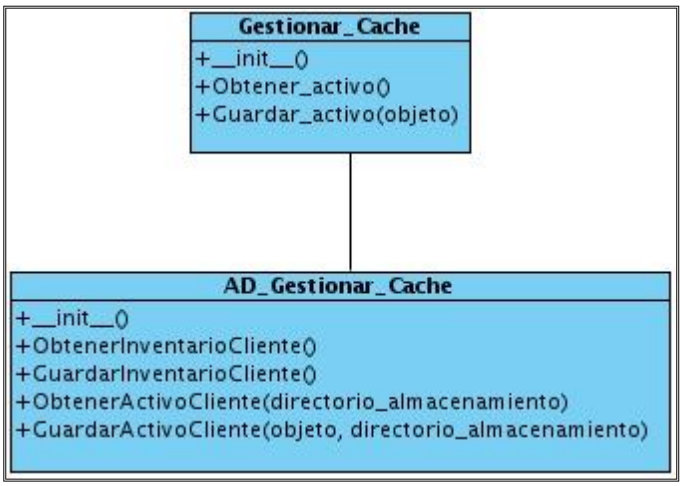

Fig 17. Diagrama de Clase del Diseño: CU Gestionar Caché

3.2.4.5 Diagrama de Clase del Diseño CU Obtener Configuración

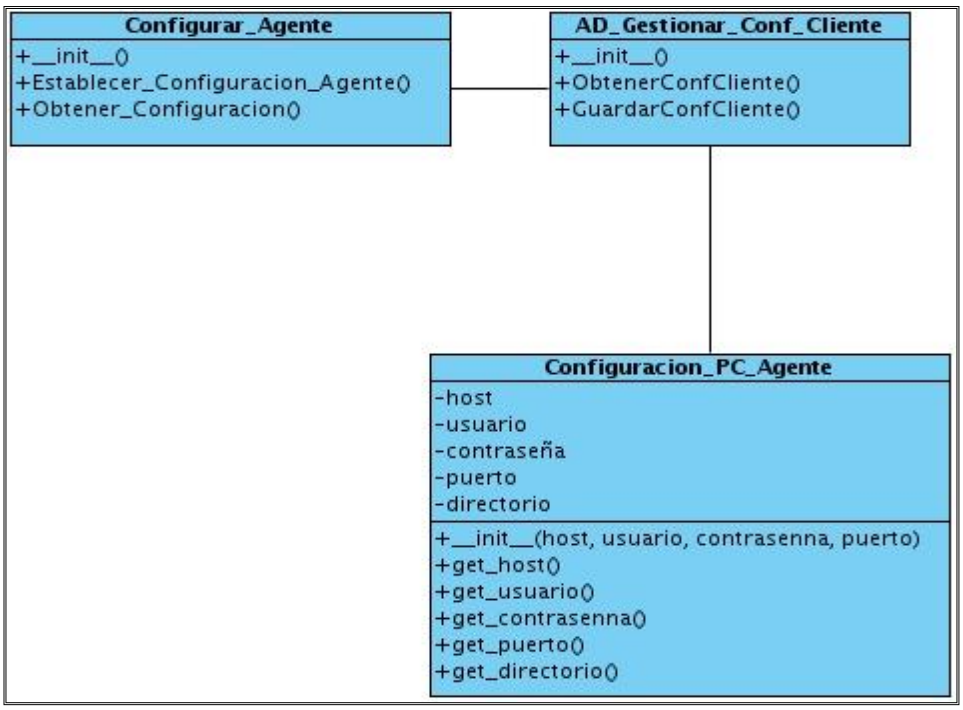

Fig 18. Diagrama de Clase del Diseño: CU Obtener Configuración

3.2.4.6 Diagrama de Clase del Diseño CU Enviar Inventario al Servidor

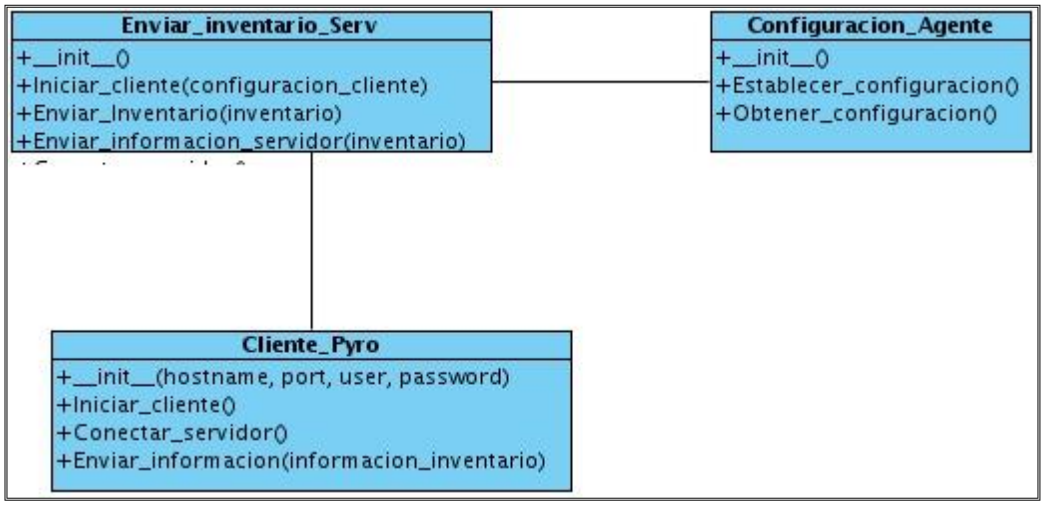

Fig 19. Diagrama de Clase del Diseño: CU Enviar Inventario al Servidor

3.2.5 Diagramas de Clases del Diseño del Sistema Servidor

3.2.5.1 Diagrama de Clase del Diseño CU Recibir Información de Agente

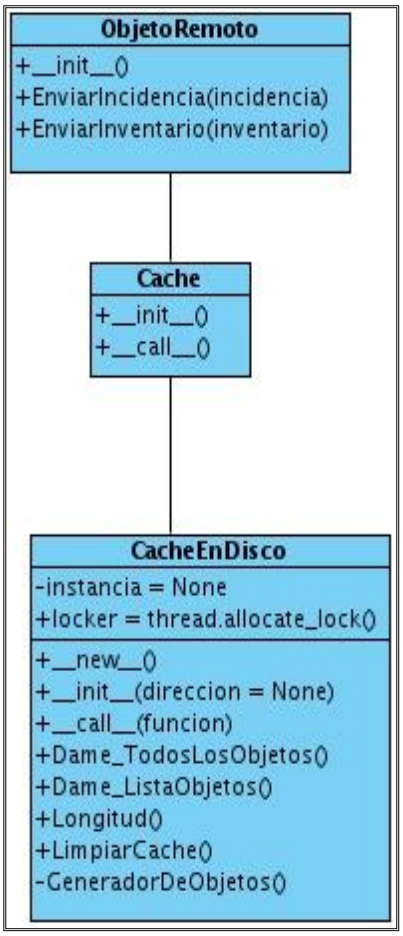

Fig 20. Diagrama de Clase del Diseño: CU Recibir Información de Agente

3.2.5.2 Diagrama de Clase del Diseño CU Guardar Inventario

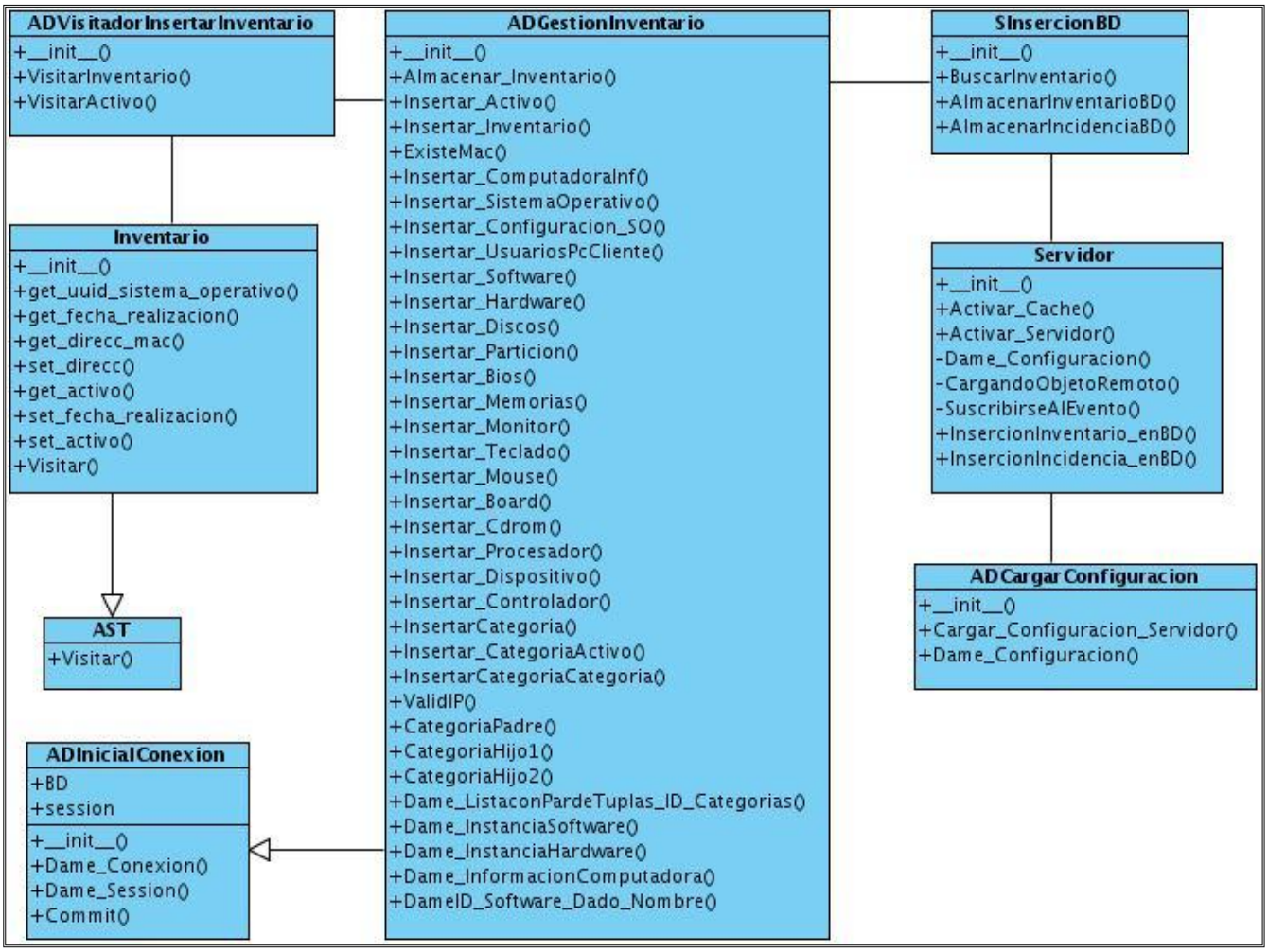

Fig 21. Diagrama de Clase del Diseño: CU Guardar Inventario

3.2.5.3 Diagrama de Clase del Diseño CU Guardar Incidencia

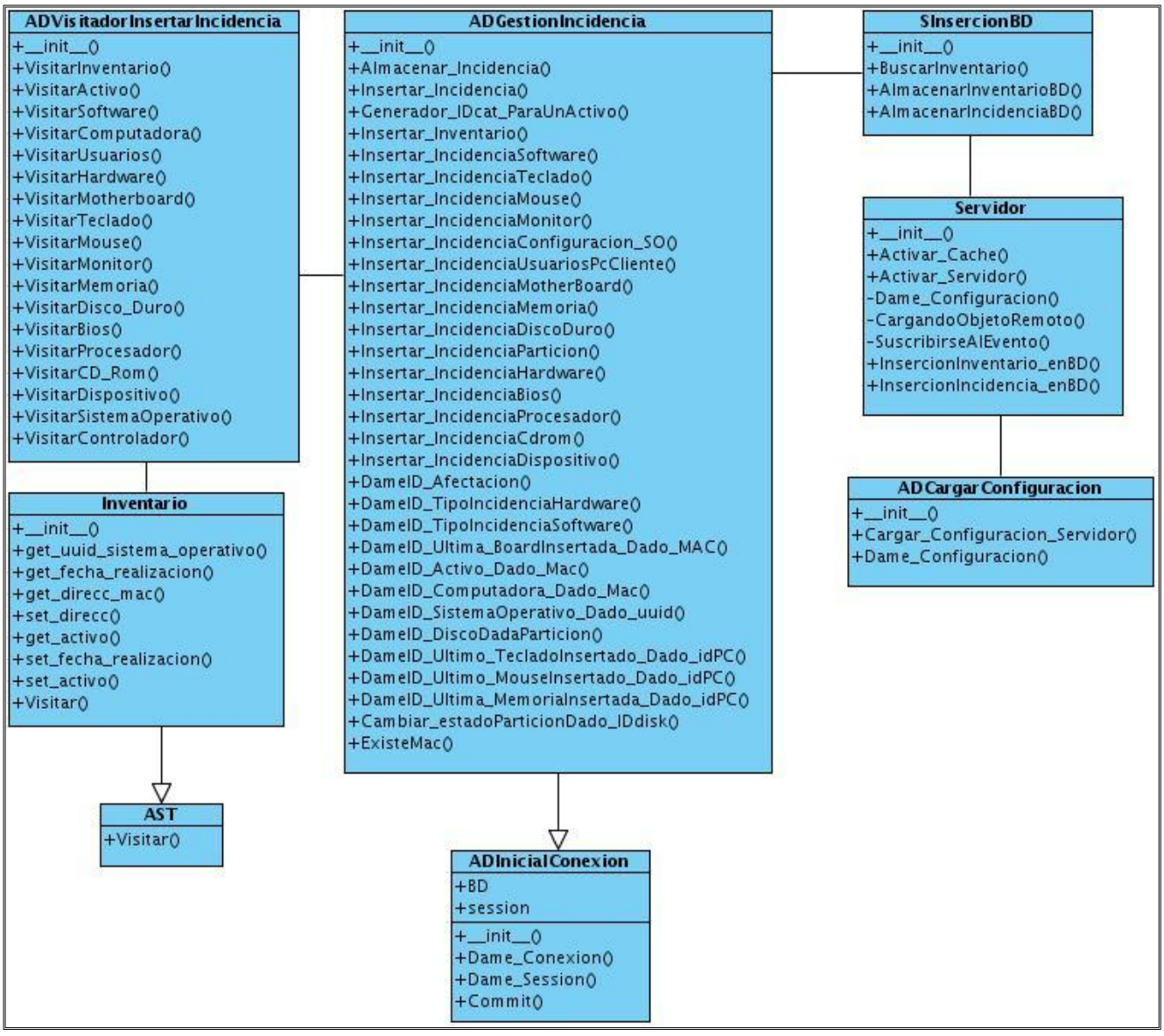

Fig 22. Diagrama de Clase del Diseño: CU Guardar Incidencia

3.2.5.4 Diagrama de Clase del Diseño CU Ejecutar Acción

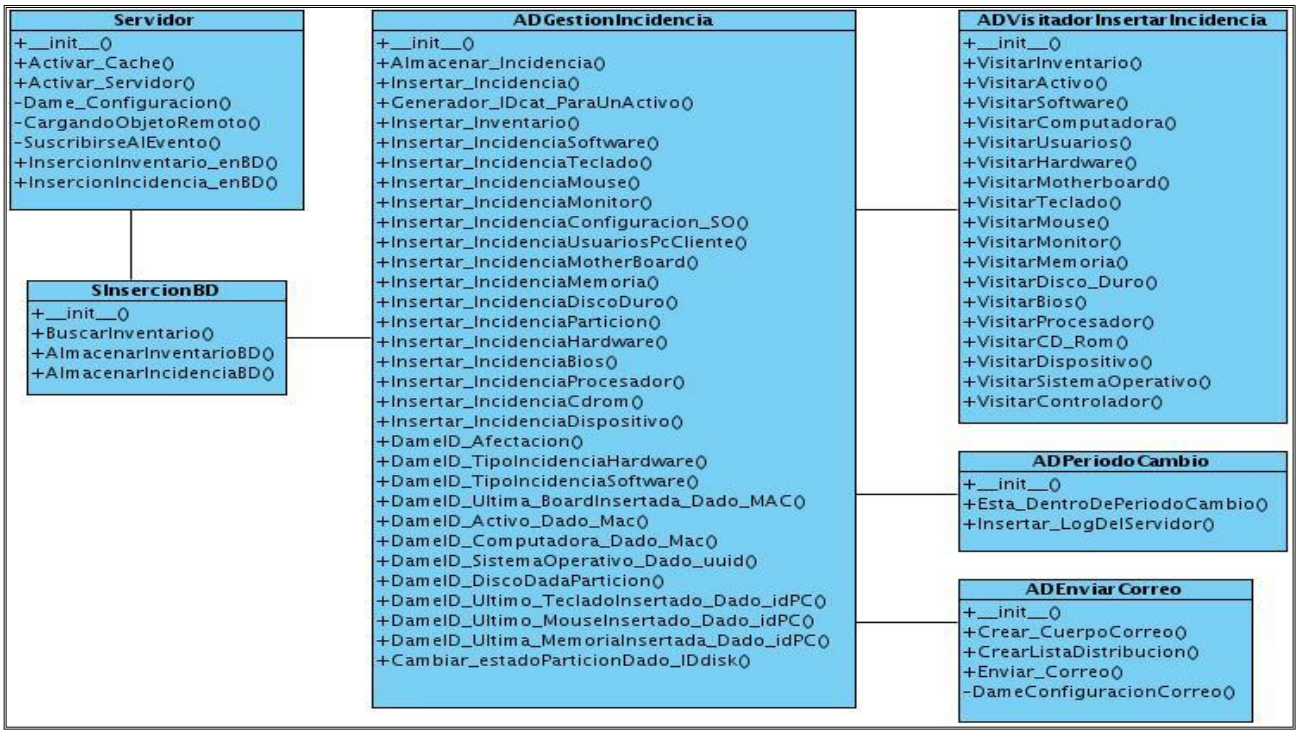

Fig 23. Diagrama de Clase del Diseño: CU Ejecutar Acción

3.2.5.5 Diagrama de Clase del Diseño CU Buscar Inventario

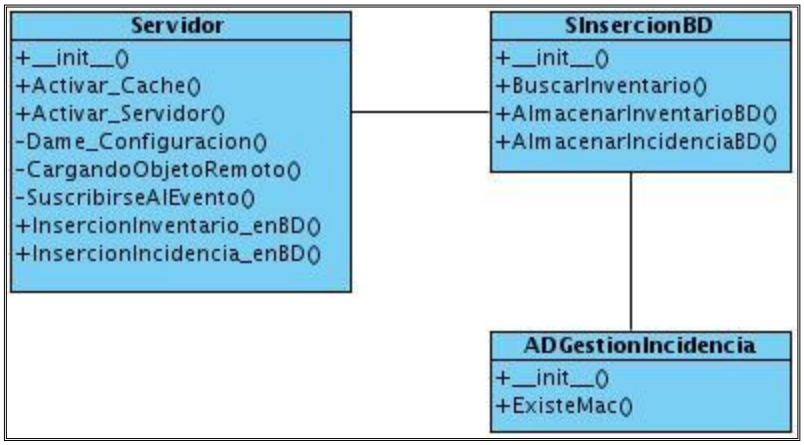

Fig 24. Diagrama de Clase del Diseño: CU Buscar Inventario

3.3 Conclusiones

El diseño de software materializó los requerimientos del cliente. Los procesos que durante esta actividad fueron realizados permitieron describir todos los aspectos del futuro sistema, enfocado en los dominios de datos, funcional y comportamientos desde el punto de vista de la implementación.

Durante el desarrollo de las actividades correspondientes al diseño de software se establecieron criterios técnicos para un buen diseño. El proceso de Diseño del Software exige buena calidad a través de la aplicación de principios fundamentales de Diseño, Metodología sistemática y una revisión exhaustiva.

CAPÍTULO 4 “Implementación del Sistema”

4.1 Introducción

Es en la implementación del sistema donde se obtienen los componentes y subsistemas en un modelo de implementación, todo esto a partir del modelo de clases del diseño definido en el capítulo anterior. En este capítulo se obtiene un sistema mucho más maduro en términos de componentes, llevando a cabo la implementación de las clases y sus relaciones en uno o varios lenguajes de programación.

4.2 Diagrama de componentes

Los diagramas de componentes se emplean para el modelamiento de las vistas estáticas de un sistema en términos de componentes, como código fuente, binarios, librerías o ejecutables, así como de las dependencias lógicas que existen entre ellos.

4.2.1 Diagrama de Componentes General del Sistema Agente

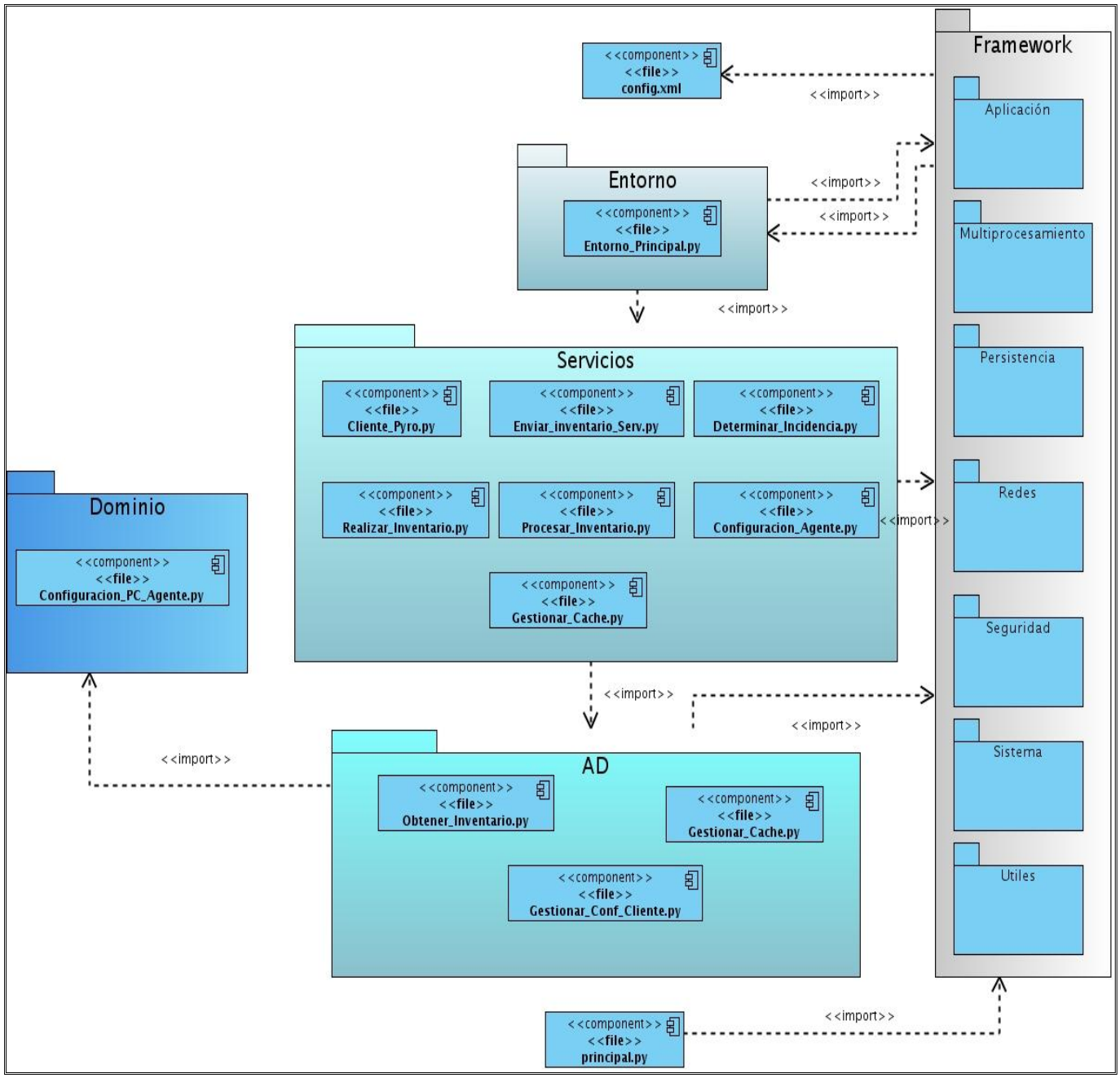

Fig 25. Diagrama de Componentes General del Sistema Agente

4.2.1.2 Diagrama de Componentes CU Buscar Información del Hardware

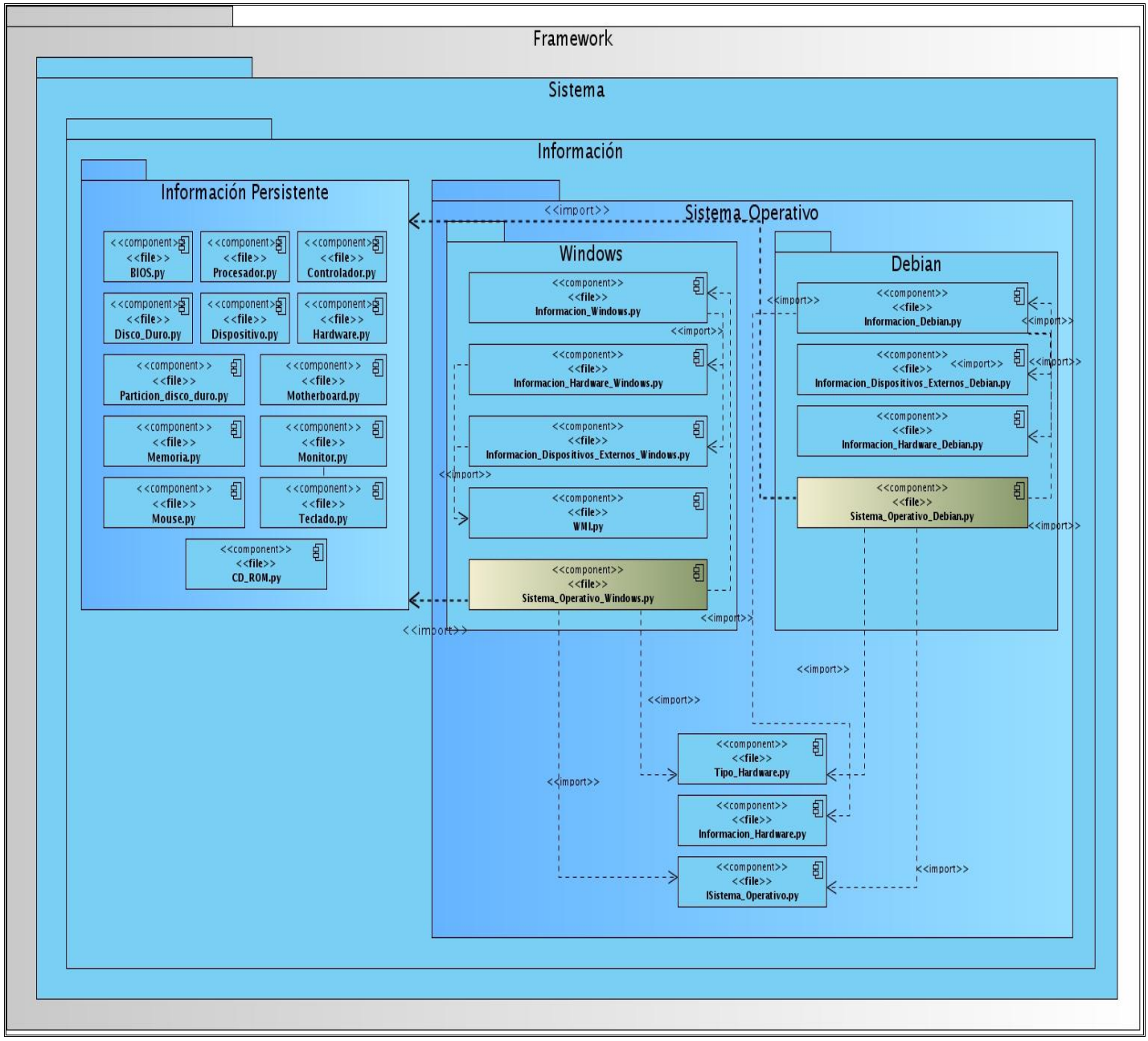

Fig 26. Diagrama de Componentes: CU Buscar Información del Hardware

4.2.1.3 Diagrama de Componentes CU Buscar Información de PC

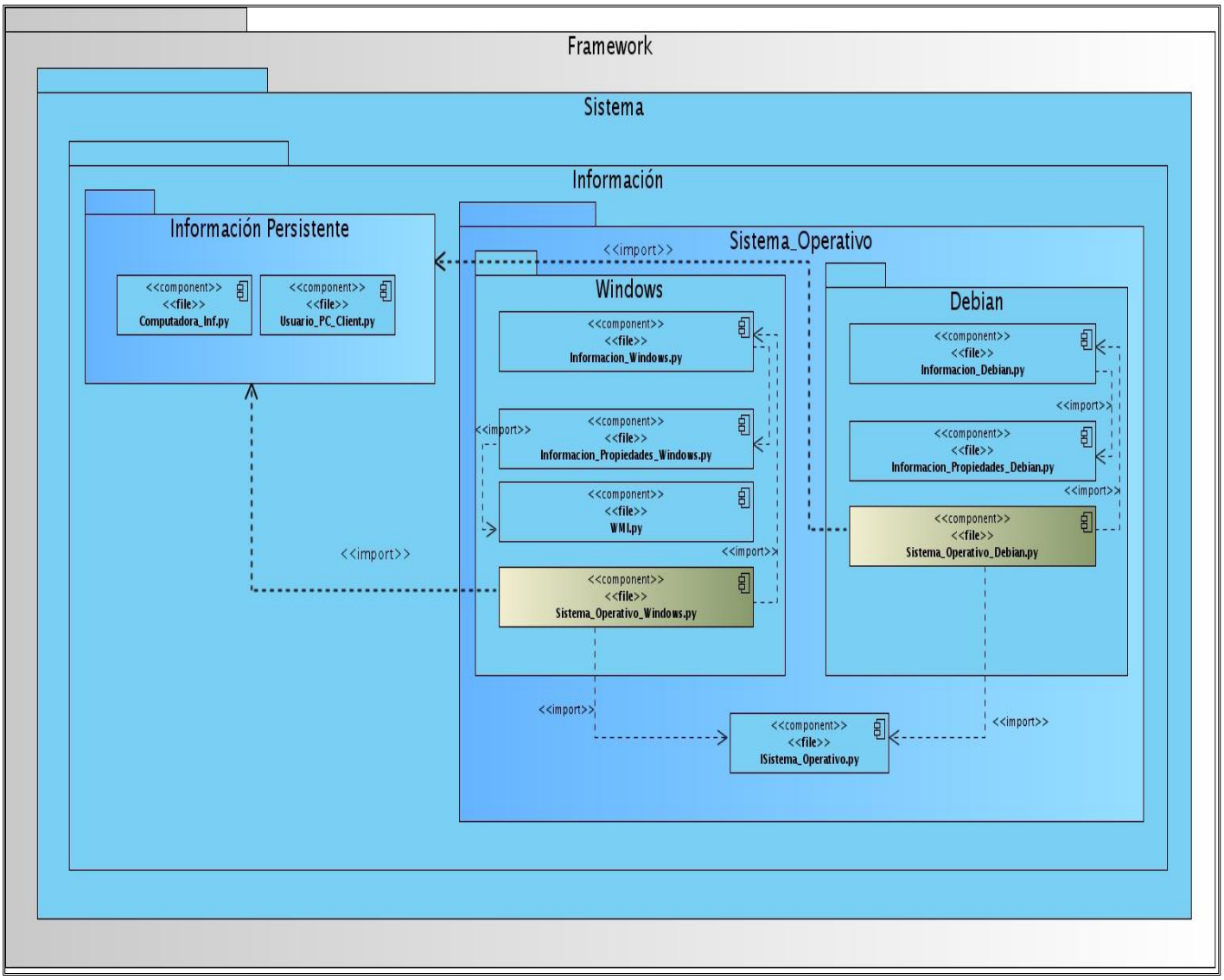

Fig 27. Diagrama de Componentes: CU Buscar Información de PC

4.2.1.4 Diagrama de Componentes CU Buscar Información de Software

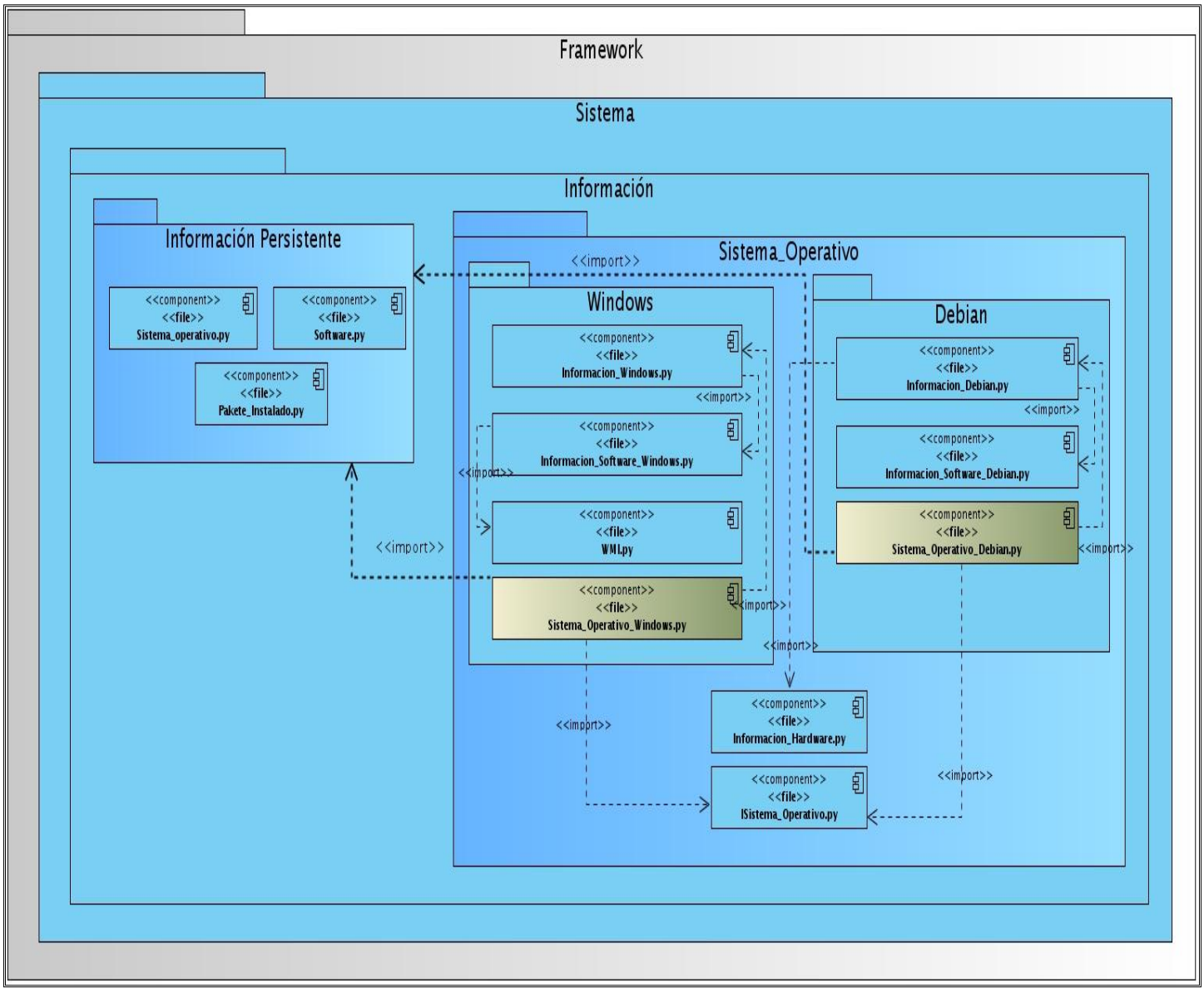

Fig 28. Diagrama de Componentes: CU Buscar Información de Software.

4.2.1.5 Diagrama de Componente CU Realizar Inventario

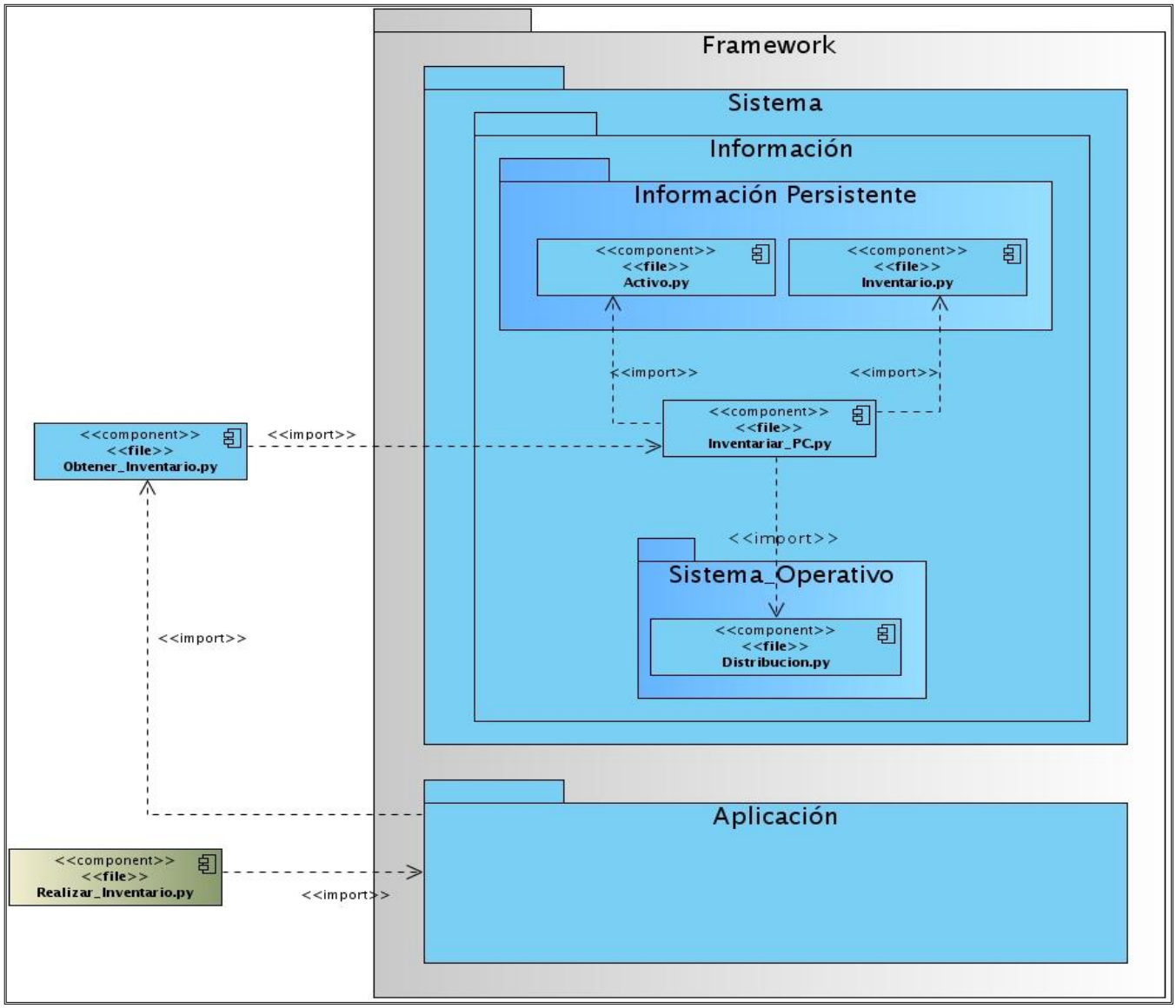

Fig 29. Diagrama de Componentes: CU Realizar Inventario

4.2.1.6 Diagrama de Componente CU Guardar Configuración

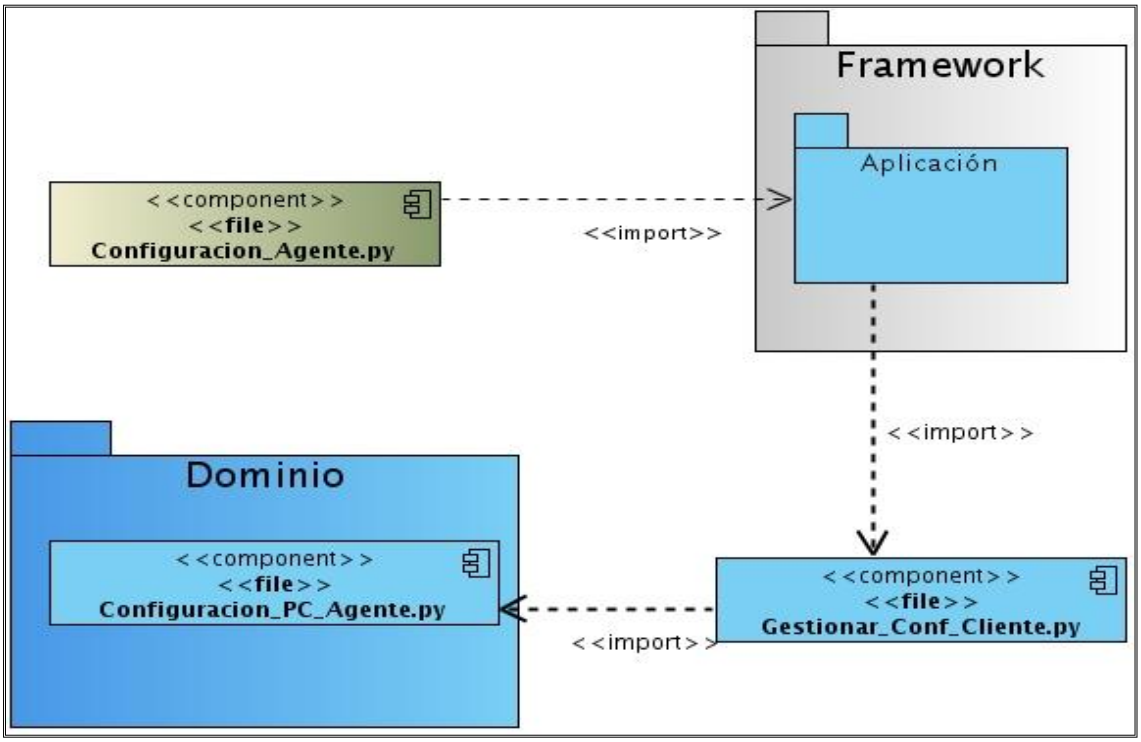

Fig 30. Diagrama de Componentes: CU Guardar Configuración

4.2.1.7 Diagrama de Componentes CU Gestionar Caché

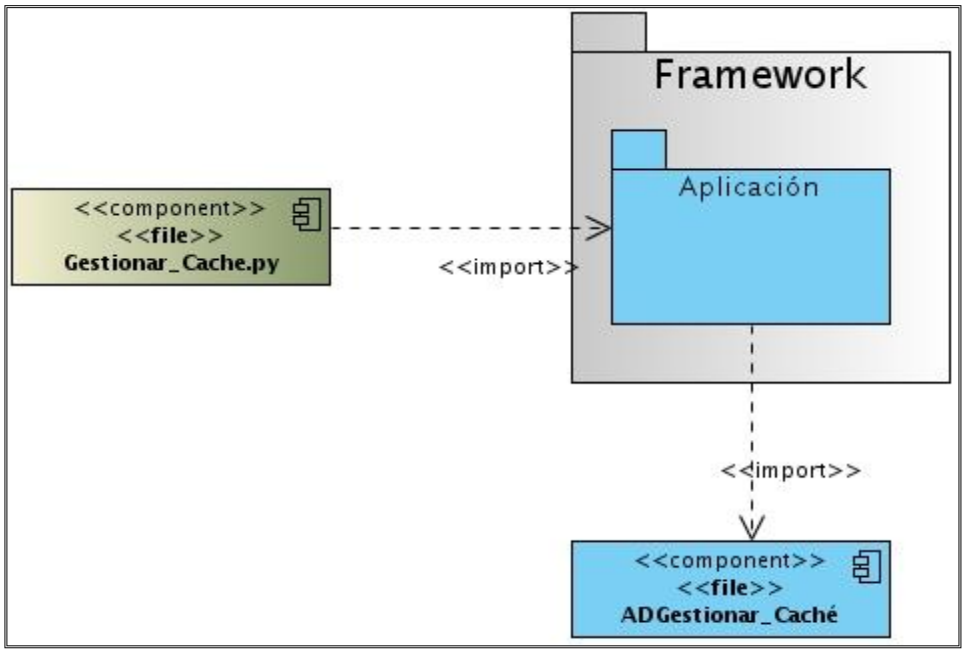

Fig 31. Diagrama de Componentes: CU Gestionar Caché.

4.2.1.8 Diagrama de Componentes Enviar Información al Servidor

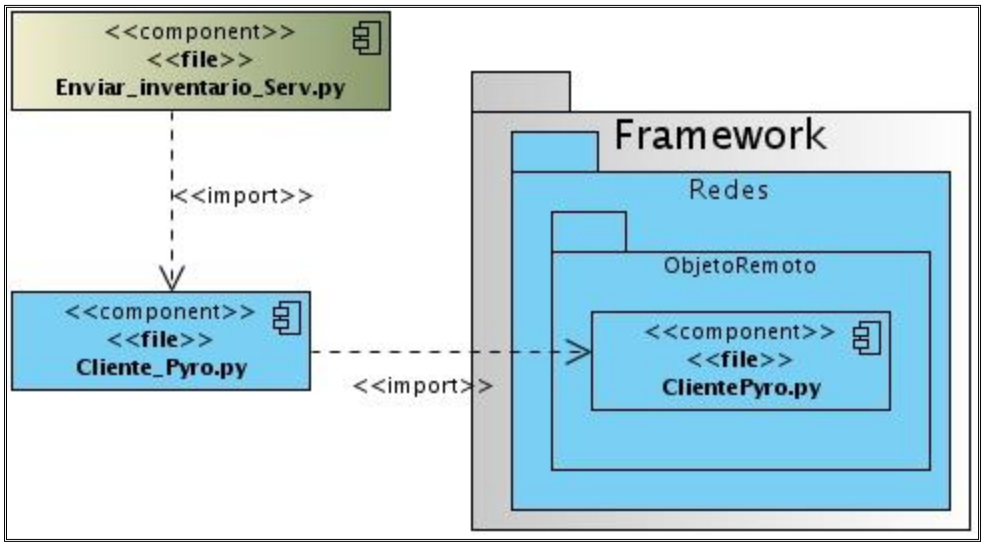

Fig 32. Diagrama de Componentes: CU Enviar Información al Servidor

4.2.1.9 Diagrama de Componentes CU Procesar Inventario

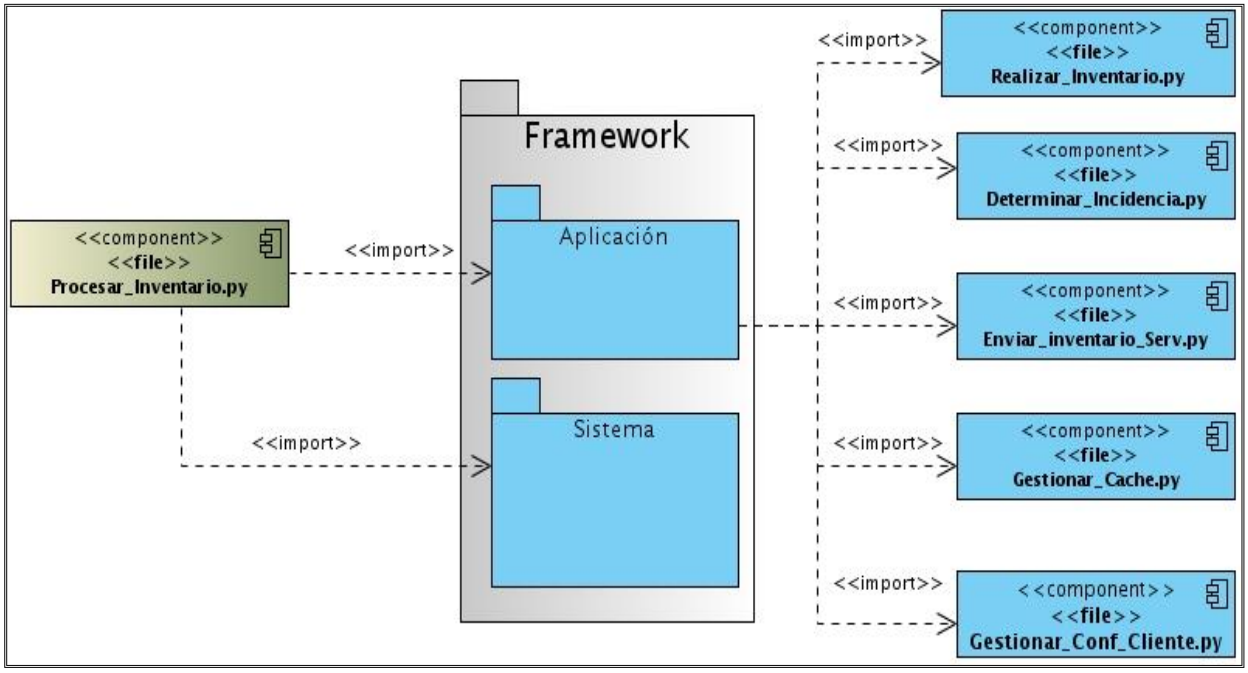

Fig 33. Diagrama de Componentes: CU Procesar Inventario

4.2.1.9.1 Diagrama de Componentes CU Determinar Incidencias

Fig 34. Diagrama de Componentes: CU Determinar Incidencias

4.2.2 Diagrama de Componentes General del Sistema Servidor

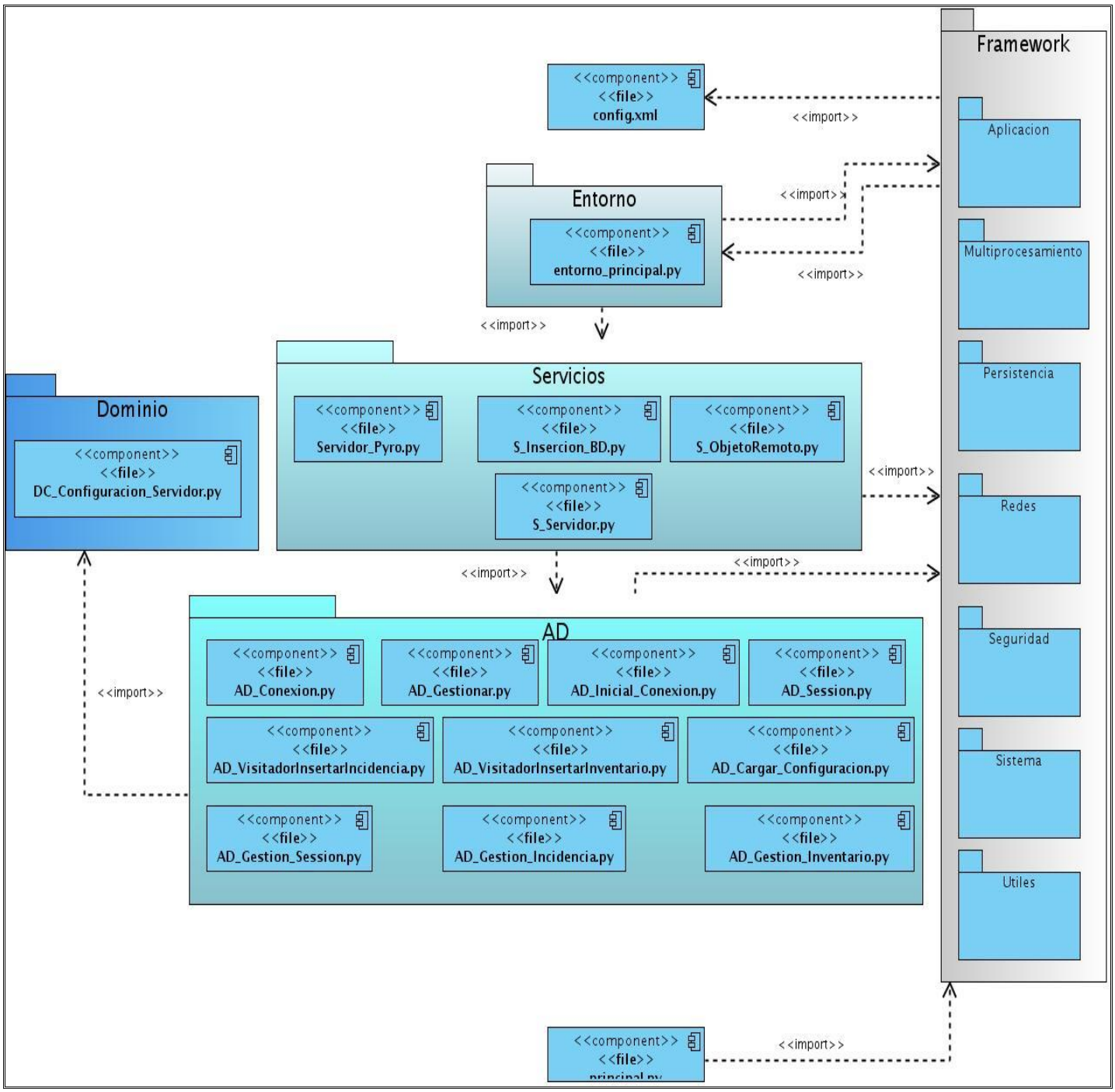

Fig 35. Diagrama de Componentes General del Sistema Servidor.

4.2.2.1 Diagrama de Componentes CU Recibir Información del Cliente

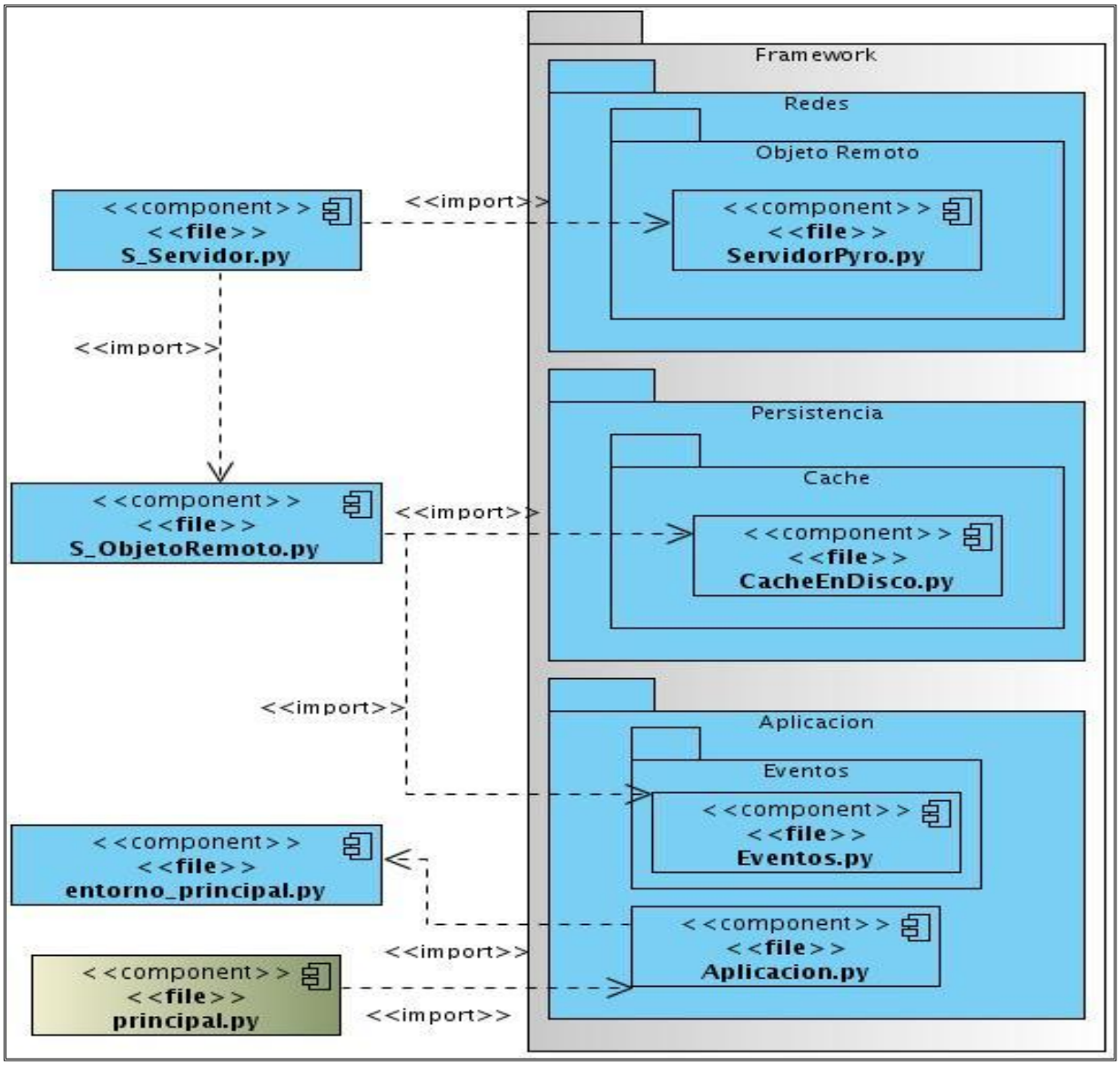

Fig 36.

Diagrama de Componentes: CU Recibir Información del Cliente

4.2.2.2 Diagrama de Componentes CU Guardar Inventario

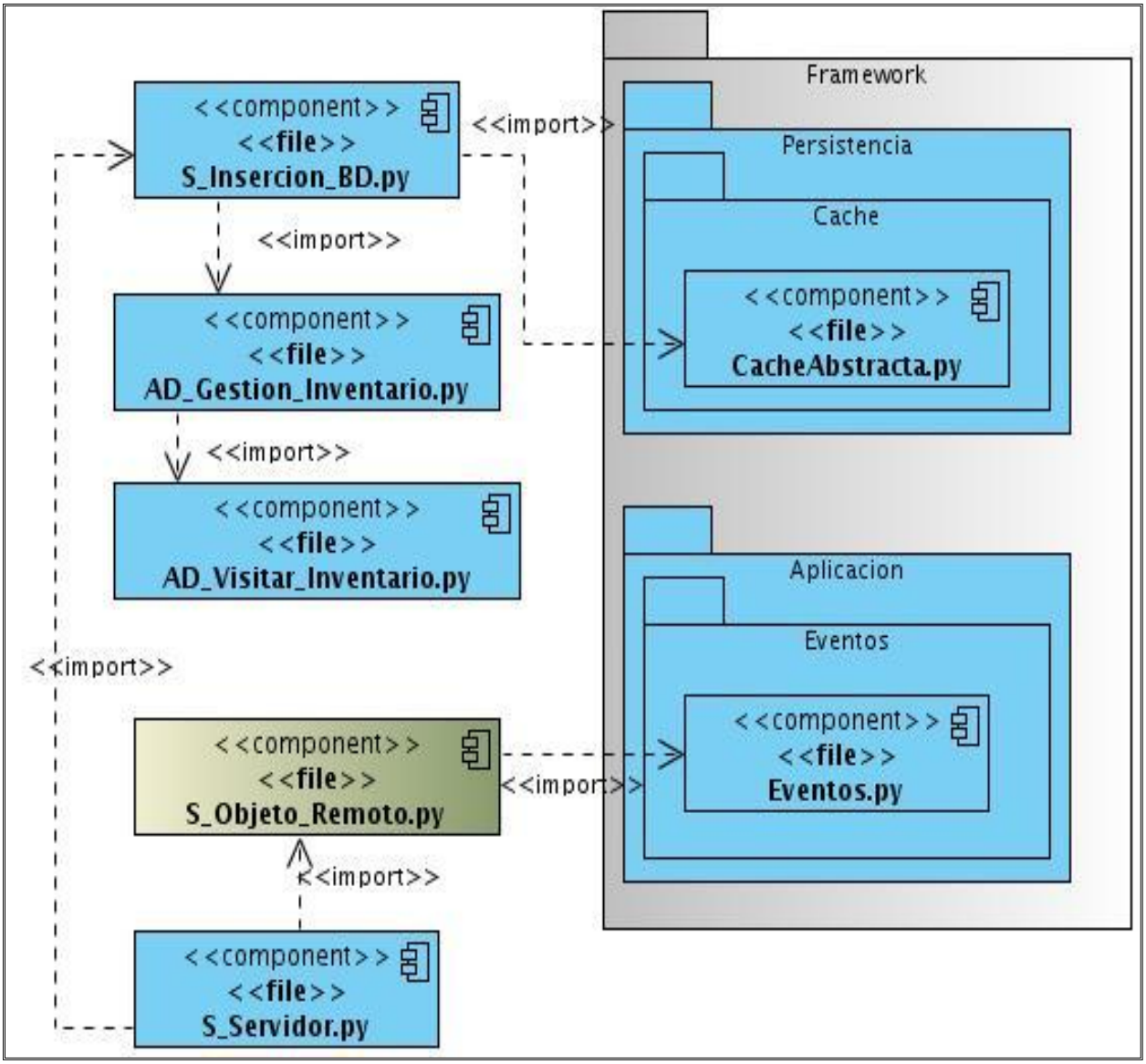

Fig 37. Diagrama de Componentes: CU Guardar Inventario

4.2.2.3 Diagrama de Componentes CU Guardar Incidencia

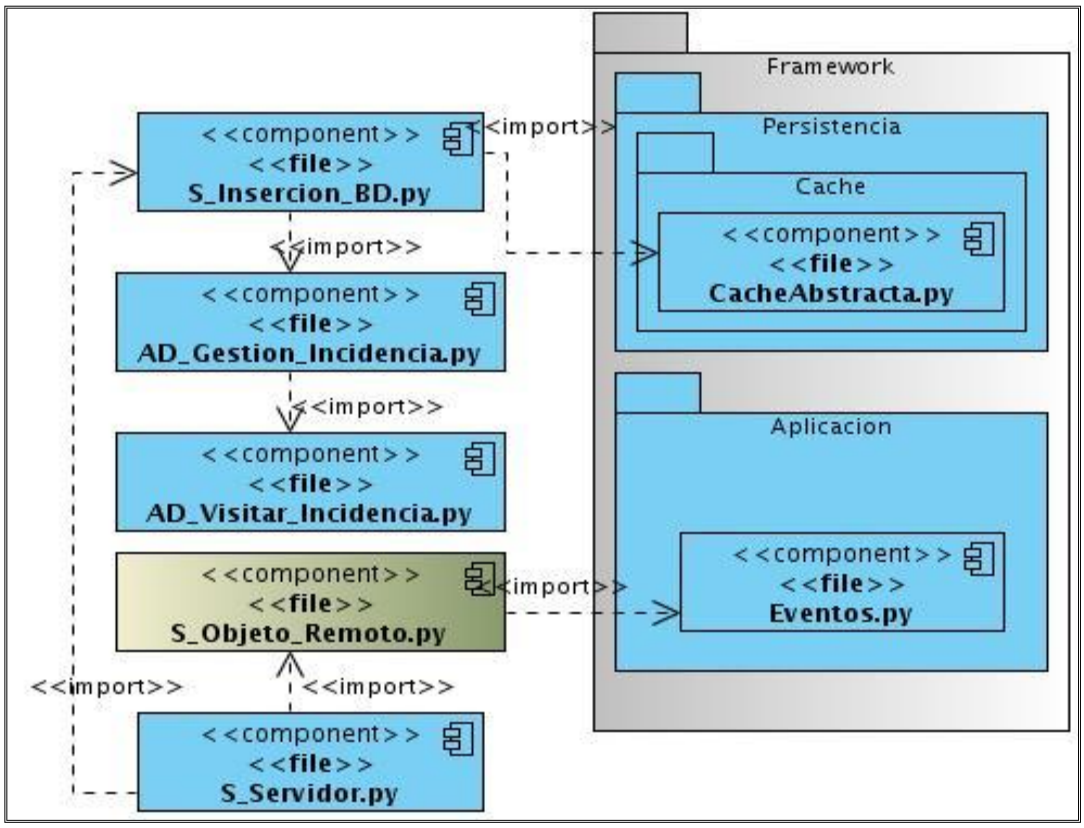

Fig 38. Diagrama de Componentes: CU Guardar Incidencia

4.2.2.3 Diagrama de Componentes CU Ejecutar Acción

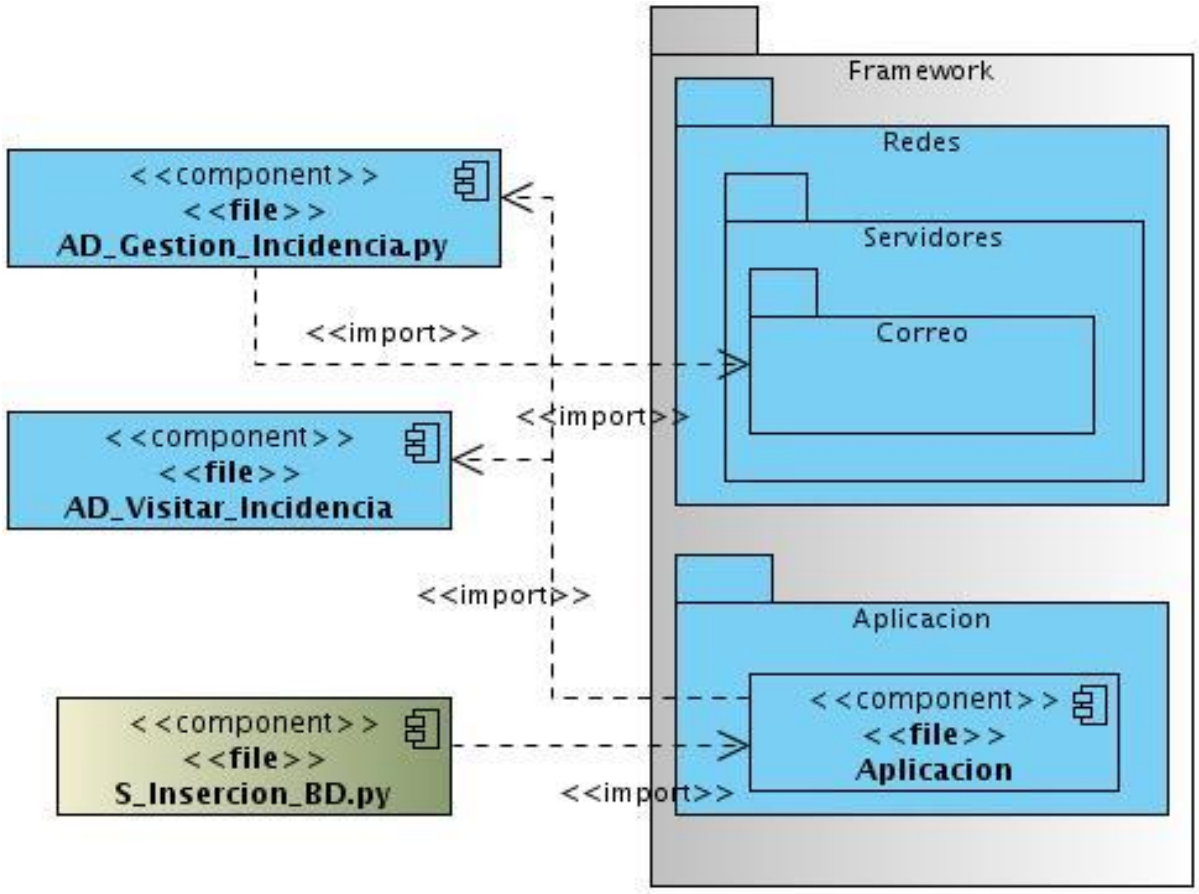

Fig 39. Diagrama de Componentes: CU Ejecutar Acción

4.3 Diagrama de despliegue

Los diagramas de despliegue se emplean para el modelamiento del hardware que es utilizado en la implementación del sistema y de las relaciones entre sus componentes, los procesadores se representan como nodos mostrando cómo se distribuye físicamente el sistema propuesto. Se emplean principalmente para modelar sistemas cliente-servidor, sistemas empotrados o los sistemas de componentes distribuidos.

- Servidor de IHS: Representa el ordenador donde reside el Sistema Servidor.
- PC Agente de IHS: Representa el ordenador donde reside el Sistema Cliente.
- Servidor de Base de Datos: Representa el ordenador donde reside la Base de Datos.

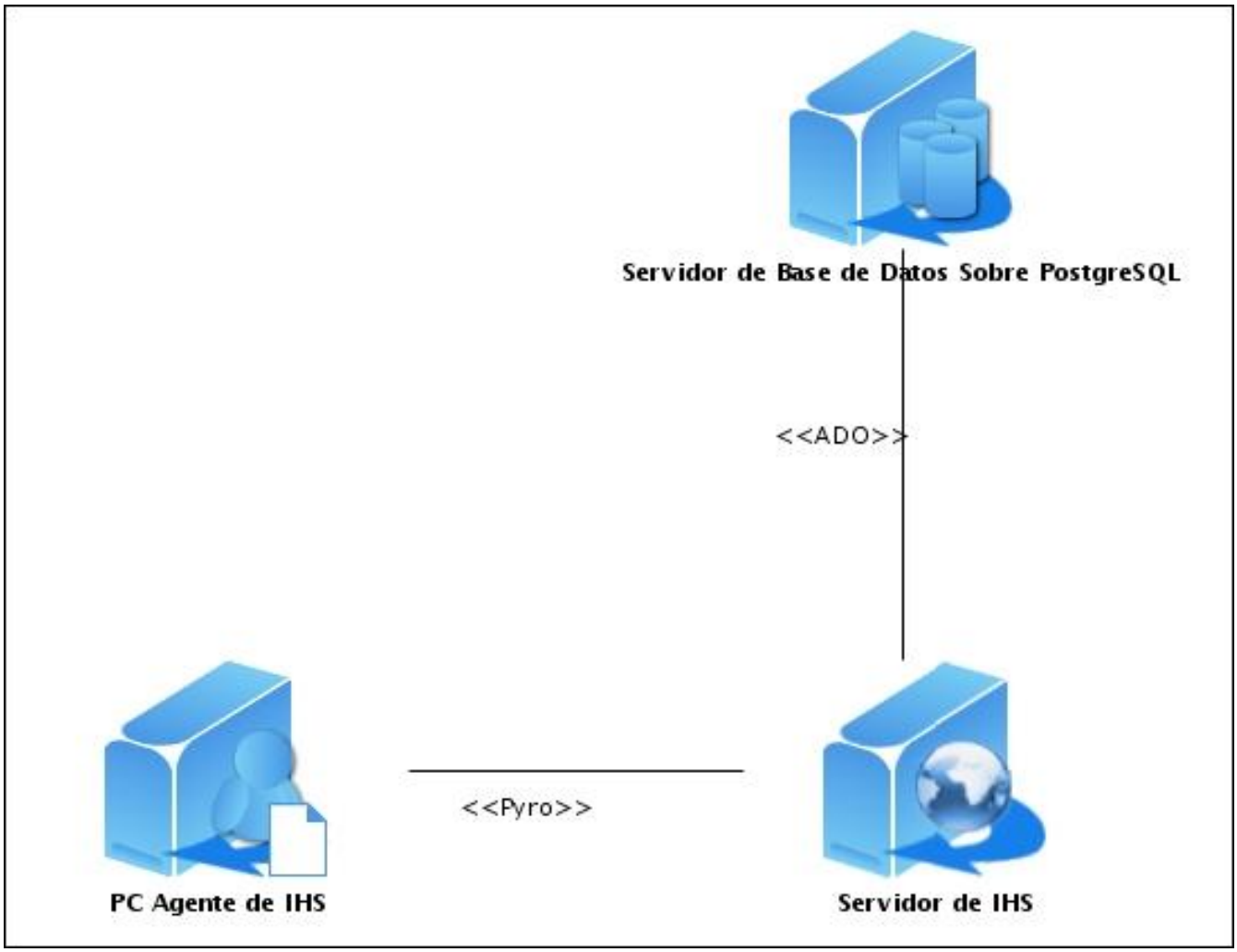

Fig 40. Diagrama de Despliegue del Sistema.

4.4 Conclusiones

En este capítulo se solucionaron los principales objetivos, como son la organización de los sistemas en términos de subsistemas de Implementación organizados en capas, la implementación de los elementos obtenidos del modelo de diseño en términos de componentes y de elementos de implementación como son los ficheros fuentes, ejecutables, binarios y otros.

5 Conclusiones

El presente trabajo se dirigió al desarrollo de un sistema de inventario de hardware y software para redes de computadoras, un software enfocado en las necesidades de los administradores de redes, capaz de competir con las mejores aplicaciones privativas existentes en el mercado internacional, por ser los sistemas más completos en la actualidad. Este sistema permite a una determinada institución o empresa mantener el control de una manera automatizada de sus activos informáticos, he aquí la importancia del manejo del inventario por parte de la misma.

Culminando el proceso de desarrollo del software se obtuvo un sistema escalable y extensible, que obtiene información de software y hardware en una red de computadoras, integrando sistemas de alerta; dando cumplimiento a las metas propuestas con anterioridad en el documento, así como a la estrategia planteada por la administración central de Cuba a la migración de sus aplicaciones hacia el software libre.

6 Recomendaciones

Se recomienda seguir trabajando en la implementación de la aplicación Cliente - Servidor para:

- Extender y mejorar el sistema de alertas.
- Incrementar la recolección de información del Agente.
- Lograr la escalabilidad en los servidores así como un incremento en funcionalidades que hagan del sistema una herramienta indispensable para el monitoreo de hardware y software en nuestro país y principalmente en nuestra universidad.
- Ofrecer la posibilidad de que el cliente se comporte como servidor o viceversa para poder escalonar el sistema en una red.
- Enviar acciones a realizar en el cliente como determinación del inventario en un determinado momento.
- Confirmar por parte del cliente el inventario.
- Integrar el agente con agentes de monitoreo como el agente de la herramientas Nagios con el objetivo de monitorear rendimiento de la pc cliente, uso de la memoria, disco duro, etc.
- Instalar la aplicación en todas las redes del país para el control de hardware y el software de las mismas.

Referencias Bibliográficas

1Definición de Sockethttp://www.masadelante.com/faqs/socket

2. OCS Inventory. [En línea] OCS Inventory. [Citado el: 16 de Mayo de 2010.] http://www.ocsinventory-ng.org/.

3. Cacic. [En línea] [Citado el: 21 de Mayo de 2010.] http://guialivre.governoeletronico.gov.br/cacic/sisp2/.

4. NetSupport DNA. [En línea] NetSupport DNA. [Citado el: 16 de Mayo de 2010.] http://www.netsupportdna.com/es/.

5. **Rodriguez, Medardo.** Entrevista a Medardo Rodriguez. [En línea] 2008. [Citado el: 16 de Mayo de 2010.] http://www.mastermagazine.info/articulo/12671.php.

6. Visual Paradigm for UML. *Visual Paradigm.* [En línea] Visual Paradigm. [Citado el: 16 de Mayo de 2010.] http://www.visual-paradigm.com/product/vpuml/.

7About Pyrohttp://pyro.sourceforge.net

8. **Vicente Aguilar, Pablo Suau.** MySQL vs. PostgreSQL. [En línea] 2004. [Citado el: 16 de Mayo de 2010.] http://www.fedora-es.com/node/189.

9. Características PostgreSQL. [En línea] [Citado el: 15 de Mayo de 2010.] http://www.postgresql.org/about/.

10. EasyEclipse. [En línea] [Citado el: 16 de Mayo de 2010.] http://mundogeek.net/archivos/2006/06/05/easyeclipse/.

11. EasyE + Pydev. [En línea] [Citado el: 16 de Mayo de 2010.] http://py.arahat.net/2008/08/pydev-extensin-de-eclipse-para-python.html.

12. Arquitectura y Patrones de diseño. [En línea] [Citado el: 16 de Mayo de 2010.] http://eva.uci.cu/mod/resource/view.php?id=14075.

Bibliografía

1Definición de Sockethttp://www.masadelante.com/faqs/socket

2. OCS Inventory. [En línea] OCS Inventory. [Citado el: 16 de Mayo de 2010.] http://www.ocsinventory-ng.org/.

3. Cacic. [En línea] [Citado el: 21 de Mayo de 2010.] http://guialivre.governoeletronico.gov.br/cacic/sisp2/.

4. NetSupport DNA. [En línea] NetSupport DNA. [Citado el: 16 de Mayo de 2010.] http://www.netsupportdna.com/es/.

5. **Rodriguez, Medardo.** Entrevista a Medardo Rodriguez. [En línea] 2008. [Citado el: 16 de Mayo de 2010.] http://www.mastermagazine.info/articulo/12671.php.

6. Visual Paradigm for UML. *Visual Paradigm.* [En línea] Visual Paradigm. [Citado el: 16 de Mayo de 2010.] http://www.visual-paradigm.com/product/vpuml/.

7About Pyrohttp://pyro.sourceforge.net

8. **Vicente Aguilar, Pablo Suau.** MySQL vs. PostgreSQL. [En línea] 2004. [Citado el: 16 de Mayo de 2010.] http://www.fedora-es.com/node/189.

9. Características PostgreSQL. [En línea] [Citado el: 15 de Mayo de 2010.] http://www.postgresql.org/about/.

10. EasyEclipse. [En línea] [Citado el: 16 de Mayo de 2010.] http://mundogeek.net/archivos/2006/06/05/easyeclipse/.

11. EasyE + Pydev. [En línea] [Citado el: 16 de Mayo de 2010.] http://py.arahat.net/2008/08/pydev-extensin-de-eclipse-para-python.html.

12. Arquitectura y Patrones de diseño. [En línea] [Citado el: 16 de Mayo de 2010.] http://eva.uci.cu/mod/resource/view.php?id=14075.

13. **Martínez, Gerardo Moreno.** Análisis y Diseño de Sistemas. [En línea] [Citado el: 16 de Mayo de 2010.] http://www.monografias.com/trabajos5/andi/andi.shtml#dise.

14*DmiDecode for Windows*http://gnuwin32.sourceforge.net/packages/dmidecode.htm

15*HAL - Hardware Abstraction Layer*http://www.freedesktop.org/wiki/Software/hal

16*Introduction To D-BUS*http://techbase.kde.org/Development/Tutorials/D-Bus/Introduction_%28es%29

17*Mensajes por D-BUS*http://library.gnome.org/devel/platform-overview/stable/dbus.html.es

18. *org.freedesktop.DeviceKit.Disks.Device.* [En línea] [Citado el: 14 de Febrero de 2010.] http://hal.freedesktop.org/docs/DeviceKit-disks/Device.html.

19. *Computer System Hardware Classes.* [En línea] [Citado el: 17 de Enero de 2010.] http://msdn.microsoft.com/en-us/library/aa389273%28v=VS.85%29.aspx.

20. *WMI Application Monitoring Classes - Sample Code .* [En línea] [Citado el: 11 de Diciembre de 2009.] http://mirror2.activexperts.com/admin/wmi/category/0001/.

21. *DmiDecode.* [En línea] [Citado el: 13 de Noviembre de 2009.] http://www.nongnu.org/dmidecode/.

22. *DmiDecode - DMI table decoder.* [En línea] [Citado el: 09 de Diciembre de 2009.] http://linux.die.net/man/8/dmidecode.

23. *ORM - SQLAlchemy.* [En línea] [Citado el: 09 de Marzo de 2010.] http://www.sqlalchemy.org/.

24. *Patrones UML.* [En línea] [Citado el: 16 de Febrero de 2010.] http://eva.uci.cu/.

25*El Lenguaje Unificado de Modelado. Manual de Referencia*Addison-Wesley1998

26Metodologías para la auditoría de informaciónhttp://www.documentalistaenredado.net/77/metodologas-para-la-auditora-de-informacin/