UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Requerimientos y diseño del módulo Acciones Centralizadas del Sistema Nacional Público para el Seguimiento de Inversiones y Sectores.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores: Arianna Leyva Campos.
Eduardo Castillo Benítez.

Tutor: Ing. Juan Carlos Montané Izaguirre.
Cotutora: Ing. Maylen Bon Pérez

Ciudad de la Habana, 2009-2010

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
(Autor)	(Autor)
Juan Carlos Montané Izaguirre	Maylen Bon Pérez
_____	_____
(Tutor)	(Cotutora)

AGRADECIMIENTOS

A mi mamá por consentirme, por sentir mi dolor y mis alegrías, por inspirarme y salvarme a diario.

A mi papá por enseñarme que no hay meta que no alcance el esfuerzo y empeño, por su paciencia y cariño y por el ejemplo que me ha dado.

A mi hermanita, por mimarme, por evitar cualquier acción o palabra que pueda herirme y por su afán de hacerme feliz. Ella es mi más preciado tesoro, lo que más quiero y cuido.

A ellos tres porque son los que me incitan a quererme superar cada día y ser una mejor persona. Son mi refugio, cuando estoy confundida o mi corazón está triste solo pensar en ellos hace que todo tenga sentido y por hacerme sentir la persona más afortunada del mundo.

A mis abuelitos por darme tanto amor especialmente a Campo y a Gladys.

A los primitos que se criaron conmigo Yunel, Daily, Osmelky y Osmarcito.

A todos mis tíos por estar pendientes de mí y por el cariño que me han dado.

A Aurelio por su paciencia y apoyo.

A mis amigos por aceptarme y hacerme sentir querida y especial. Si sucede algo malo no tengo miedo sé que puedo contar con ustedes.

A todos los maestros y profesores que han contribuido a mi formación.

Ary

AGRADECIMIENTOS

Ante todo, incluso antes de respirar, agradecerle a mis padres, a los dos juntos, son lo más grande que tengo en la vida y mi vida se las debo a ustedes, me han criado con tanto afán y cariño que para mí es imposible verlos por separado, los quiero a los dos, los amo a los dos, y por dejarme existir, les agradezco a los dos.

A mis tías, a mis tíos, abuelos, primos y a mis vecinos por su apoyo y su amor incondicional, para mí son el vínculo y el soporte más fuerte que poseo…

A mis amistades y empiezo por las damas, amigas tan especiales y cariñosas que las considero hermanitas de corazón, sin ustedes es imposible ser feliz porque desde que las conocí se ganaron un pedacito de mi corazón y ahí estarán por siempre, gracias por existir, gracias por cruzarse en mi camino, gracias por darme tanto y ojalá me alcance la vida para devolverles aunque sea un poquito de lo mucho que me han dado. Gracias también a una persona muy especial, que ha significado mucho para mí, gracias a ella por dejarme saber que existen cosas muy buenas en este mundo, más allá del alcance de las palabras y del entendimiento humano, por enseñarme donde consigue un hombre su felicidad y lo mucho que duele perderla, gran parte de lo que soy, de lo que quiero y de lo que sé, se lo debo a ella y le agradezco por eso.

A los colegas, hermanos-amigos, a los piquetes de los desbombes, ja, dicen que las amistades de la uni nos duran para toda la vida y ahora sé que eso es absolutamente cierto, es difícil poder resumir en tan sólo una página todo lo que significan para mí y lo mucho que los quiero, admiro y respeto, difícil no, en realidad creo que es imposible, primero, gracias le doy al mundo y a la vida el que existan personas como ustedes, segundo que me hayan concedido la oportunidad de conocerlos y finalmente, gracias a ustedes por compartir en estos años conmigo, sin lugar a dudas son lo mejor que me ha pasado en esta escuela.

A los profes Joan, Noel, Coppen, a las mega profes de Inglés, Ingeniería de Software y Programación, a Zárate, profe primero y tutor después, para él todo el respeto y admiración que puede obtener un maestro de su discípulo, personas con su calidad quedan pocas, en Cuba y en el mundo, a todos ellos gracias, indiscutiblemente parte de lo que soy y de lo que he logrado se los debo a ustedes.

A esta magnífica escuela y a su creador nuestro Comandante Fidel, por las experiencias, alegrías y satisfacciones que aquí he vivido y estoy seguro, jamás olvidaré…

DEDICATORIA

A mi mamá, mi papá y mi hermanita porque todo lo que soy es gracias a ellos.
Ary

A mis padres porque han sido, son y serán mi ejemplo, mi guía y mi inspiración…
A toda mi familia, porque sé que un triunfo mío es orgullo y victoria de ellos también...
Eduard

RESUMEN

El sistema informático “Nueva Etapa” utilizado en Venezuela para la planificación del presupuesto anual del país presenta un conjunto de dificultades que han provocado que este proceso no se realice de manera óptima. Por ello se hizo necesaria la implementación del Sistema Nacional Público para el Seguimiento de Inversiones y Sectores (SINAPSIS) que permitirá realizar efectivamente este proceso, informatizando la planificación y seguimiento de los proyectos y la gestión de las acciones centralizadas.

El módulo Acciones Centralizadas de SINAPSIS automatiza la formulación y aprobación de solicitudes de recursos para las actividades regulares que aseguran el correcto funcionamiento de los organismos. Los artefactos de software existentes del módulo no garantizan el comienzo y continuidad de su implementación, siendo el propósito fundamental de la investigación **generar los artefactos necesarios del módulo Acciones Centralizadas que permitan el comienzo y continuidad de su implementación**.

La fundamentación teórica de la investigación se basó en el estudio de aspectos teóricos y técnicos de la disciplina Ingeniería de Software, lo que permitió adquirir los conocimientos necesarios para especificar y diseñar el módulo. Como resultado del presente trabajo se obtuvieron los artefactos: Especificación de requerimientos del sistema, Modelo de casos de uso del sistema y Modelo de diseño a los que se le aplicaron métricas y listas de chequeos para valorar su calidad.

Palabras claves: especificación del módulo.

ÍNDICE DE FIGURAS

ÍNDICE

INTRODUCCIÓN.

Desde el primero de enero de 2006 la República Bolivariana de Venezuela lleva a cabo el presupuesto por proyectos por disposición de su Presidente, el Comandante Hugo Rafael Chávez Frías. A través del presupuesto el gobierno proporciona bienes y servicios para atender las necesidades de la comunidad, paga a proveedores y contratistas y resuelve problemas de pasivos laborales y contractuales con sus trabajadores.

Para poner en práctica la gestión del presupuesto por proyectos, el Ministerio del Poder Popular para la Planificación y Desarrollo (MPPPD), conjuntamente con el Ministerio de Economía y Finanzas y la Vicepresidencia de la República, decidieron implantar un sistema que facilitara el registro, seguimiento y control de todos los proyectos y las acciones centralizadas que formarían parte del presupuesto anual de la nación, por lo que se implementa la solución informática “Nueva Etapa”.

Actualmente este sistema no satisface todos los requerimientos de los clientes y usuarios ya que presenta un conjunto de dificultades: No está disponible para sus usuarios durante todo el año, provocando que los proyectos y acciones centralizadas (AC) se formulen y planifiquen con estimaciones de costos y tiempos irreales. No existe un proceso homogéneo de registro y aprobación de los proyectos y AC para todos los órganos y entes. Los flujos y niveles de aprobación por los que transita una acción centralizada no están bien definidos. Los procesos para: la solicitud de nuevas denominaciones de AC; la creación, modificación y eliminación de las AC; y la aprobación de las AC y solicitudes de nuevas denominaciones de AC no están informatizados.

El desarrollo de Nueva Etapa se realizó sin utilizar una metodología de desarrollo de software por lo que no cuenta con la documentación necesaria que facilite su comprensión, dificultando así, su optimización y mantenimiento. Para darle solución a la problemática planteada anteriormente se decidió desarrollar el Sistema Nacional Público para el Seguimiento de Inversiones y Sectores (SINAPSIS), un sistema informático que sustituirá completamente el anterior y se dividirá en siete módulos, entre los que se encuentra el de Acciones Centralizadas.

Dicho módulo se encarga de gestionar la formulación y aprobación de solicitudes de recursos para las actividades regulares que garantizan el correcto funcionamiento de los organismos. Además se gestionan las solicitudes de nuevas denominaciones de AC para cubrir las necesidades que puedan surgir en el momento de llevar a cabo las operaciones institucionales.

Después de tener identificados y documentados los procesos de negocio se deben elaborar un conjunto de artefactos (diagramas, modelos) que le proporcionen a los desarrolladores las bases para implementar el módulo exitosamente.

Problema Científico Investigativo.

Los artefactos de software existentes del módulo Acciones Centralizadas de SINAPSIS no garantizan el comienzo y continuidad de su implementación.

Objeto de estudio.

Proceso de desarrollo del software.

Campo de acción.

Requerimientos y diseño.

Objetivo.

Generar los artefactos necesarios del módulo Acciones Centralizadas de SINAPSIS que permitan el comienzo y continuidad de su implementación.

Tareas de la investigación.

1. Estudio del proceso de desarrollo de software.
2. Estudio de herramientas utilizadas para especificar y diseñar sistemas.
3. Estudio de técnicas para la especificación y diseño de software.
4. Elaboración de la Especificación de requerimientos del sistema.
5. Confección del Modelo de casos de uso del sistema.
6. Confección del Modelo de diseño.
7. Validación de la Especificación de requerimientos del sistema.
8. Validación del Modelo de casos de uso del sistema.

9. Validación del Modelo de diseño.

Métodos y técnicas de investigación a utilizar.

Métodos Científicos.

Teóricos:

- Histórico – lógico: Para el estudio del estado del arte del proceso de desarrollo de software y las técnicas de especificación y diseño de software más usadas.
- Modelación: Para la creación de modelos y diagramas que reflejen la especificación y el diseño del módulo Acciones Centralizadas.

Empíricos:

- Entrevista: Para interactuar con los clientes e identificar los requerimientos del sistema y con analistas, diseñadores y especialistas de otros proyectos que aporten sus experiencias en la especificación y diseño de sistemas.

Resultados esperados.

- Especificación de requerimientos del sistema.
- Modelo de casos de uso del sistema.
- Modelo de diseño.

Breve descripción de la estructuración del contenido.

A continuación se describe brevemente el contenido de los tres capítulos que conforman el documento.

Capítulo 1. Fundamentación teórica.

En este capítulo se tratan aspectos teóricos y técnicos necesarios para realizar satisfactoriamente la especificación y el diseño del módulo Acciones Centralizadas: Se hace una descripción de varias metodologías de desarrollo de software. Se analizan las etapas y técnicas de la Ingeniería de Requerimientos. Se estudian un conjunto de métricas encaminadas a valorar la calidad de la especificación y diseño del módulo. Se exponen los principales patrones de casos de uso y de diseño, así como las herramientas y lenguajes más usados en el desarrollo de software a nivel mundial. Finalmente se muestran cuales son las herramientas, lenguajes, metodologías y tecnologías que se usarán en el desarrollo de la investigación.

Capítulo 2. Solución propuesta. Requerimientos y diseño del sistema.

En este capítulo se describe la solución que se propone para la especificación y diseño del módulo Acciones Centralizadas de SINAPSIS. Se explica cómo fueron aplicadas las técnicas de la disciplina Ingeniería de Requerimientos para definir y especificar las condiciones y capacidades que el sistema debe cumplir. Se definen y describen los actores y casos de uso del sistema y se confecciona el diagrama de casos de uso. Finalmente se elaboran los artefactos resultantes de las actividades de diseño obteniéndose subsistemas de diseño, diagramas de clases, diagramas de secuencia y el modelo de datos.

Capítulo 3. Análisis de los resultados.

En este capítulo se describe el análisis de los resultados obtenidos, mostrándose cómo fueron aplicados un conjunto de procedimientos y métodos que permitieron valorar objetivamente la calidad de la Especificación de requerimientos del sistema, del Modelo de casos de uso del sistema y del Modelo de diseño, demostrándose que los mismos poseen la calidad requerida.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En este capítulo se tratan aspectos teóricos y técnicos necesarios para realizar satisfactoriamente la especificación y el diseño del módulo Acciones Centralizadas: Se hace una descripción de varias metodologías de desarrollo de software. Se analizan las etapas y técnicas de la Ingeniería de Requerimientos. Se estudian un conjunto de métricas encaminadas a valorar la calidad de la especificación y diseño del módulo. Se exponen los principales patrones de casos de uso y de diseño, así como las herramientas y lenguajes más usados en el desarrollo de software a nivel mundial. Finalmente se muestran cuales son las herramientas, lenguajes, metodologías y tecnologías que se usarán en el desarrollo de la investigación.

1.2 Introducción a la Ingeniería de Software.

La Ingeniería de Software es una disciplina o Área de la Informática, que ofrece métodos y técnicas para desarrollar y mantener software de calidad. Aporta una serie de conocimientos que permiten comprender los procesos, datos, e interrelaciones que se establecen en un modelo real. Proporciona los elementos que hacen que la interacción con el usuario se torne más amigable, sin descuidar por ello la asignación de responsabilidades y la determinación de los roles de los participantes de un proyecto. A lo largo de la última década la Ingeniería de Software ha sufrido una evolución considerable en una gran parte de las distintas áreas que la constituyen, convirtiéndose en una disciplina ampliamente aceptada tanto a nivel académico como industrial. (Pressman, 2005).

1.3 Metodologías de desarrollo de software.

El fundamento de la Ingeniería del Software es la capa de proceso la cual es la unión que mantiene juntas las capas de tecnología y que permite un desarrollo racional y oportuno de la misma. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar. A continuación se relaciona un estudio de un conjunto de ellas con el objetivo de determinar la más adecuada para el desarrollo del sistema en cuestión.

1.3.1 El Proceso Unificado de Desarrollo (RUP).

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo a través de UML y de las experiencias de muchas metodologías utilizadas por los clientes. Como es un proceso, en su modelación define como sus principales elementos a los **trabajadores** (“quién”), las **actividades** (“cómo”), los **artefactos** (”qué”) y el **flujo de actividades** (“cuándo”). Las actividades se organizan en grupos lógicos definiéndose 9 flujos de trabajo. En la Figura 1 se representan gráficamente los flujos de trabajo y las 4 fases. (Jacobson, y otros, 2000)

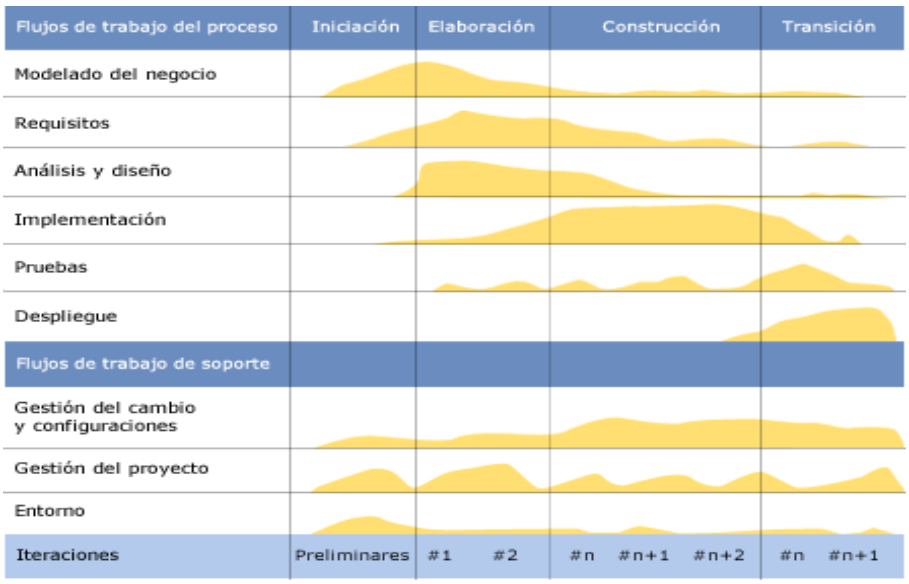


Figura 1 RUP en dos dimensiones.

El ciclo de vida de RUP se caracteriza por: (Jacobson, y otros, 2000)

1. **Dirigido por casos de uso:** Estos reflejan las necesidades de los clientes, extraídas en el modelamiento del negocio y representadas después a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

2. **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

3. **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

En RUP la captura de requerimientos y el modelamiento del sistema son las actividades fundamentales que se desarrollan en el Flujo de Requerimientos durante la fase de inicio del desarrollo de un software. Mientras que en el Flujo de Análisis y Diseño se traducen los requerimientos a una especificación que describe cómo implementar el sistema. (Universidad de las Ciencias Informáticas, 2009)

1.3.2 Extreme Programming (XP).

La metodología XP promueve cuatro valores: **comunicación**, **retroalimentación**, **simplicidad** y **coraje**. Construye sobre ellos una docena de prácticas que los proyectos deben seguir. Muchas de estas prácticas son técnicas antiguas, tratadas y probadas. XP retoma estas técnicas entrelazándolas en un todo donde cada una colabora y le da soporte a las demás.

Prácticas de Extreme Programming:

- **Planificación incremental:** asume que la planificación nunca será perfecta y que variará en función de cómo varíen las necesidades del negocio.
- **Testing:** la ejecución automatizada de pruebas es un elemento clave de la XP. El cliente es el responsable de definir las pruebas de aceptación, no necesariamente de implementarlas.
- **Programación en parejas:** nadie programa en solitario, siempre hay dos personas delante del ordenador.
- **Refactorización:** mantener el código en buen estado modificándolo activamente para que conserve claridad y sencillez.
- **Diseño simple:** utilizar el diseño más sencillo que consiga que todo funcione.
- **Propiedad colectiva del código:** todo el mundo tiene autoridad para hacer cambios a cualquier código y es responsable de ellos.
- **Integración continua:** evitar que se acumulen defectos, los defectos que cada programador inyecta los elimina él mismo.
- **Cliente en el equipo:** integra un representante del negocio dentro del equipo de desarrollo.

- **Releases pequeñas:** dar el máximo valor posible en cada momento, se intenta liberar nuevas versiones de las aplicaciones con frecuencia.
- **Semanas de 75 horas:** equipo siempre al 100%, se trabaja 15 horas al día 5 días a la semana.
- **Estándares de codificación:** estilo de codificación consistente.
- **Uso de Metáforas:** utilizar el vocabulario del negocio, enfatizando el qué por delante del cómo.

Tanto el análisis como el diseño son tareas importantes en XP. El análisis es parte fundamental ya que se intentan recoger en él todas las necesidades del usuario. De él surgen las "user stories", equivalentes a la captura y especificación de los requerimientos que servirán para empezar a comenzar a desarrollar el sistema. Las labores de diseño son similares a las que se realizan en otras metodologías. (Beck, 2002)

1.3.4 Microsoft Solutions Framework.

Microsoft Solutions Framework (MSF) es un marco de trabajo para construir e implantar sistemas empresariales distribuidos basados en herramientas y tecnologías de Microsoft. MSF comprende un conjunto de modelos, conceptos y guías que contribuyen a alinear los objetivos de negocio y tecnológicos, reducir los costos de la utilización de nuevas tecnologías, y asegurar el éxito en la implantación de las tecnologías Microsoft. Es el resultado de las experiencias de diferentes áreas de Microsoft con proyectos exitosos. (Microsoft Corporation, 2000)

Los cinco modelos de MSF son:

- Modelo de Arquitectura Empresarial de MSF (Enterprise Architecture Model).
- Modelo de Aplicaciones de MSF (Application Model).
- Modelo de Equipos de Trabajo de MSF (Team Model).
- Modelo de Procesos de MSF (Process Model).
- Proceso de Diseño de Soluciones con Componentes (Designing Component Solutions - DCS).

Posee 5 fases: Fase de visión donde se realizan la especificación funcional del proyecto, el plan de gestión de riesgos y el plan maestro del proyecto. Fase de planificación en la que el equipo prepara la especificación funcional, los planes de trabajo, los costos y calendarios. Fase de desarrollo en la que se obtienen el código fuente y el ejecutable así como los instaladores y la documentación de apoyo como son los manuales de usuario. Fase de estabilización en la cual se deja el sistema establecido ya que no se

sabe si el ambiente real está listo para ir a operación. El código fuente, el ejecutable y la documentación deben estar disponibles. Fase de utilización donde se realiza el despliegue y se le pregunta a los usuarios hasta qué punto están satisfechos con el producto.

1.4 Ingienería de Requerimientos.

Todas las metodologías incluyen actividades para la obtención de requerimientos, por ser este un proceso complejo se necesita de una disciplina que facilite un mecanismo apropiado para realizarlo. La Ingeniería de Requerimientos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software.

Definición de requerimientos según la IEEE:

- Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
- Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
- Un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste.

Clasificación de requerimientos.

- Requerimientos funcionales: Describen los servicios que proporciona el sistema (funciones), la respuesta del sistema ante determinadas entradas y el comportamiento del sistema en situaciones particulares.
- Requerimientos no funcionales: Son restricciones de los servicios o funciones que ofrece el sistema (ej. cotas de tiempo, proceso de desarrollo, rendimiento, etc.). Estos pueden clasificarse en requerimientos del producto, requerimientos organizativos y requerimientos externos.

1.4.1 Fases de la Ingeniería de Requerimientos.

Existen varios criterios sobre la cantidad de fases de la Ingeniería de Requerimientos. Un grupo de autores considera 5 fases: elicitación, análisis, especificación, validación y gestión de requerimientos. Otros autores proponen la unión del análisis y la especificación, refiriéndose a esta última el Licenciado en Análisis de Sistemas Nicolás Davyt Dávila expresó: “En la práctica, esta etapa se va realizando

conjuntamente con el análisis, pero podríamos decir que la especificación es el "pasar en limpio" el análisis realizado previamente aplicando técnicas y/o estándares de documentación, como la notación UML.” El desarrollo de la presente investigación se regirá por la segunda vertiente procediéndose con 4 etapas: elicitación, análisis, validación y gestión de requerimientos.

1.4.1.1 Elicitación.

La etapa de elicitación de requerimientos abarca la primera y quizás más importante fase dentro del desarrollo de un sistema informático. Uno de los retos más importantes de la elicitación de requerimientos es garantizar que los requerimientos del sistema sean consistentes con las necesidades de la organización donde se utilizará el mismo y con las futuras necesidades de los usuarios.

A continuación se presentan un grupo de técnicas que de forma clásica han sido utilizadas para esta etapa en el proceso de desarrollo de todo tipo de software:

- Entrevistas: resultan una técnica muy aceptada dentro de la Ingeniería de Requerimientos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural.
- JAD (Joint Application Development/Desarrollo conjunto de aplicaciones): esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Está basada en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación WYSIWYG (What You See Is What You Get/lo que ves es lo que obtienes). (IBM, 1997)
- Brainstorming (Tormenta de ideas): es una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. (Raghavan, y otros, 1994)
- Concept Mapping (Mapas conceptuales): Los mapas conceptuales son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. (Pan, y otros, 2001)
- Sketches y Storyboards: Está técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno web. La misma consiste en representar sobre papel en forma muy

esquemática las diferentes interfaces al usuario (sketches). Estos sketches pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación (storyboard).

- Casos de Uso: Aunque inicialmente se desarrollaron como técnica para la definición de requerimientos, algunos autores proponen casos de uso como técnica para la captura de requerimientos. Los casos de uso permiten mostrar el contorno (actores) y el alcance (requerimientos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. (Jacobson, y otros, 2000)

1.4.1.2 Análisis.

En esta etapa se estudian los requerimientos extraídos en la etapa previa a los efectos de poder detectar, entre otros, la presencia de áreas no especificadas, requerimientos contradictorios y peticiones que aparecen como vagas e irrelevantes. El resultado de haber llevado a cabo las tareas que involucran estos términos puede, en más de una oportunidad, hacer que se deba regresar a la primera etapa, a los efectos de eliminar todas las inconsistencias que se han detectado. Además se realizan aproximaciones a un lenguaje técnico y se modelan los resultados utilizando diagramas UML.

Técnicas válidas para esta etapa:

- Glosario y ontologías: La diversidad de personas que forman parte de un proyecto de software hace que sea necesario establecer un marco de terminología común. Esta necesidad se vuelve más patente en los sistemas de información web puesto que el equipo de desarrollo en ellas suele ser más interdisciplinario. (Koch, 2001)
- Plantillas o patrones: Esta técnica, recomendada por varios autores, tiene por objetivo el describir los requerimientos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando, usando para ello el lenguaje del usuario. (Duran, y otros, 1999)
- Escenarios: La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia una representación gráfica en forma de diagramas de flujo. (Weidenhaupt, y otros, 1999)

- Lenguajes Formales: Otro grupo de técnicas que merece la pena resaltar como extremo opuesto al lenguaje natural, es la utilización de lenguajes formales para describir los requerimientos de un sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción formal, han sido aplicadas en el mundo de la Ingeniería de Requerimientos desde hace años. Sin embargo, resultan muy complejas en su utilización y para ser entendidas por el cliente. El mayor inconveniente es que no favorecen la comunicación entre clientes y analistas. (Peña, 1998)

1.4.1.3 Validación.

La validación es la etapa final de la IR. Su objetivo es, ratificar los requerimientos, es decir, verificar todos los requerimientos que aparecen en el documento especificado para asegurarse que representan una descripción aceptable del sistema que se debe implementar. Esto implica verificar que los requerimientos sean consistentes y que estén completos.

Técnicas que pueden aplicarse:

- Reviews o Walk-throughs: Está técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requerimientos. Con ello solamente se puede validar la correcta interpretación de la información transmitida.
- Auditorías: La revisión de la documentación consiste en una revisión de los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada.
- Matrices de trazabilidad: Esta técnica consiste en marcar los objetivos del sistema y chequear que estén reflejados en los requerimientos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos. (Duran, y otros, 1999)
- Prototipos: Algunas propuestas se basan en obtener de la definición de requerimientos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final. (Olsina, 1999)

1.4.1.4 Gestión de requerimientos.

Los cambios de los requerimientos durante el desarrollo del sistema son inevitables, por tal razón se recomienda realizar un proceso de gestión. “La gestión de requerimientos es un conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requerimientos y los cambios en cualquier momento”. (Pressman, 2005)

Para la gestión de requerimientos se procede primeramente a identificar los requerimientos y luego se lleva el control de éstos a través de matrices de trazabilidad basadas en varios conceptos, dentro de éstas se tiene:

- Matriz de seguimiento de características.
- Matriz de seguimiento de orígenes.
- Matriz de seguimiento de dependencias.
- Matriz de seguimiento de subsistemas.
- Matriz de seguimiento le interfaces.

El control de los cambios de los requerimientos es importante para el buen desarrollo del sistema, por lo que es recomendable además de las actividades expuestas anteriormente para la gestión de los requerimientos, tener en cuenta otras de las actividades pertenecientes a la Gestión de Configuración del Software (GCS), que son de gran utilidad para la gestión de los requerimientos. (Pressman, 2005)

1.5 Patrones.

Los patrones constituyen un conjunto de buenas prácticas resultantes de las experiencias obtenidas en el proceso de desarrollo de software. Un patrón es una pareja de problema / solución con un nombre, que estandariza buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Compuesto por Nombre, Solución, Problema y Explicación. El uso de los mismos contribuye a lograr mejores resultados de forma más rápida, por ello se realizó un estudio de patrones de casos de uso y de diseño.

1.5.1 Patrones de casos de uso.

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requerimientos reales, haciendo más fácil el trabajo con los sistemas, y

mucho más simple su mantenimiento. Permiten resolver problemas comunes que se les planteen a los analistas de una forma ágil y sistemática. Estos patrones se enfocan hacia el diseño y las técnicas utilizadas en modelos de alta calidad, y no en cómo modelar usos específicos.

Los patrones de casos de uso son los siguientes: (Övergaard, et al., 2004)

- Reglas de negocio.
- Concordancia (Commonality).
- Componente jerárquico (Component hierarchy).
- Extensión concreta o Inclusión.
- CRUD (Creating, Reading, Updating, Deleting).
- Caso de uso grande (Large Use case).
- Sistema de Capas.
- Múltiples actores.
- Servicio opcional.
- Vistas ortogonales.
- Secuencia de casos de uso.

De estos mencionados sólo abordaremos cuatro patrones: Concordancia, CRUD, Sistema de capas y Múltiples actores.

1.5.1.1 Concordancia (Commonality).

Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

Reusabilidad.

Consta de 3 casos de uso. El primero llamado subsecuencia común, modela una secuencia de acciones que aparecerán en múltiples casos de uso en el modelo. Los otros casos de uso modelan el uso del sistema que comparte la subsecuencia común de acciones. De manera que deben existir al menos dos de ellos.

Adición.

En el caso de este patrón alternativo, la subsecuencia común de casos de uso, extiende los casos de uso compartiendo la subsecuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la subsecuencia. Este patrón es preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una subsecuencia común de acciones para modelar los usos completos del sistema.

Especialización.

Otro patrón de concordancia que contiene casos de uso del mismo tipo. En este caso, estos son modelados como una especialización de casos de uso de tipo de uso común. Todas las acciones en estos casos de uso son heredadas por los casos de uso hijos, donde otras acciones serán adicionadas o acciones heredadas que serán especializadas. Este patrón es aplicable cuando la utilización de los casos de uso que han sido modelados son del mismo tipo, y este tipo debe hacerse visible en el modelo.

1.5.1.2 CRUD (Creating, Reading, Updating, Deleting).

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

Completo.

Este patrón consta de un caso de uso, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

Parcial.

Este patrón alternativo modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es más significativa, larga o más compleja que las otras.

1.5.1.3 Múltiples actores.

Roles diferentes.

Captura la concordancia entre actores manteniendo roles separados. Consiste de un caso de uso y por lo menos dos actores. Es utilizado cuando dos actores juegan diferentes roles en un caso de uso, o sea, interactúan de forma diferente con el caso de uso.

Roles comunes.
Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso.

1.5.2 Patrones de Diseño.

El concepto de patrones de diseño fue el resultado de un trabajo realizado por un grupo de 4 personas (Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) que se publicó en 1995 en un libro titulado "Patrones de diseño: Elementos de software orientado a objetos reutilizables" en el que se esbozaban 23 patrones de diseño.

Un patrón de diseño puede considerarse como un documento que define una estructura de clases que aborda una situación particular. Los patrones de diseño se dividen en tres grupos principales:

1.5.2.1 Patrones de creación.

- Abstract Factory. Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.
- Builder. Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.
- Factory Method. Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
- Prototype. Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.
- Singleton. Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

1.5.2.2 Patrones estructurales.

- Adapter. Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.

- Bridge. Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.
- Composite. Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.
- Decorator. Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.
- Facade. Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.
- Flyweight. Usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.
- Proxy. Proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.

1.5.2.3 Patrones de comportamiento.

- Chain of Responsibility. Evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.
- Command. Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones.
- Interpreter. Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.
- Iterator. Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.
- Mediator. Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.
- Memento. Representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.
- Observer. Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.

- State. Permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Parecerá que cambia la clase del objeto.
- Strategy. Define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.
- Template Method. Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.
- Visitor. Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

1.6 Métricas.

Las métricas juegan un papel fundamental en la construcción de productos de software. Estas proporcionan una indicación de la calidad de los mismos en función de sus atributos y características, permitiendo obtener una visión más profunda de los artefactos que se crean durante el ciclo de vida de un proyecto y de esta forma corregirlos para que alcancen un nivel superior de calidad. (Pressman, 2005)

Seguidamente se hará un estudio de varias métricas encaminadas a validar la Especificación de requerimientos y el diseño en el cual se recogerán conocimientos que constituirán la base para analizar los resultados que se obtengan con la presente investigación.

1.6.1 Métricas para la Especificación de requerimientos.

Existen un conjunto de características de la Especificación de requerimientos que se representan usando métricas y son utilizadas para valorar su calidad. A continuación se explican algunas de estas características y sus métricas correspondientes.

1.6.1.1 Ausencia de ambigüedad de los requerimientos.

Está métrica se basa en la consistencia de la interpretación de los revisores para cada requisito y se representa con la ecuación $Q_i = n_{ui} / n_r$ donde n_{ui} es el número de requerimientos para los que todos los revisores tuvieron interpretaciones idénticas y n_r es el total de requerimientos. La ausencia de ambigüedad es un factor crítico para la calidad de la especificación por lo que se recomienda que el resultado al aplicar esta métrica alcance el valor máximo es decir 1. (Davis, et al., 1993)

1.6.1.2 Compleción de los requerimientos funcionales.

Esta métrica se fundamenta en la completitud de la Especificación de requerimientos y se puede calcular usando la ecuación $Q_2 = \frac{n_A}{n_A + n_B}$ donde n_A es el número de requerimientos funcionales completos y n_B es el número de requerimientos funcionales pobremente especificados. Como la compleción es difícil de medir, los valores obtenidos una vez aplicada esta métrica deben oscilar entre 0.7 y 1. (Davis, et al., 1993)

1.6.1.3 Corrección de los requerimientos.

Una especificación se considera correcta cuando cada requisito contenido en ella represente una característica que el sistema debe poseer. La corrección de los requerimientos se define usando la ecuación $Q_3 = \frac{n_c}{n_c + n_{nv}}$ donde n_c es el número de requerimientos que se han validado como correctos y n_{nv} es el número de requerimientos que no se han validado como correctos todavía. Se recomiendan valores cercanos a 1 para esta métrica. (Davis, et al., 1993)

1.6.2 Métricas para el diseño.

Las métricas de diseño permiten evaluar varios aspectos de la calidad del mismo. En el presente epígrafe se describirán las siguientes:

- Métricas de diseño arquitectónico.
- Métricas de diseño a nivel de componente.
- Métricas orientadas a clases.

1.6.2.1 Métricas de diseño arquitectónico.

Tienen en cuenta las características relativas a la estructura y eficiencia de los componentes que forman la arquitectura de un sistema.

La complejidad estructural $S(i)$ de un módulo i se determina usando la ecuación $S(i) = f_{out}^2(i)$ donde $f_{out}(i)$ es la expansión del módulo.

La complejidad de datos $D(i)$ de un módulo i se representa con la fórmula $D(i) = \frac{V(i)}{f_{out}(i)+1}$ y se define como la proporción entre el número de variables de entrada y salida del módulo $V(i)$ y su expansión $f_{out}(i)$.

La complejidad del sistema $C(i)$ se calcula sumando las complejidades estructural y de datos $C(i) = S(i) + D(i)$. A medida que disminuyen los valores de complejidad del sistema, su complejidad arquitectónica disminuye también. (Pressman, 2005)

1.6.2.2 Métricas de diseño a nivel de componente

Se basan en las características internas de los componentes de software permitiendo valorar la calidad del diseño en cuanto a cohesión, acoplamiento y complejidad. (Pressman, 2005)

Cohesión Funcional Fuerte.

La siguiente fórmula es empleada para determinar la cohesión funcional fuerte. En la misma $SU(SA(i))$ representa el número de señales de super-unión (el conjunto de señales de datos que se encuentran en todas las porciones de datos de un módulo i:

$$CFF(i) = \frac{SU(SA(i))}{señales(i)}$$

Entre más cercano estén los valores de CFF a 1 mayor será la cohesión del módulo. (Pressman, 2005)

1.6.2.3 Métricas orientadas a clases.

Las características de las clases son la base de un conjunto de métricas de gran utilidad para evaluar la calidad de diseños orientados a objetos. A continuación se explicarán algunas de estas métricas.

Árbol de Profundidad de Herencia (APH).

Esta métrica se define como la longitud máxima desde el nodo hasta la raíz del árbol. A medida que crece el APH la complejidad del módulo aumenta haciéndose más difícil predecir el comportamiento de las clases que ocupan niveles inferiores.

Tamaño de clase (TC).

El tamaño de una clase i está basado en el número de sus operaciones y atributos. Valores elevados de TC, indican que la clase posee demasiada responsabilidad, por lo que disminuye la reusabilidad de la misma y se dificulta su mantenimiento. (Pressman, 2005)

1.7 El Lenguaje Unificado de Modelado (UML).

En la actualidad, pensar en un desarrollo de software sin pasar por una etapa de modelado es prácticamente imposible. La utilización de modelos es una metodología aceptada y recomendada no solo

académicamente, sino también dentro del ámbito profesional. En el presente epígrafe se enunciarán las propiedades más relevantes de UML.

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Algunas de las propiedades de UML que lo han convertido en un lenguaje de modelado estándar son: (Rumbaugh, y otros)

- Concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actuales y futuras.
- Ampliamente utilizado por la industria desde su adopción por OMG.
- Reemplaza a decenas de notaciones empleadas con otros lenguajes.
- Modela estructuras complejas.
- Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario.
- Comportamiento del sistema: casos de uso, diagramas de secuencia y de colaboraciones.

1.8 Herramientas para el desarrollo de software.

Las herramientas CASE (Computer Aided Software Engineering/Ingeniería de Software Asistida por Ordenador) facilitan el desarrollo de software, reduciendo el esfuerzo, el costo y el tiempo. A continuación se hace una caracterización de varias de ellas que permitirá adquirir los elementos necesarios para determinar cuál es la más idónea para especificar y diseñar el módulo Acciones Centralizadas.

1.8.1 Enterprise Architect.

Enterprise Architect es una herramienta CASE para el diseño y construcción de sistemas de software. (Sparx Systems Pty Ltd., 2000-2007)

Principales funcionalidades que ofrece**:** (Sparx Systems Pty Ltd., 2000 - 2010)

- Modelado del ciclo de vida completo para “Sistemas de negocio”, “Ingeniería de software y sistemas” y “Desarrollo en tiempo real y embebido “.

- Trazabilidad completa desde los modelos de requerimientos, análisis y diseño, hasta la implementación y despliegue.
- Herramientas que facilitan la administración de la complejidad. Estas incluyen “Diagramas para modelar conceptos a niveles estratégicos y de negocio”, “Perfiles de dominios específicos y patrones de modelos reusables”, “Administración de líneas base y versiones para rastrear e integrar cambios” y “Seguridad basada en roles”.
- Generación de documentos y herramientas de reporte con un completo editor de plantillas WYSIWYG.
- Generación de código fuente e ingeniería inversa para los lenguajes ActionScript, Ada, C, C++, C#, Java, Delphi, Verilog, PHP, VHDL, Python, System, CVB.Net y Visual Basic.
- Modelado de esquemas de base de datos y generación automática de scripts DDL.

1.8.2 Rational Rose.

Rational Rose es una herramienta CASE basada en UML compatible con la metodología RUP que permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo del software.

Funcionalidades que ofrece: (G.S.I, 2007)

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software".
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.
- Capacidad de análisis de calidad de código.
- El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones Web.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad para integrarse con sistemas de control de versiones.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo.
- Sistemas operativos y plataformas de hardware apropiadas: Windows 2000, Windows NT, Windows XP.

1.8.3 Visual Paradigm

Visual Paradigm es una poderosa herramienta CASE que utiliza UML para el modelado.

Características:

- Licencia: Gratuita y Comercial.
- Producto de calidad.
- Soporta el desarrollo de aplicaciones Web.
- Varios idiomas.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

Funcionalidades: (Visual Paradigm, 1999-2010)

- Soporta un conjunto de estándares entre los que se encuentran UML, SysML, BPMN, XML y XMI.
- Soporte de modelado UML, modelado de procesos de negocios y un generador de mapeo de objetos-relacionales para los lenguajes de programación Java, .NET y PHP.
- Integración con herramientas Java (Eclipse/IBM WebSphere , JBuilder , NetBeans IDE , Oracle JDeveloper y BEA Weblogic).
- Permite la generación de código y la ingeniería inversa para un conjunto de lenguajes entre los que se encuentran Java, C++, CORBA IDL, PHP, XML Schema, Ada y Python. Cualquiera de los cambios en el código existente puede reflejarse en el modelo y viceversa.
- Ofrece herramientas para la generación de reportes en formatos html, pdf y doc.
- Interoperabilidad e integración. Permite la integración con un conjunto de herramientas (Visio drawing, Rational Rose, ERwin Data Modeler Project, Microsoft Excel y Microsoft Word document) e intercambiar diagramas UML y modelos usando representaciones industriales comunes.
- Modelado de base de datos. Proporciona una mayor documentación de la base de datos y diagramas de mapeo de relación de objetos.
- Integración con herramientas para el control de versiones.
- Diseño de prototipo de Interfaz de Usuario. Permite insertar información adicional a los diagramas mediante notas y comentarios para describir sus elementos lo que facilita la revisión de los prototipos así como el trabajo en equipo.

1.9 Herramientas de modelado de Prototipos de interfaz de usuario.

Los prototipos de interfaz de usuario representan una maqueta del producto final que puede utilizarse para verificar los requerimientos del producto en fases tempranas del ciclo de vida del proyecto. En el presente epígrafe se realiza un estudio de las herramientas Axure RP y Microsoft Office Visio 2007 que permiten el diseño de prototipos de interfaz de usuario.

1.9.1 Axure RP.

Axure RP es una aplicación para crear prototipos y especificaciones muy precisas para páginas web. Permite componer la página web visualmente, añadiendo, quitando y modificando los elementos con suma facilidad. Con el uso de anotaciones permite especificar el estado de cada elemento (Propuesto, Aceptado, Incorporado), el beneficio esperado (Crítico, Importante, Útil), el riesgo, la estabilidad, a quién va dirigido y a quién se le asignará la tarea. Integra un módulo de control de versiones que permite distribuir tareas y gestionar proyectos dentro de un equipo. Corre sobre los Sistemas Operativos Win98/98SE/Me/2000/XP. Genera documentos en formato de Microsoft Word. (grupo INTERCOM, 1997-2010)

1.9.2 Microsoft Office Visio 2007.

Microsoft Office Visio 2007 facilita a los profesionales la visualización, el análisis y la comunicación de información compleja.

Características nuevas o mejoradas de Office Visio 2007: (Microsoft Corparation, 2010)

- Guarda diagramas en formato de archivo PDF o XPS para facilitar su portabilidad y llegar a un mayor número de destinatarios.
- Integración con otras aplicaciones de Microsoft Office.
- Vinculación de diagramas a datos proporcionando una visión más completa de lo que se está modelando.
- Permite crear prototipos de interfaz de usuario de gran calidad.
- Proporciona una plantilla “Interfaz de usuario de Windows XP” que permite la creación de prototipos con formas diseñadas según las directrices de Microsoft Windows XP.
- Sistemas operativos apropiados: Microsoft Windows XP y Windows Server 2003.

1.10 Lenguajes de programación web.

La aparición a principios de los noventa del servicio web revolucionó el campo de la informática y las telecomunicaciones propiciando que se fomentaran nuevas formas de negocio. La facilidad para actualizar y mantener aplicaciones web explica el enorme auge que han experimentado en los últimos años así como el surgimiento de un grupo de tecnologías que facilitan la construcción de este tipo de sistemas. A continuación se realiza un estudio de un conjunto de estas tecnologías con el fin de determinar cuáles son las más apropiadas para el desarrollo del módulo Acciones Centralizadas de SINAPSIS.

1.10.1 Tecnologías del lado del cliente.

1.10.1.1 Lenguaje HTML.

El Lenguaje para Marcado de Hipertexto HTML es un lenguaje estático para el desarrollo de sitios web permitiendo estructurar las páginas web, fue desarrollado por el World Wide Web Consortium (W3C) y se ha convertido en un estándar reconocido mundialmente constituyendo el idioma común para todos los navegadores. Los archivos pueden tener las extensiones htm y html.

1.10.1.2 JScript 8.0.

JScript 8.0 es la siguiente generación de la implementación por parte de Microsoft del lenguaje ECMA 262. Entre las mejoras de JScript 8.0, se destacan código compilado real, variables con tipos y sin tipos, enlace en tiempo de compilación y en tiempo de ejecución, clases (con herencia, sobrecarga de funciones, descriptores de acceso de propiedades, etc.), paquetes, compatibilidad entre lenguajes y acceso completo a .NET Framework. (Microsoft Corporation, 2010)

Al igual que las versiones anteriores, JScript 8.0 no es compatible con otro navegador que no sea Microsoft Internet Explorer y otro servidor web que no sea el servidor Web Microsoft Internet Information Server, lo que le resta competitividad con respecto a otros lenguajes.

1.10.1.3 Applets de Java.

Un applet es un programa escrito en el lenguaje de programación Java que puede ser incluido en una página HTML. Cuando una página contiene un applet el código es transferido al sistema y ejecutado por la máquina Virtual de Java (JVM).

Entre las principales ventajas que tienen los applets se encuentran que son menos dependientes del navegador que los scripts en Javascript, son multiplataforma (funcionan en Linux, Windows, Mac OS, y en cualquier sistema operativo para el cual exista una JVM). Java es un lenguaje más potente que JavaScript por lo que el número de aplicaciones de los applets podrá ser mayor. (Oracle Corporation, 2010)

Como desventajas en relación con Javascript se debe señalar que los applets son más lentos de procesar y que tienen espacio muy delimitado en la página donde se ejecutan, es decir, no se mezclan con todos los componentes de la página ni tienen acceso a ellos. Es por ello que con los applets de Java no se pueden hacer directamente cosas como abrir ventanas secundarias, controlar Frames, formularios, capas, etc. (Alvarez, 2002)

1.10.1.4 JavaScript

Javascript es un lenguaje utilizado para realizar acciones dentro del ámbito de una página web. Aunque no es un lenguaje orientado a objetos se pueden implementar muchas de las características de este paradigma y aplicar diversos patrones de código y diseño permitiendo crear aplicaciones web con un alto grado de calidad. Existen un conjunto de librerías JavaScript que facilitan la implementación de aplicaciones con las más diversas funcionalidades.

Entre las características admirables de este lenguaje se destaca su compatibilidad con la mayoría de los navegadores existentes. En cuanto a sus limitaciones, JavaScript fue diseñado de forma que se ejecutara en un entorno muy limitado que permitiera a los usuarios confiar en la ejecución de los scripts. De esta forma, los scripts de JavaScript no pueden comunicarse con recursos que no pertenezcan al mismo dominio desde el que se descargó el script.

1.10.2 Tecnologías del lado del servidor.

1.10.2.1 Páginas Activas en el Servidor.

Las siglas ASP corresponden a las palabras *Active Server Pages* (Páginas Activas en el Servidor). ASP es una tecnología desarrollada por Microsoft para crear páginas web de contenido dinámico apoyándose en scripts ejecutados en el servidor. (Villaverde, 2006-2007)

La ventaja principal de las tecnologías dependientes del servidor radica en la seguridad que tiene el programador sobre su código, ya que éste se encuentra inicialmente en los archivos del servidor que al

ser solicitado a través de la web, es ejecutado, por lo que los usuarios solo tienen acceso a la página resultante en su navegador. Entre sus funciones principales están el acceso a base de datos, envío de correo electrónico, creación dinámica de gráficos y otros.

Es una tecnología propietaria de Microsoft, su uso implica el empleo de productos Microsoft: MS Internet Information System y MS Windows en el servidor, lo que le resta competitividad con respecto a otras tecnologías.

1.10.2.2 Hypertext Preprocessor.

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Permite procesar la información de formularios, generar páginas con contenidos dinámicos, enviar y recibir cookies.

Puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS). Soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTP y muchos otros.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos entre las que se encuentran Adabas D ,dBase , Empress , FilePro (read-only) , Hyperwave , IBM DB2, Informix , Ingres , InterBase , FrontBase , mSQL, Direct MS-SQL, MySQL, ODBC , Oracle (OCI7 and OCI8) , Ovrimos , PostgreSQL ,Solid ,Sybase , Velocis , Unix dbm . (the PHP Documentation Group, 1997-2008). Entre sus desventajas se encuentra el hecho de que no es un lenguaje orientado a objetos.

1.10.2.3 Java Server Pages.

JSP (Java Server Pages) desarrollado por Sun Microsystems, está orientado a desarrollar páginas web en Java. Permite crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis java y son compilados en Servlets por lo que actúan como una puerta a todos los servicios java del servidor y librerías Java para

aplicaciones web. Al usar Java como lenguaje de script incluye las características de Java (alto rendimiento, orientado a objetos, robusto, seguro, portátil).

1.11 Metodologías, lenguajes, herramientas y tecnologías seleccionadas para el desarrollo de los requerimientos y diseño del módulo Acciones Centralizadas.

La metodología seleccionada para el desarrollo es RUP ya que es necesario que todo el proceso de desarrollo esté bien detallado y documentado, esta metodología es recomendable para proyectos donde no existe mucha interacción con los clientes como es el caso de SINAPSIS. Posee un flujo de diseño sólido que facilitará el trabajo de los programadores e implementa algunas de las mejores prácticas recomendadas en la industria y que son necesarias en el proyecto, ya que se va a desarrollar iterativamente, realizándose un control de los cambios y se utilizará la modelación visual como apoyo a todo este proceso. Por otra parte el cliente solicita el uso de tecnologías libres y RUP constituye un marco de trabajo genérico aplicable a cualquier tecnología, además el equipo de trabajo de SINAPSIS ha obtenido buenos resultados anteriormente aplicando esta metodología por lo que posee experiencia en su uso.

Se decidió utilizar UML porque además de ser el lenguaje de modelado propuesto por la metodología seleccionada establece una notación estándar y semánticas esenciales que permitirán que se realice satisfactoriamente la especificación y diseño del módulo. Otro elemento positivo es que el equipo de desarrollo y los clientes están familiarizados con su uso por lo que les será más fácil la comprensión de los diagramas y modelos que se generen.

Se usará JavaScript como lenguaje de programación del lado de cliente por ser compatible con la mayoría de los navegadores incluyendo Internet Explorer y Firefox. Del lado del servidor se empleará el lenguaje de programación orientado a objetos, Java. Su utilización permitirá obtener una solución que se podrá ejecutar en variados sistemas operativos y le proporcionará una mayor seguridad al programa puesto que es un lenguaje compilado que evita el acceso al código fuente por parte de los usuarios. A partir de que se elige el lenguaje Java se asume la tecnología JSP para la creación de páginas web dinámicas.

Se usará la herramienta CASE multiplataforma Visual Paradigm, que permite el modelado usando el lenguaje UML. Dicha herramienta ofrece la generación de código para el lenguaje previamente

seleccionado y facilita el diseño de prototipos de interfaz de usuario, útiles en la validación de los requerimientos. Además se integra con herramientas para el control de versiones favoreciendo el desarrollo en paralelo.

1.11 Conclusiones.

En este capítulo se profundizó en el estudio de las metodologías de desarrollo de software más usadas. Se caracterizaron los patrones de casos de uso y de diseño más utilizados. Se analizaron un conjunto de métricas que validan la calidad de la Especificación de requerimientos y el diseño. Se estudiaron las características fundamentales del lenguaje de modelado UML y de los lenguajes de programación tanto del lado del cliente como del lado del servidor para el desarrollo de aplicaciones web. Se analizaron herramientas que dan soporte al proceso de desarrollo de software.

Después de tener estos elementos se arribó a las siguientes conclusiones:

- Se utilizará la metodología de desarrollo RUP por ser una metodología robusta que genera un gran volumen de información e implementa un conjunto de mejoras prácticas recomendadas para desarrollar software exitosamente.
- De la ingeniería de requerimientos se aplicarán las etapas de elicitación, análisis y validación de requerimientos.
- Se aplicarán los patrones de casos de uso y de diseño necesarios para evitar cometer errores comunes en la especificación y diseño del módulo.
- Se usarán el lenguaje de modelado UML y la herramienta CASE Visual Paradigm para la especificación y el diseño del módulo.
- Los lenguajes de programación que se usarán son JavaScript y Java porque permiten el desarrollo de aplicaciones de gran calidad.

CAPÍTULO 2. SOLUCIÓN PROPUESTA.

2.1 Introducción.

En este capítulo se describe la solución que se propone para la especificación y diseño del módulo Acciones Centralizadas de SINAPSIS. Inicialmente se aplicaron técnicas de la disciplina de Ingeniería de Requerimientos que permitieron definir y especificar las condiciones y capacidades que el sistema debe cumplir. Una vez determinados y especificados los requerimientos se identificaron los actores del sistema, los casos de uso y sus relaciones, estructurándose el diagrama de casos de uso por paquetes. Seguidamente se describieron los casos de uso y se elaboró el Modelo de casos de uso del sistema.

Estos artefactos fueron utilizados para diseñar el módulo, obteniéndose el Modelo de diseño el cual describe la realización física de los casos de usos centrándose en como los requerimientos funcionales y no funcionales tienen impacto en el sistema a considerar. Para su elaboración se identificaron subsistemas, paquetes y clases de diseño, además de distribuir el comportamiento de los casos de uso entre los elementos de diseño identificados.

2.2 Requerimientos de software.

Teniendo como entrada los procesos de negocios definidos para el módulo Acciones Centralizadas y con la realización de entrevistas a los clientes y reuniones donde cada participante expuso sus ideas se determinaron un conjunto de necesidades que debe cumplir el sistema.

Estas necesidades se fueron refinando hasta convertirse en requerimientos técnicos, para ello, se analizó el sistema “Nueva Etapa”, se hicieron reuniones con un enfoque de equipo entre clientes y desarrolladores, además de aplicarse la técnica “El Despliegue de Función de Calidad” que permitió, a través de encuestas y entrevistas realizadas a los clientes, identificar los requerimientos más valiosos para los mismos.

Luego estos requerimientos obtenidos fueron analizados para agruparlos en funcionales y no funcionales. Los funcionales fueron clasificados además según su prioridad en Alta (Esencial), Media (Deseado) o Baja (Opcional), en función de las necesidades de los clientes, y según su costo de desarrollo en Alto, Medio o

Bajo. Los no funcionales a su vez, se clasificaron según las categorías de Usabilidad, Fiabilidad, Eficiencia, Seguridad, Portabilidad, Reusabilidad y Capacidad (Consumo de recursos).

A continuación se muestran los requerimientos funcionales y no funcionales más importantes del sistema, para acceder a la descripción íntegra de los mismos ver los documentos Especificación de requerimientos funcionales del sistema y Especificación de requerimientos no funcionales del sistema respectivamente.

2.2.1 Requerimientos funcionales.

RF_AC.1 Obtener acción centralizada a elaborar.

El sistema permitirá obtener una acción centralizada con los siguientes datos:

- Código
- Denominación
- Descripción
- Montos imputados (Bs.F)
- Estado
- Detalles de las acciones específicas asociadas

Prioridad: Alta

Costo de desarrollo: Bajo

RF_AC.2 Mostrar detalles de acción centralizada a elaborar.

El sistema permitirá mostrar los siguientes detalles de la acción centralizada:

- Código
- Denominación
- Descripción
- Montos imputados (Bs.F)
- Estado
- Detalles de las acciones específicas imputadas

Prioridad: Alta

Costo de desarrollo: Bajo

RF_AC.3 Obtener acción específica.

El sistema permitirá obtener los siguientes detalles de la acción específica:

- Enunciado
- Descripción
- Plazo de ejecución
- Partidas presupuestarias con sus montos imputados
- Fuentes de Financiamientos con sus montos
- Monto total

Prioridad: Alta

Costo de desarrollo: Bajo

RF_AC.4 Mostrar detalles de acción específica.

El sistema permitirá mostrar los siguientes detalles de la acción específica:

- Enunciado
- Descripción
- Plazo de ejecución
- Partidas presupuestarias con sus montos imputados
- Fuentes de Financiamientos con sus montos
- Monto total

Prioridad: Alta

Costo de desarrollo: Bajo

RF_AC.5 Imputar acción específica.

El sistema permitirá imputar una acción específica a partir de los siguientes datos:

- Fecha inicio
- Fecha de fin
- Monto por cada partida presupuestaria
- Monto por cada fuente de financiamiento

Prioridad: Alta

Costo de desarrollo: Bajo

2.2.2 Requerimientos no funcionales.

Usabilidad

RNF_AC.3 Mostrar los mensajes, títulos y demás textos que aparezcan en la interfaz del sistema en idioma español.

Tanto los títulos de los componentes de la interfaz, como los mensajes para interactuar con los usuarios, así como los mensajes de error, deberán ser en idioma español y tener una apariencia uniforme en todo el sistema. Los mensajes de error deberán ser lo suficientemente informativos para dar a conocer la severidad del error.

Seguridad

RNF_AC.14 Restringir el acceso al sistema.

Para acceder al sistema cada usuario debe ser autenticado, de manera que se pueda determinar si tiene acceso al mismo y en caso de ser así determinar las funcionalidades a las que tiene acceso y restringir su actividad al uso de las mismas las cuales están determinadas por los roles que posee el usuario. Para autenticarse el usuario debe indicar: nombre de usuario y contraseña.

Portabilidad

RNF_AC.16 El sistema debe ser una aplicación Web.

El sistema se debe ejecutar sobre un entorno web de manera que se permita su acceso desde cualquier punto del país utilizando la red.

RNF_AC.17 Garantizar compatibilidad con navegadores de uso común.

El sistema deberá ser compatible con los navegadores:

• Microsoft Internet Explorer 6.0 o superior.

• Mozilla Firefox 2.0 o superior.

Reusabilidad

RNF_AC.20 Definir un modelo en capas para el sistema.

El sistema deberá considerar en su arquitectura un modelo n capas, donde se definen n componentes lógicos de manera independiente.

2.3 Actores del sistema.

Una vez definidos los requerimientos se identificaron los actores del sistema quedando definido el entorno externo del mismo.

Actor	Descripción
Planificador	El actor Planificador es el encargado de gestionar los datos de las acciones centralizadas de su organismo, es decir, es el responsable de elaborar los datos de las acciones centralizadas, imputando cada una de las acciones específicas asociadas a estas y enviarlas a revisión. También el actor Planificador es el encargado de gestionar las solicitudes de nuevas acciones centralizadas, es decir, es el responsable de adicionar nuevas solicitudes, modificarlas, eliminarlas y enviarlas a revisión.
Administrador ONAPRE	El actor Administrador ONAPRE es el encargado de gestionar las acciones centralizadas, es decir, es el responsable de adicionar nuevas acciones centralizadas, modificarlas, eliminarlas y habilitarlas a los organismos.
Revisor	El actor Revisor es el encargado de aceptar las acciones centralizadas y las solicitudes de nuevas acciones centralizadas, es decir, es el responsable de aceptarlas, rechazarlas y enviarlas a revisión.

Tabla 1 Descripción de los actores del sistema.

2.4 Casos de uso.

Cada forma en que los actores usan el sistema constituye un caso de uso (CU). Los mismos son “fragmentos” de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. La identificación de los CU se realizó a partir de la Especificación de requerimientos, aplicándose patrones de CU como CRUD para lograr un mejor resultado. A continuación se muestra uno de los casos de uso más significativos del módulo, para acceder a la descripción íntegra de todos los CU consultar el documento Modelo de casos de uso del sistema.

2.4.1 Gestionar acción centralizada a elaborar.

Caso de Uso:	Gestionar acción centralizada a elaborar

Actores:	Planificador
Resumen:	El caso de uso se inicia cuando el Planificador necesita gestionar una acción centralizada a elaborar. Consiste en que el Planificador selecciona la opción de elaborar una acción centralizada y luego la opción de imputar una acción específica y llena los campos requeridos para la imputación. También tiene la posibilidad de seleccionar una acción centralizada ya existente para ver sus detalles o enviarla a revisión. El caso de uso termina con la elaboración o envío a revisión de una acción centralizada.
Referencias	RF_AC.1, RF_AC.2, RF_AC.3, RF_AC.4, RF_AC.5, RF_AC.6, RF_AC.7, RF_AC.8, RF_AC.9, RF_AC.10, RF_AC.11, RF_AC.12, RF_AC.13, RF_AC.14, RF_AC.15

Tabla 2 CU: Gestionar acción centralizada a elaborar.

2.5 Diagrama de casos de uso del sistema

Una vez definidos los actores, los casos de uso del sistema y sus relaciones, se elaboró el diagrama de casos de uso del sistema el cual fue estructurado en paquetes lo que permitió agrupar los CU que dan soporte a un determinado actor.

2.5.1 Diagrama de casos de uso del paquete "Administración de Acciones Centralizadas".

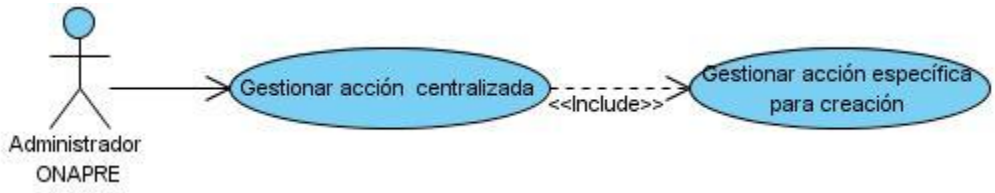


Figura 2 Diagrama de casos de uso del paquete "Administración de Acciones Centralizadas".

2.5.2 Diagrama de casos de uso del paquete "Revisión de Acciones Centralizadas y Solicitudes de nuevas AC".

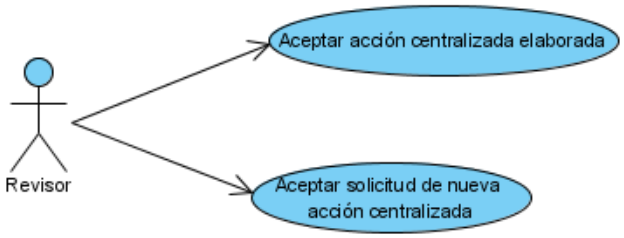


Figura 3 Diagrama de casos de uso del paquete "Revisión de Acciones Centralizadas y Solicitudes de nuevas AC".

2.5.3 Diagrama de casos de uso del paquete "Planificación de Acciones Centralizadas y Solicitudes de nuevas AC".

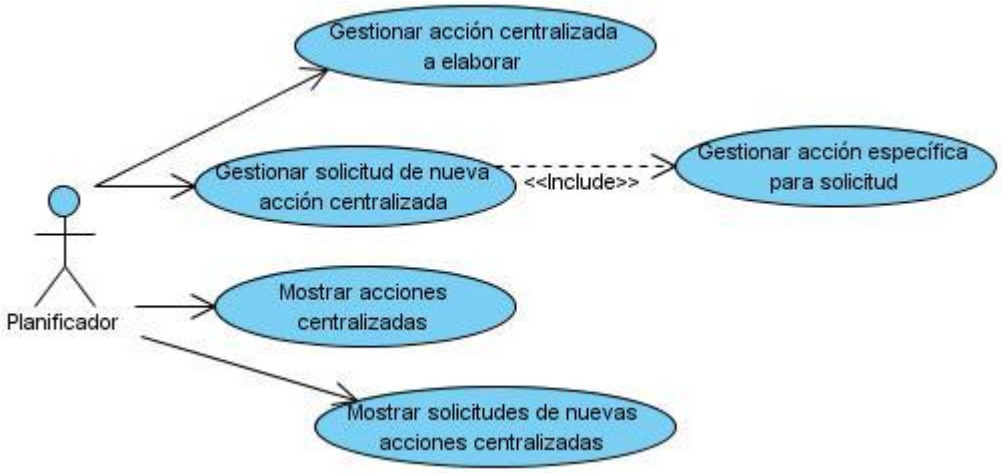


Figura 4 Diagrama de casos de uso del paquete "Planificación de Acciones Centralizadas y Solicitudes de nuevas AC".

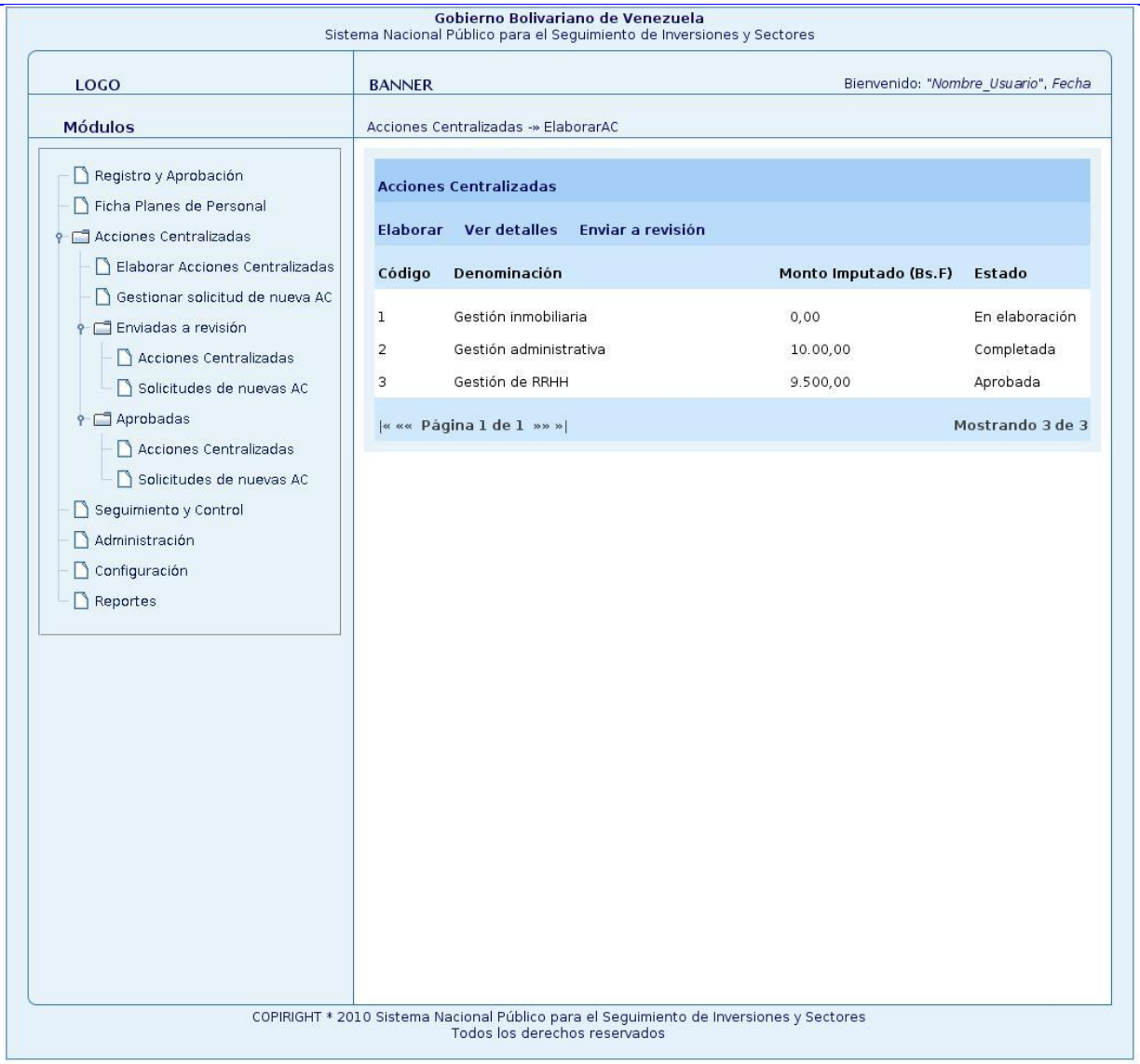
2.6 Descripción de casos de uso del sistema.

Luego de definir los CU del sistema se procede a especificarlos, para ello se describen los diferentes flujos de sucesos entre el actor y el sistema incluyéndose prototipos de interfaz de usuario no funcionales que permiten que se logre un mayor entendimiento entre clientes y desarrolladores. La descripción de los casos de uso constituye una guía para los desarrolladores y un documento de obligatorio cumplimiento. A continuación se muestra un fragmento de la descripción de uno de los CU más importantes del módulo. Para acceder a la descripción íntegra de todos los CU, ver el documento Modelo de casos de uso del sistema.

2.6.1 Gestionar acción centralizada a elaborar.

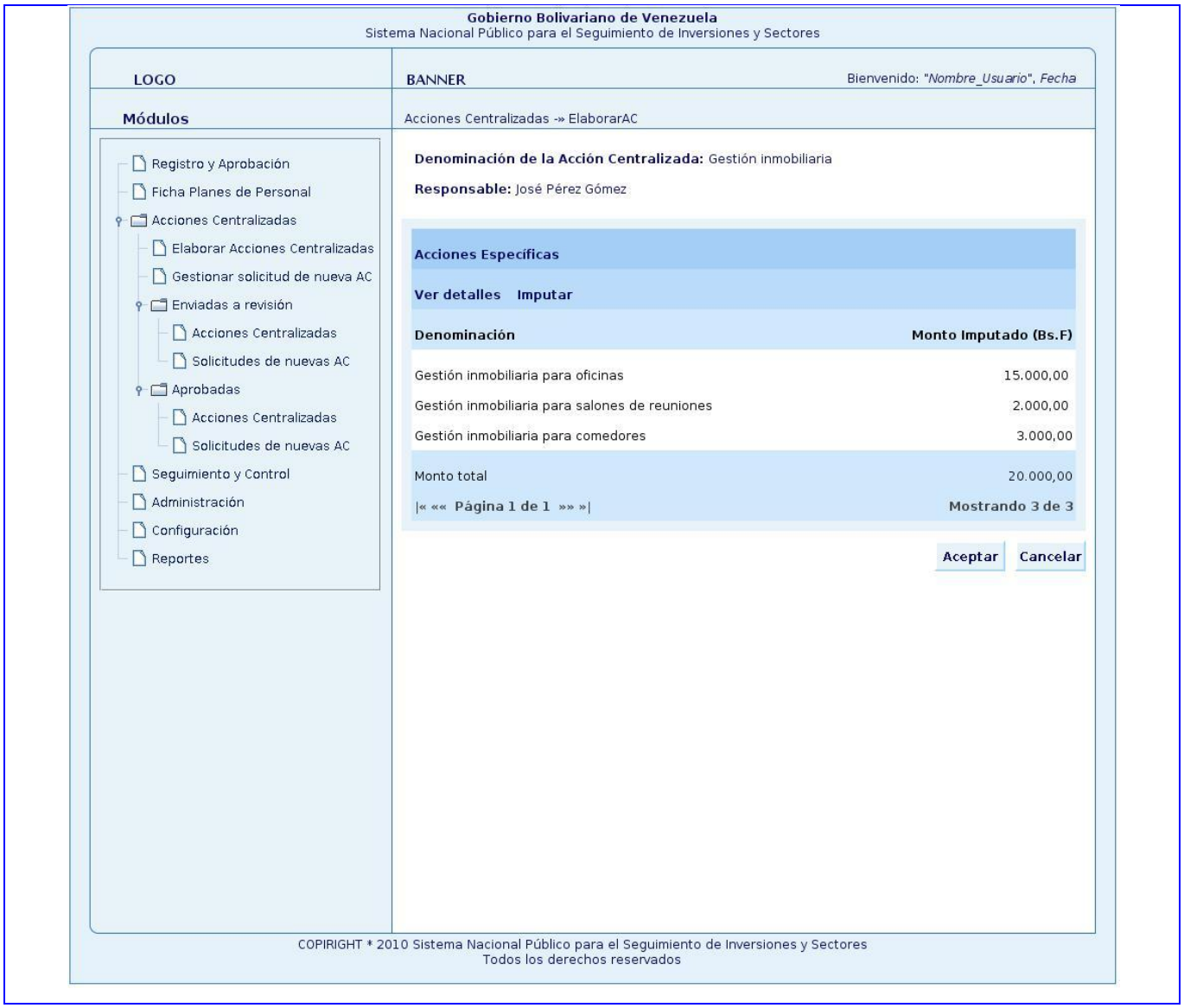
Caso de Uso:	Gestionar acción centralizada a elaborar
Actores:	Planificador
Resumen:	El caso de uso se inicia cuando el Planificador necesita gestionar una acción centralizada a elaborar. Consiste en que el Planificador selecciona la opción de elaborar una acción centralizada y luego la opción de imputar una acción específica y llena los campos requeridos para la imputación. También tiene la posibilidad de seleccionar una acción centralizada ya existente para ver sus detalles o enviarla revisión. El caso de uso termina con la elaboración o envío a revisión de una acción centralizada.
Precondiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El usuario debe estar autenticado con los permisos necesarios.
Referencias	RF_AC.1, RF_AC.2, RF_AC.3, RF_AC.4, RF_AC.5, RF_AC.6, RF_AC.7, RF_AC.8, RF_AC.9, RF_AC.10, RF_AC.11, RF_AC.12, RF_AC.13, RF_AC.14, RF_AC.15
Reglas del Negocio	• La sumatoria de la distribución de financiamiento de las acciones específicas tiene que ser igual al monto definido en la acción Centralizada correspondiente. • La sumatoria de la distribución por fuente de financiamiento de las acciones específicas tiene que corresponderse con el monto definido para cada fuente de financiamiento. • El plazo de ejecución de una acción Centralizada está determinado por el plazo mayor de las acciones específicas.
Prioridad	Crítico
Complejidad	Complejo
Nivel del caso de uso	Usuario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

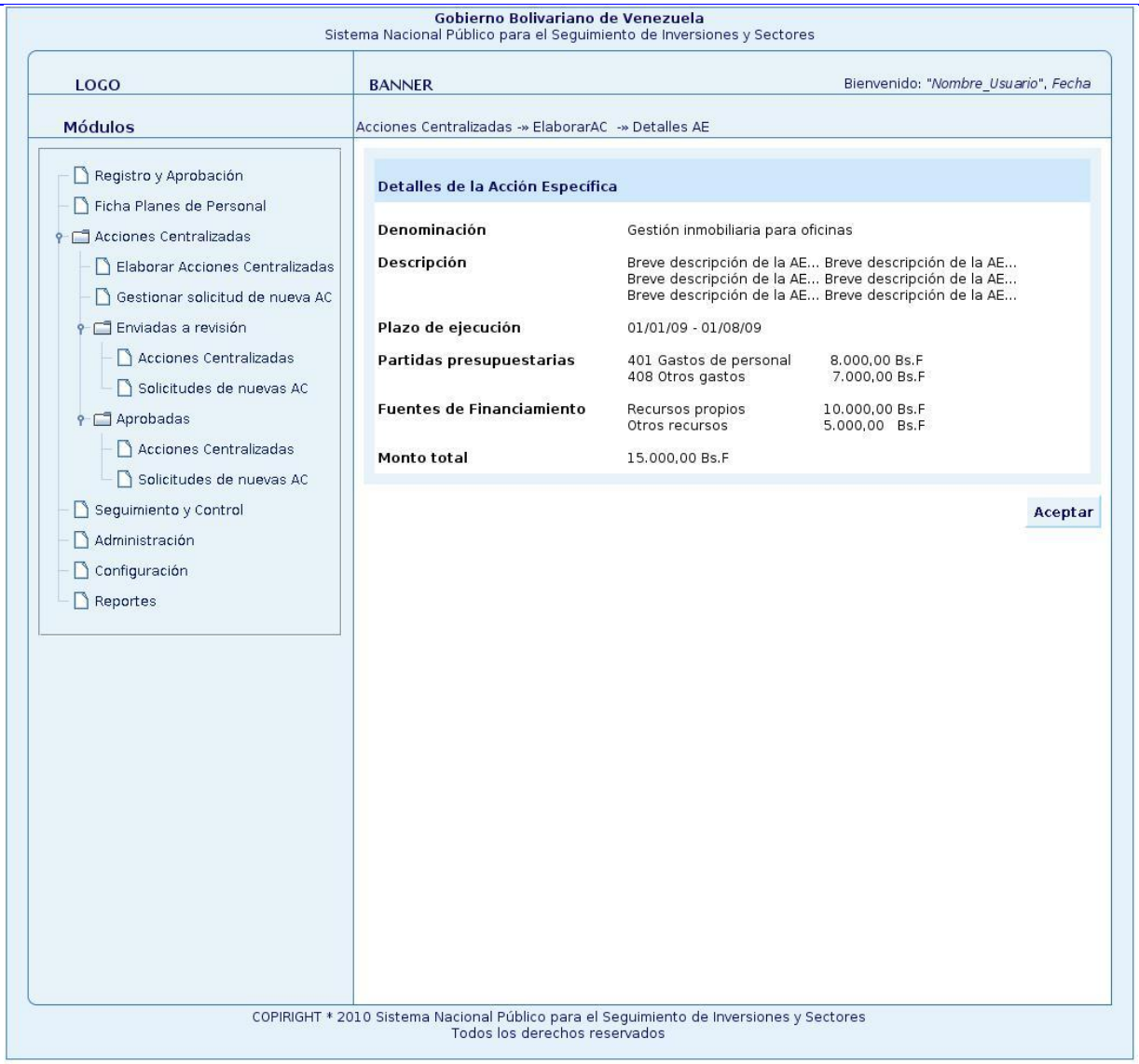
1. El caso de uso inicia cuando el actor Planificador selecciona la opción “Acciones Centralizadas” y dentro de esta la opción “Elaborar acciones centralizadas”.	2. El sistema muestra una interfaz donde se lista el conjunto de acciones centralizadas que tiene habilitadas su organismo. De las mismas se muestra: • Código • Denominación • Monto Imputado (Bs.F) • Estado
3. El actor Planificador selecciona una de las siguientes opciones: • Elaborar (una vez que haya seleccionado la acción centralizada que desea elaborar, ver sección **“Elaborar AC”**) • Ver detalles (una vez que haya seleccionado la acción centralizada de la que desea ver sus detalles, ver sección **“Ver detalles AC”**) • Enviar a revisión (una vez que haya seleccionado la acción centralizada que desea enviar a revisión, ver sección **“Enviar a revisión”**)	
Prototipo de Interfaz	



Sección "Elaborar AC"	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra una interfaz donde se lista el conjunto de acciones específicas de la acción centralizada seleccionada, además de los siguientes datos de la acción centralizada:

	• Denominación de acción centralizada • Responsable
2. El actor Planificador selecciona la opción “Ver detalles” una vez que haya seleccionado una acción específica.	3. El sistema muestra una interfaz donde se muestran los siguientes detalles de la acción específica: • Denominación • Descripción • Plazo de ejecución • Partidas presupuestarias con sus montos imputados • Fuentes de Financiamientos con sus montos • Monto total
4. El actor Planificador selecciona la opción “Aceptar”.	5. El sistema muestra la interfaz del paso 1 del flujo normal de eventos de esta sección.
6. El actor Planificador selecciona la opción “Aceptar”.	7. El sistema regresa al paso 2 del flujo normal de eventos, terminando así el caso de uso.
Prototipo de Interfaz	

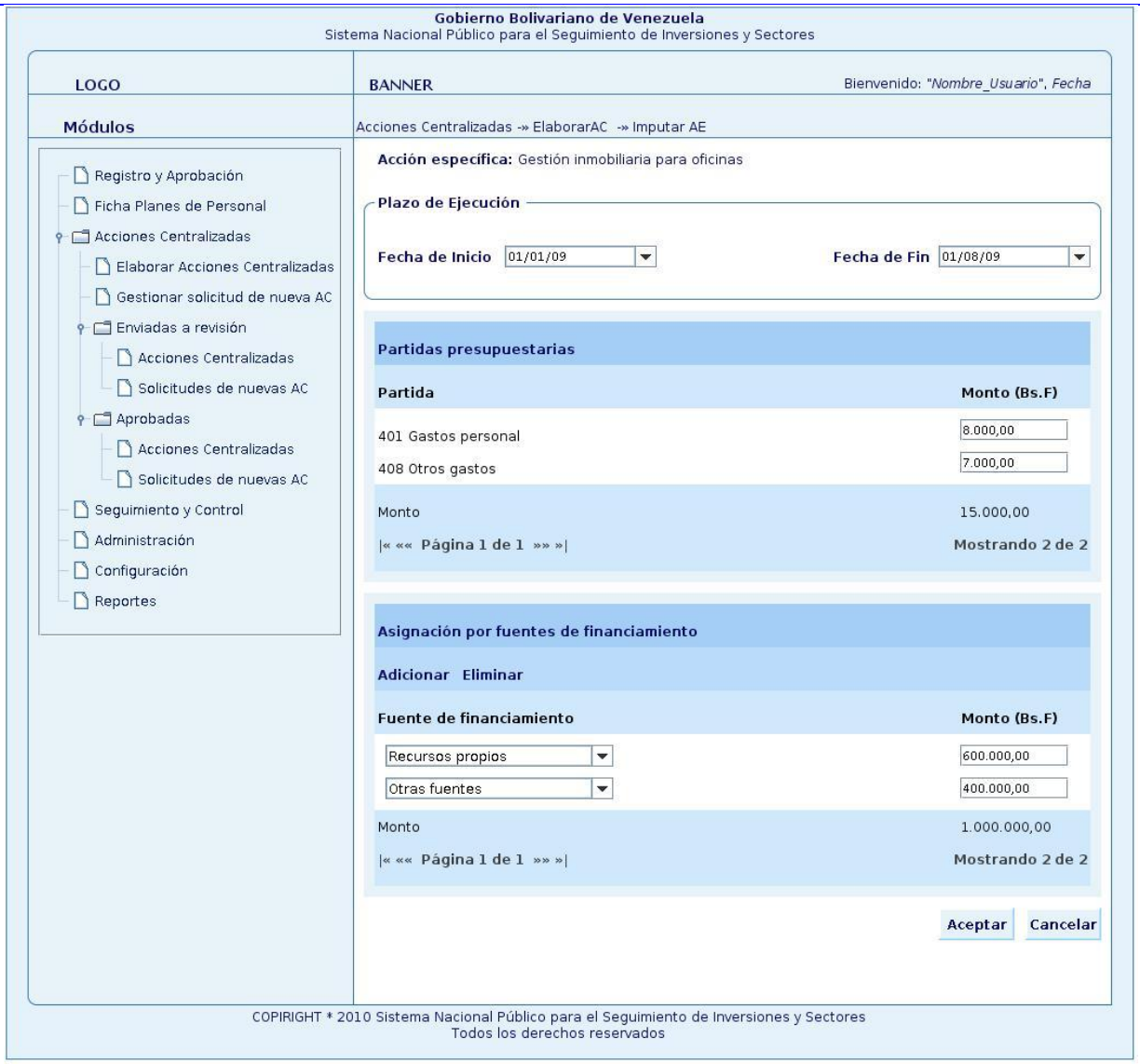

Gobierno Bolivariano de Venezuela
Sistema Nacional Público para el Seguimiento de Inversiones y Sectores
LOGO
BANNER
Bienvenido: "Nombre_Usuario", Fecha
Módulos
Acciones Centralizadas -» ElaborarAC
Registro y Aprobación
Ficha Planes de Personal
Acciones Centralizadas
Elaborar Acciones Centralizadas
Gestionar solicitud de nueva AC
Enviadas a revisión
Acciones Centralizadas
Solicitudes de nuevas AC
Aprobadas
Acciones Centralizadas
Solicitudes de nuevas AC
Seguimiento y Control
Administración
Configuración
Reportes
Denominación de la Acción Centralizada: Gestión inmobiliaria
Responsable: José Pérez Gómez
Acciones Específicas
Ver detalles Imputar
Denominación
Monto Imputado (Bs.F)
Gestión inmobiliaria para oficinas 15.000,00
Gestión inmobiliaria para salones de reuniones 2.000,00
Gestión inmobiliaria para comedores 3.000,00
Monto total 20.000,00
|« «« Página 1 de 1 »» »| Mostrando 3 de 3
Aceptar
Cancelar
COPIRIGHT * 2010 Sistema Nacional Público para el Seguimiento de Inversiones y Sectores
Todos los derechos reservados



Flujo alterno al paso 2 "AE a imputar"

Acción del Actor	Respuesta del Sistema
2.a El actor Planificador selecciona la opción "Imputar" una vez que haya seleccionado una acción específica.	2.b El sistema muestra la interfaz para imputar una acción específica, solicitando los siguientes datos: • Fecha de inicio • Fecha de fin

	• Monto por cada partida presupuestaria • Monto por cada fuente de financiamiento
2.c El actor Planificador introduce los datos correspondientes y selecciona la opción “Aceptar”.	2.d El sistema valida que los datos introducidos son correctos y que no hay campos obligatorios vacíos.
	2.e El sistema imputa la acción específica y actualiza el estado de la acción centralizada.
	2.f El sistema muestra la interfaz del paso 1 del flujo normal de eventos de esta sección.
2.g El actor Planificador selecciona la opción “Aceptar”.	2.h El sistema regresa al paso 2 del flujo normal de eventos, terminando así el caso de uso.
Prototipo de Interfaz	



Pos-condiciones	• El sistema queda con una acción centralizada elaborada (con sus acciones específicas imputadas). • El sistema queda con una acción centralizada con estado “En revisión”.

Tabla 3 Especificación del CU Gestionar acción centralizada a elaborar.

2.7 Subsistema de diseño.

Los subsistemas de diseño son una forma de organizar los artefactos del Modelo de diseño en piezas más manejables. Los elementos del subsistema deben tener un alto grado de cohesión. El subsistema en su conjunto debe tener bajo acoplamiento con respecto a otros subsistemas, lo cual facilita su reutilización, pudiéndose convertir en componentes genéricos.

Para ganar en reusabilidad y lograr que el mantenimiento fuese menos costoso por cada subsistema se definió una interfaz, de esta forma un subsistema no depende específicamente de otro, sino de los servicios brindados, o sea la interfaz, con lo cual el subsistema podría ser reemplazado por otro que cumpla con los servicios descritos sin que el subsistema cliente se vea afectado.

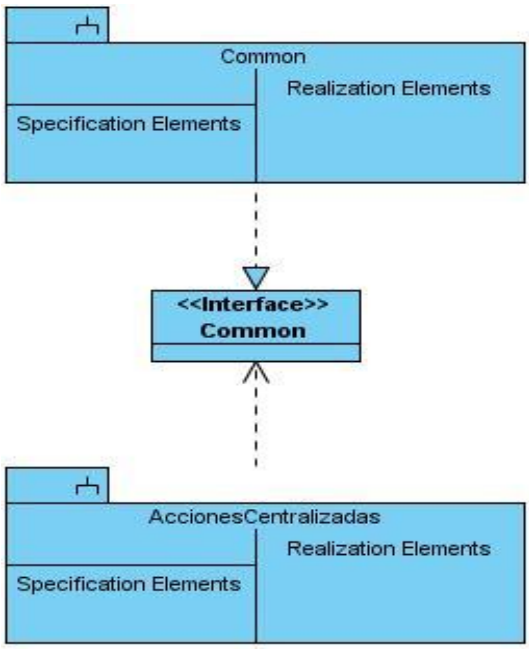


Figura 5 Subsistema de diseño del módulo Acciones Centralizadas de SINAPSIS.

2.8 Paquetes de diseño.

Para organizar el subsistema de diseño Acciones Centralizadas se definieron siete paquetes de diseño. A continuación se describe el contenido de cada uno de ellos.

Interfaz: agrupa las clases interfaz.

Web: agrupa las clases que manejan las peticiones del usuario.

Facade: agrupa las clases que implementan el patrón fachada.

Bussines: agrupa las clases de lógica de negocio.

Dao: agrupa las clases de acceso a datos.

Vo: agrupa las clases persistentes.

Util: agrupa las clases auxiliares.

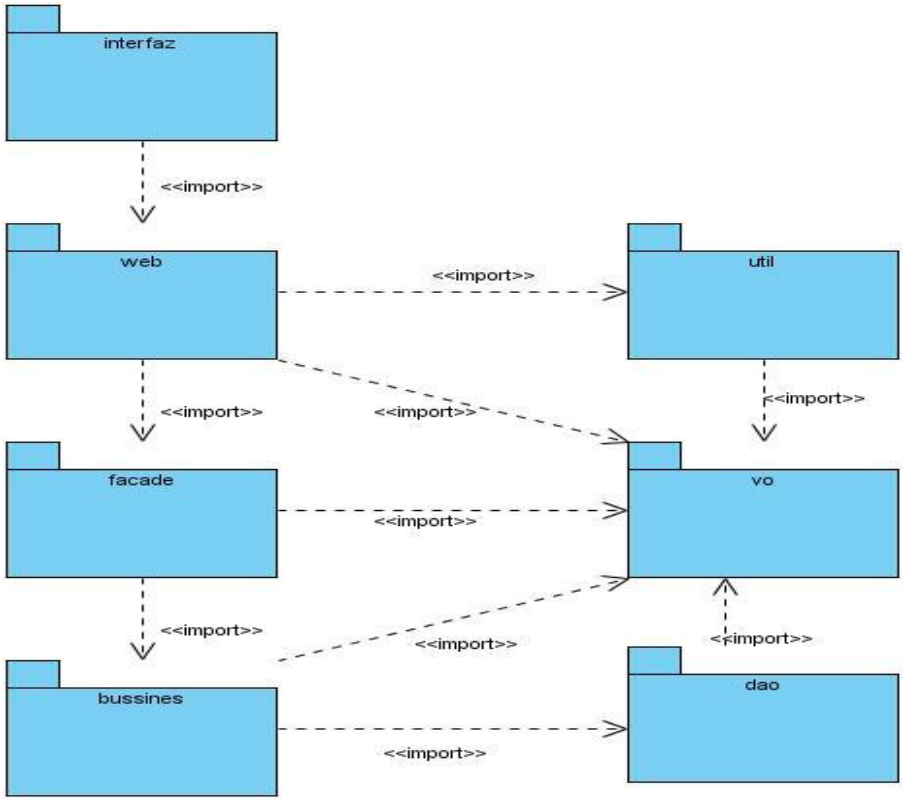


Figura 6 Paquetes de diseño del módulo Acciones Centralizadas de SINAPSIS.

2.9 Diagramas de clases del diseño.

Para modelar la vista de diseño estático del módulo se confeccionaron los diagramas de clases del diseño. Los mismos muestran las relaciones entre clases, interfaces así como la colaboración entre ellos. Los diagramas de clases, son importantes, no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

Entre los requerimientos no funcionales de SINAPSIS se encuentra que debe ser una aplicación web, por lo que para la elaboración de los diagramas de clases se hizo uso de estereotipos web lo que posibilitó

que estos diagramas alcanzaran el nivel de abstracción adecuado. Se tuvieron en cuenta los patrones de diseño aplicándose el patrón de diseño Fachada para integrar las clases controladoras con las de lógica de negocio. A continuación se muestra el diagrama de clases para el CU Gestionar acción centralizada a elaborar. Para consultar el resto ver el documento Modelo de diseño.

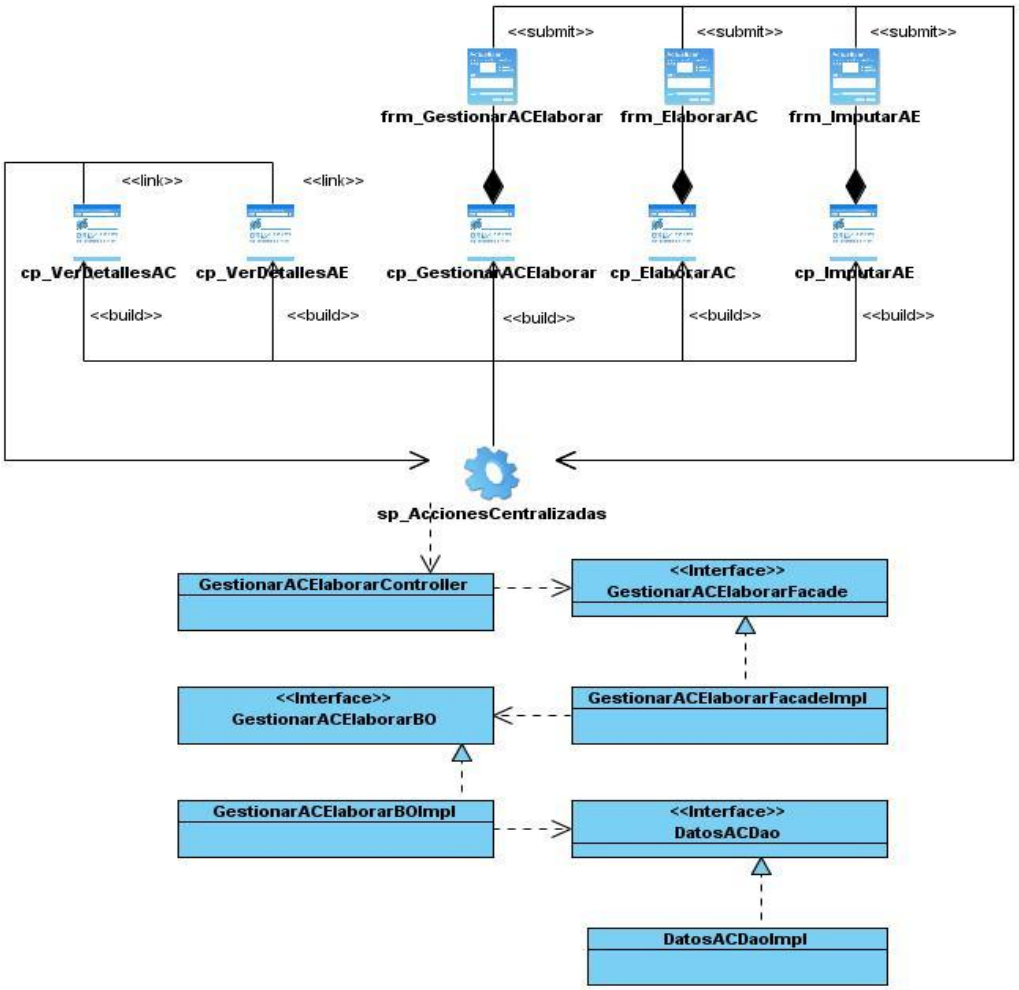


Figura 7 Diagrama de clases de diseño: CU Gestionar acción centralizada a elaborar.

2.10 Diagramas de secuencia.

Para lograr un mejor entendimiento de cómo son realizados y ejecutados los CU en términos de clases y objetos del diseño se elaboraron los diagramas de secuencia. A los casos de uso que tienen varias

secciones se les realizó un diagrama de secuencia por cada flujo para un mejor entendimiento de los mismos.

A continuación se muestra el diagrama de secuencia para la sección más importante del CU Gestionar acción centralizada a elaborar, el resto puede ser consultado en el documento Modelo de diseño.

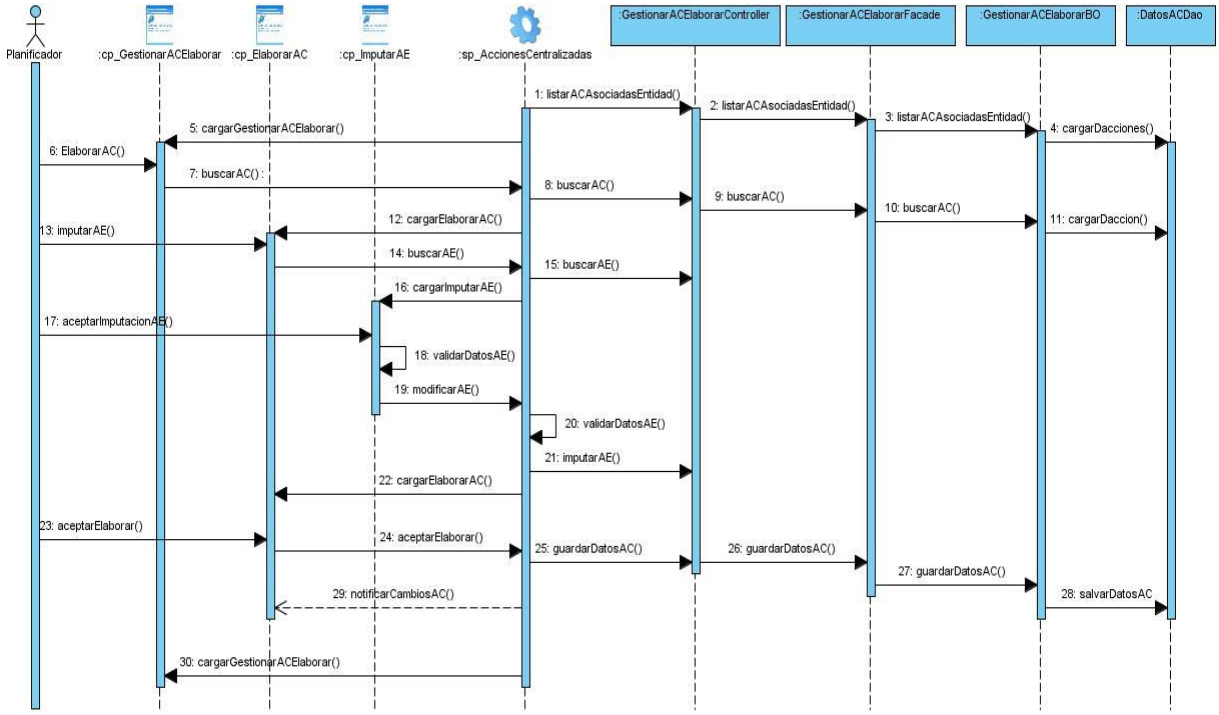


Figura 8 Diagrama de secuencia: CU Gestionar acción centralizada a elaborar, sección “Elaborar AC”, flujo alterno al paso 2 “AE a imputar”.

2.11 Modelo de datos.

El modelo de datos describe el comportamiento lógico y físico de los elementos persistentes de utilidad para dar soporte de información al sistema. Con el fin de garantizar la persistencia de los datos se modeló y normalizó el modelo de entidad - relación del módulo Acciones Centralizadas de SINAPSIS, el cual se muestra a continuación.

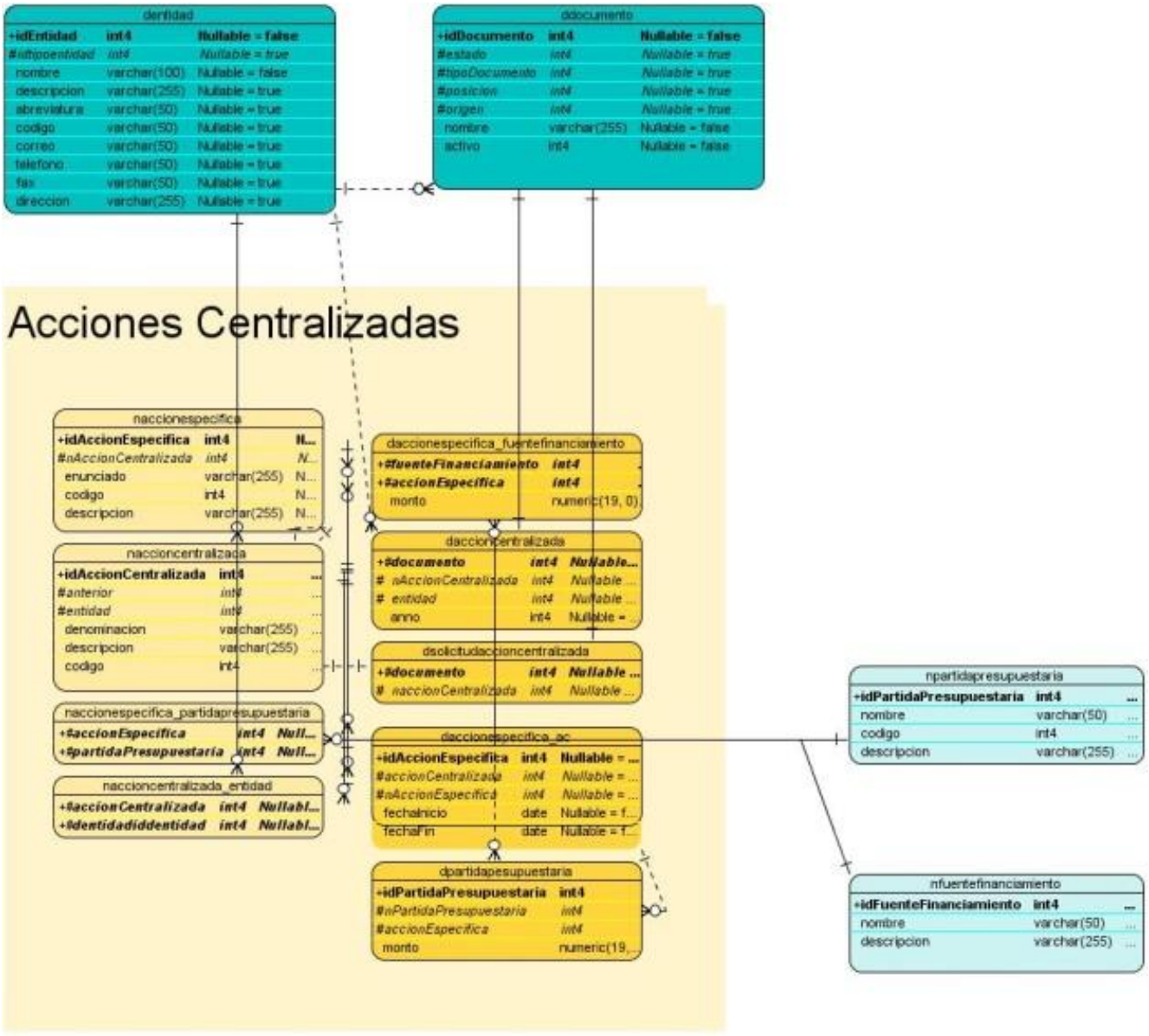


Figura 9 Modelo de datos para el módulo Acciones Centralizadas de SINAPSIS.

2.12 Diagrama de clases persistentes.

Para facilitar la persistencia de los datos se generó el diagrama de clases persistentes a partir del modelo de datos. Este diagrama representa la relación entre las clases persistentes incluyendo dependencias y cardinalidades. Con el uso de Hibernate los objetos persistentes son mapeados con las tablas de la base

de datos. La sincronización del modelo de entidad - relación con este diagrama quedó de la siguiente forma.

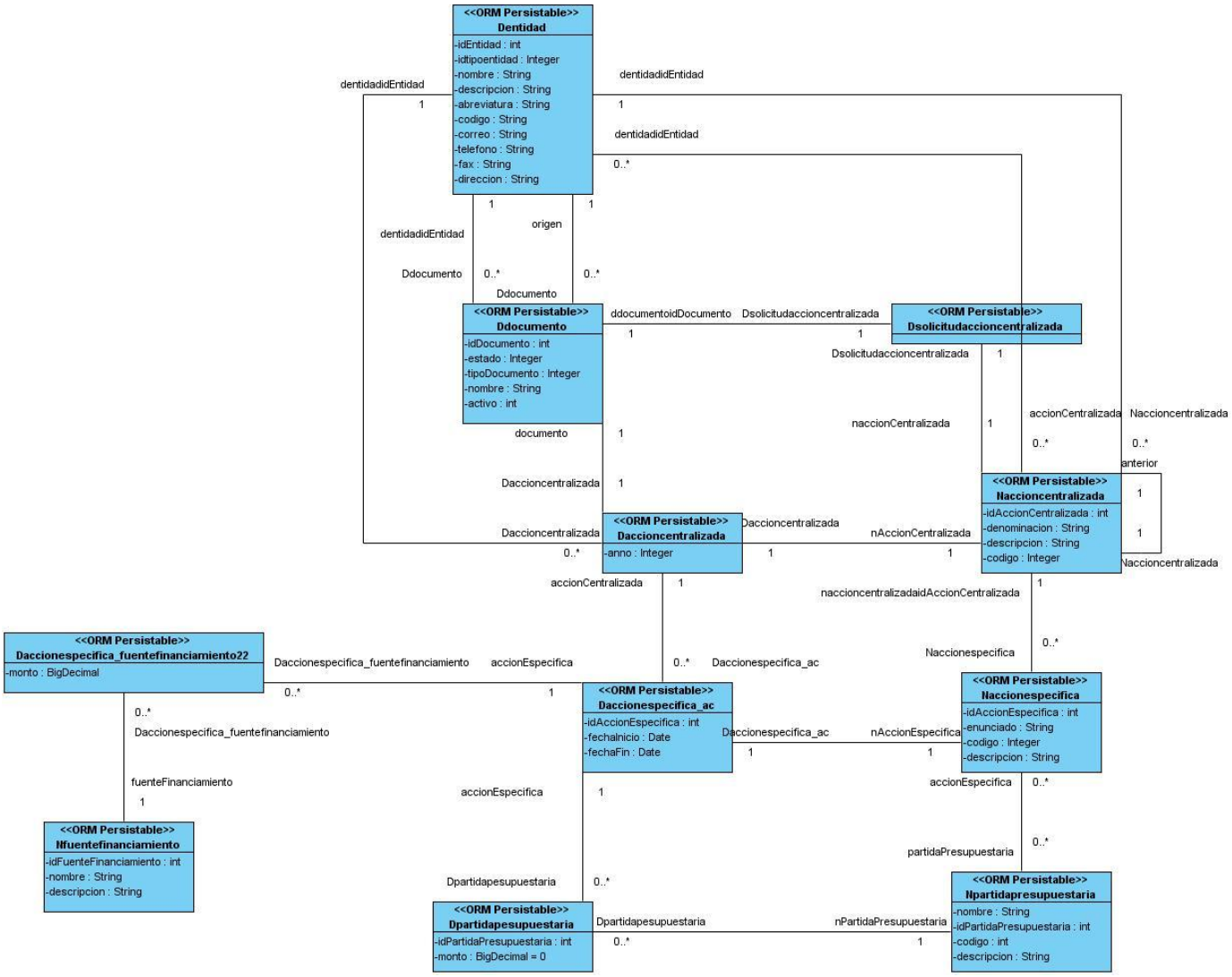


Figura 10 Diagrama de clases persistentes para el módulo Acciones Centralizadas de SINAPSIS.

2.13 Conclusiones.

En este capítulo se analizaron las particularidades referentes al sistema SINAPSIS específicamente del módulo Acciones Centralizadas, arribándose a las siguientes conclusiones:

- La aplicación de técnicas para la identificación y análisis de requerimientos posibilitó la obtención de resultados satisfactorios en la especificación del sistema.

- La elaboración del Modelo de casos de uso del sistema facilitó un mayor entendimiento y un acuerdo común entre los clientes y los desarrolladores, en cuanto a la concepción de las funcionalidades que el sistema debe cumplir.
- La modelación de los artefactos producidos utilizando Visual Paradigm y UML como lenguaje de modelado contribuyó a alcanzar una mayor claridad de los mismos, propiciando un mayor entendimiento entre los involucrados en el módulo Acciones Centralizadas de SINAPSIS.
- La construcción del Modelo de diseño permitió describir la realización física de los CU constituyendo la entrada fundamental para las actividades de implementación.
- La aplicación de patrones, tanto de casos de uso como de diseño influyó positivamente en la calidad de los artefactos generados.

CAPÍTULO 3. ANÁLISIS DE LOS RESULTADOS.

3.1 Introducción.

La medición permite tener una visión más profunda de los productos de ingeniería que se crean. En este capítulo se describe el análisis de los resultados obtenidos, mostrándose cómo fueron aplicados un conjunto de procedimientos y métodos que permitieron valorar objetivamente la calidad de la Especificación de requerimientos del sistema, del Modelo de casos de uso del sistema y del Modelo de diseño, demostrándose que los mismos poseen la calidad requerida.

3.2 Validación de la Especificación de requerimientos.

Para la validación de la Especificación de requerimientos y del Modelo de casos de uso del sistema fueron aplicadas listas de chequeos y métricas que permitieron verificar que estos artefactos cumplen con un conjunto de características imprescindibles que garantizan su calidad.

3.2.1 Lista de chequeo para la Especificación de requerimientos.

Para la verificación de la Especificación de requerimientos se aplicó una lista de chequeo. Las preguntas contenidas en la misma se agruparon bajo una categoría relativa a una característica de calidad. En cada una de las revisiones se fueron arreglando las no conformidades hasta que la Especificación de requerimientos satisfizo todas las características de calidad contenidas en la lista.

3.2.2 Métricas para la Especificación de requerimientos.

Para complementar la validación de la Especificación de requerimientos mediante la aplicación de las listas de chequeo, se aplicaron además las métricas propuestas por Davis y sus colegas, a continuación se muestran los resultados obtenidos.

Número de requerimientos en la especificación: $n_r = n_f + n_{nf}$

n_r : Total de requerimientos en una especificación.

n_f : Número de requerimientos funcionales.

n_{nf} : Número de requerimientos no funcionales.

$n_r = 43 + 21 = 64$

Especificidad: $Q_1 = n_{ui} / n_r$

Q_1 : Especificidad de los requerimientos.

n_{ui} : Número de requerimientos para los que todos los revisores tuvieron interpretaciones idénticas.

$$Q_1 = \frac{64}{64} = 1.00$$

Compleción: $Q_2 = \frac{n_A}{n_A + n_B}$

Q_2 : Compleción de los requerimientos.

n_A : Número de requerimientos funcionales completos.

n_B : Número de requerimientos pobremente especificados.

$$Q_2 = \frac{43}{43} = 1.00$$

Corrección: $Q_3 = \frac{n_c}{n_c + n_{nv}}$

Q_3 : Corrección de los requerimientos.

n_c : Número de requerimientos que se han validado como correctos.

n_{nv} : Número de requerimientos que no se han validado como correctos todavía.

$$Q_3 = \frac{64}{64 + 0}$$

$$Q_3 = \frac{64}{64} = 1.00$$

Comprensión: $Q_4 = \frac{n_{cu}}{n_r}$

Q_4 : Comprensión de los requerimientos.

n_{cu} : Número de requerimientos que todos los revisores entienden.

$$Q_4 = \frac{64}{64} = 1.00$$

Capacidad de verificación: $Q_5 = \frac{n_r}{n_r + \sum_i c(r_i) + \sum_i t(r_i)}$

Q_5 : Capacidad de verificación.

$\sum_i c(r_i)$: Costo para verificar la presencia del requerimiento i.

$\sum_i t(r_i)$: Tiempo para verificar la presencia del requerimiento i.

$$Q_5 = \frac{64}{64 + 6.4 + +8.5} = 0.81$$

Consistencia interna: $Q_6 = \frac{n_u - n_n}{n_u}$

Q_6 : Consistencia interna.

n_u : Número de requerimientos especificados.

n_n : Número de requerimientos en conflicto con otros requerimientos en la especificación.

$Q_6 = \frac{64 - 0}{64}$ = 1.00

Consistencia externa: $Q_7 = \frac{n_{EC}}{n_r}$

Q_7 : Consistencia externa.

n_{EC} : Número de requerimientos que son consistentes con otros documentos.

$Q_7 = \frac{64}{64}$ = 1.00

No redundancia: $Q_{18} = \frac{n_f}{n_u}$

n_f : Número de requerimientos funcionales.

n_u : Número de requerimientos funcionales únicos.

$Q_{18} = \frac{43}{43}$ = 1.00

Al observar los resultados obtenidos de la aplicación de las métricas propuestas por Davis y sus colegas, ver Figura 18, se manifiestan un conjunto de características en la Especificación de requerimientos que demuestran su alto nivel de calidad lo que influye positivamente en que se construya el sistema correcto.

Entre los atributos a destacar está la ausencia de ambigüedad lo que garantiza que cada uno de los requerimientos sea interpretado correctamente. Además se puede observar el alto grado de compleción, este valor se debe fundamentalmente a que todos los requerimientos que el software debe cumplir han sido incluidos y especificados.

Dos factores críticos en la calidad de la especificación son que cada requisito contenido en la misma represente una capacidad o cualidad que debe estar presente en el sistema a construir y que todos los

interesados puedan comprenderlos con un mínimo de explicación. Los resultados demuestran que esto se ha logrado en cada uno de los requerimientos.

Otro de los resultados relevantes que arrojan las métricas aplicadas a la Especificación de requerimientos es que la capacidad de verificación de los mismos es aceptable. Por último se debe destacar la ausencia de inconsistencias tanto internas como externas y que la redundancia fue llevada al valor mínimo, previniendo así de futuras inconsistencias como resultado de modificaciones en los requerimientos.

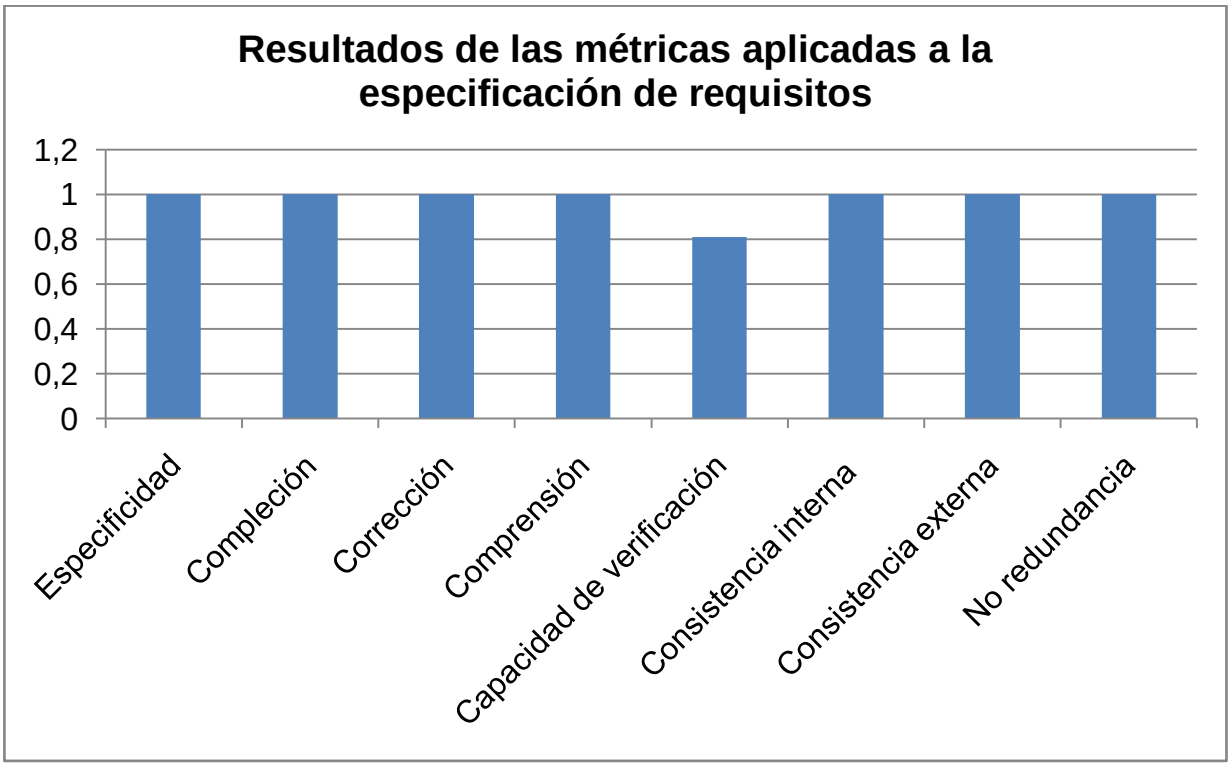


Figura 11 Resultados de las métricas aplicadas a la Especificación de requerimientos.

3.2.3 Lista de chequeo para el Modelo de casos de uso del sistema

Para la validación del Modelo de casos de uso del sistema se aplicó una lista de chequeo encaminada a verificar la calidad de cada uno de los artefactos contenidos dentro del documento. Se hicieron varias revisiones corrigiéndose cada una de las no conformidades detectadas.

La gráfica siguiente muestra los resultados obtenidos en la aplicación de la lista de chequeo al Modelo de casos de uso del sistema evidenciando que ya en la tercera revisión cada artefacto satisfizo en un 100% los parámetros evaluados.

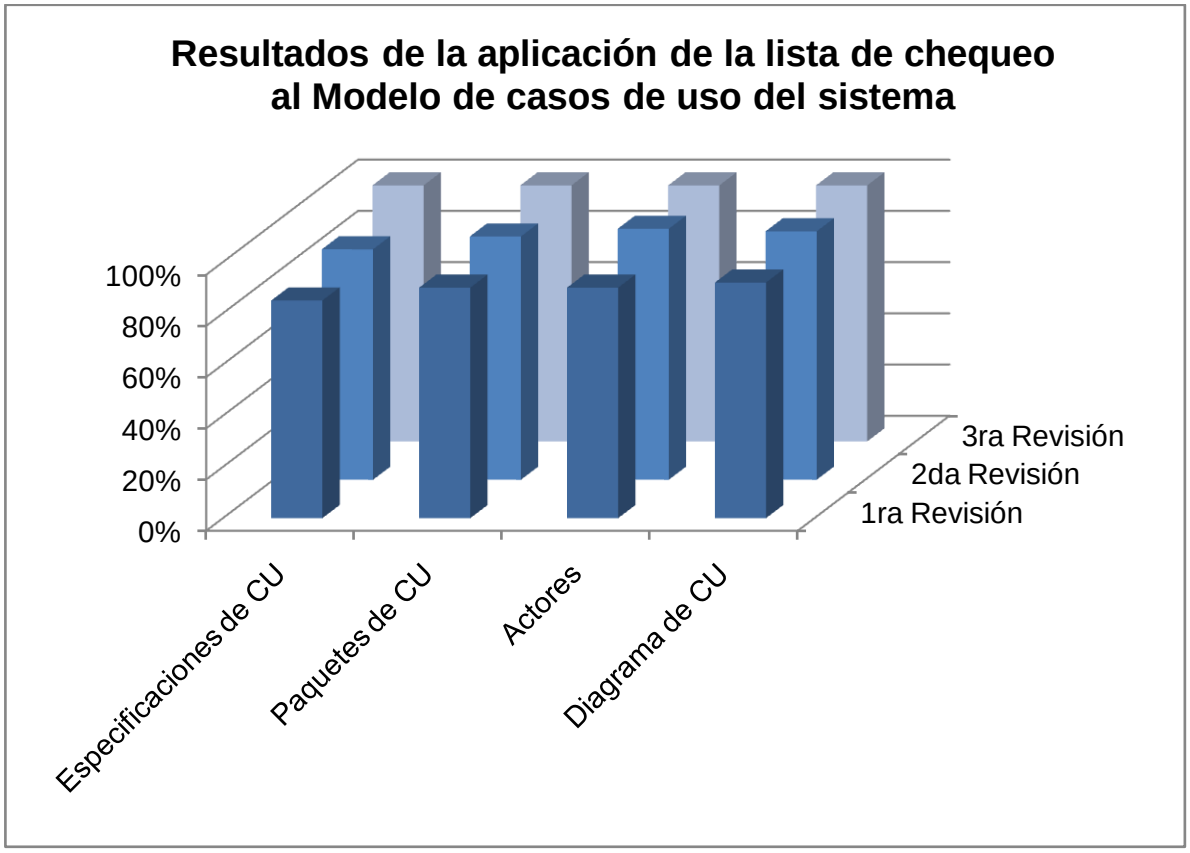


Figura 12 Resultados de la aplicación de las listas de chequeos al Modelo de casos de uso del sistema.

3.3 Validación para el Modelo de diseño.

Para la validación del Modelo de diseño se emplearon varias métricas que permitieron evaluar la calidad del mismo en diferentes aspectos tales como arquitectura, componentes y clases contribuyendo de esta forma a un análisis abarcador y profundo.

3.3.1 Métricas de diseño Arquitectónico.

Complejidad Estructural: $S(i) = f_{out}^2(i)$.

$S(i)$: Complejidad estructural.

$f_{out}\,(i)$: Expansión del módulo Acciones Centralizadas que indica el número de módulos que son invocados directamente por este.

$S(i) = 2^2 = 4$

La complejidad de datos: $D(i) = \frac{V(i)}{f_{out}\,(i)+1}$

$D(i)$: Complejidad en la interfaz interna del módulo Acciones Centralizadas.

$V(i)$: Número de variables de entrada y salida que entran y salen del módulo Acciones Centralizadas.

$f_{out}\,(i)$: Expansión del módulo Acciones Centralizadas, que indica el número de módulos que son invocados directamente por el módulo Acciones Centralizadas.

$D(i) = \frac{16}{2+1} = 5.33$

La complejidad del sistema: $C(i) = S(i) + D(i)$

$C(i)$: Complejidad del sistema.

$C(i) = 4 + 5.33 = 9.33$

Los umbrales definidos para evaluar el resultado de esta métrica se muestran a continuación:

Complejidad	**Estructural $S(i)$**	**Datos $D(i)$**	**Sistema $C(i)$**
No complejo	1, 4. 9, 16 ,25	<=7	<=32
Complejo	36, 49, 64	>7 y <=12	>32 y <=50
Muy Complejo	81…	>12	>50

Tabla 4 Umbrales para las métricas de diseño arquitectónico.

Los resultados obtenidos arrojaron que el módulo no es complejo lo que facilitará los procesos de integración y pruebas del módulo Acciones Centralizadas.

3.3.2 Métricas de diseño a nivel de componente

Cohesión Funcional Débil: $CFD = \frac{número\ de\ elementos\ adhesivos(i)}{número\ de\ elementos\ (i)}$

CFD**:** Cohesión funcional débil.

(i)**:** Se define como la muestra.

$Elementos\ adhesivos$: Se le llamará adhesivo a un elemento que aparece en dos o más rebanadas.

Para la aplicación de esta métrica se analizaron las relaciones entre las clases del módulo determinándose las señales de unión. A continuación se muestran los resultados obtenidos:

Clases	Usabilidad
1. GestionarACElaborarController	3
2. GestionarSolicitudACController	1
3. GestionarACController	1
4. GestionarACElaborarFacade	1
5. MostrarACFacade	2
6. AceptarACFacade	2
7. GestionarACElaborarBO	3
8. GestionarSolicitudACBO	2
9. GestionarACBO	1
10. BaseDao	4
11. DocumentoDao	3
12. DatosACDao	4
13. SolicitudACDao	3
14. NomencladorACDao	2

Tabla 5 Usabilidad de las clases.

Según los datos de las clases analizadas se tiene que:

$$CFD = \frac{\text{número de elementos adhesivos (i)}}{\text{número de elementos (i)}}$$

$$CFD = \frac{10}{14} = 0.71$$

Mientras más cerca están los valores de CFD de 1 mayor será la cohesión del módulo. Los resultados demuestran que el valor de CFD es alto. La relación del número de clases adhesivas con el número total de elementos de la muestra determina que el subsistema de diseño Acciones Centralizadas de SINAPSIS posee una cohesión funcional alta para un 71% de fortaleza.

3.3.3 Métricas Orientadas a clases.

Tamaño de clase (TC)

Para medir el tamaño de clase se tienen en cuenta los siguientes aspectos:

- **Total de operaciones** incluyendo las heredadas de las clases padres e interfaces que implementen.
- **Cantidad de atributos** incluyendo los heredados de las clases padres.
- **Promedio general** de los dos anteriores para el sistema completo.

Los umbrales para la evaluación de esta métrica se muestran a continuación.

Clasificación	Valores de los umbrales
Pequeña	<=20
Media	>20 y <=30
Grande	>30

Tabla 6 Umbrales para la métrica Tamaño de clase.

En la aplicación de esta métrica se analizaron las principales clases del módulo.

Clases	Nº. de atributos	Nº. de operaciones
1. GestionarACElaborarController	3	6
2. GestionarSolicitudACController	4	11
3. GestionarACController	3	12
4. GestionarACElaborarFacade	1	4
5. MostrarACFacade	1	4
6. AceptarACFacade	2	7
7. GestionarACElaborarBO	2	6
8. GestionarSolicitudACBO	3	9
9. GestionarACBO	2	8
10. BaseDao	0	5
11. DocumentoDao	0	8
12. DatosACDao	0	12
13. SolicitudACDao	0	12
14. NomencladorACDao	0	13

Tabla 7 Tamaño de clases.

El promedio de atributos y operaciones de las 14 clases analizadas fue 9.86 que está dentro del rango de clases pequeñas. Este resultado influye positivamente en la reutilización e implementación de las clases del módulo.

Árbol de Profundidad de Herencia (APH).

Para la aplicación de esta métrica se tuvieron en cuenta los diagramas de clases pertenecientes a los paquetes Web, Facade, Bussines y DAO. Los resultados obtenidos se evaluaron de acuerdo a lo enunciado por Lorenz y Kidd, que plantean que más de seis niveles de herencia indican un abuso en el empleo de este recurso.

Paquetes	Profundidad
1. vnz.sinapsis.acciones.web	1
2. vnz.sinapsis.acciones.facade	1
3. vnz.sinapsis.acciones.bussines	1
4. vnz.sinapsis.acciones.dao	3

Tabla 8 Profundidad de los paquetes de diseño.

A partir de los datos obtenidos se arriba a la conclusión de que el nivel más alto de herencia es de tres, este valor se encuentra dentro del umbral definido para determinar que el diseño no es complejo y denota un uso adecuado de este recurso de la Programación Orientada a Objetos.

El certificado de liberación por parte del equipo de calidad de la facultad quince para la Especificación de requerimientos, el Modelo de casos de uso del sistema y el Modelo de diseño se encuentra en el **anexo 3**.

3.4 Conclusiones

En este capítulo se analizaron los resultados obtenidos durante el desarrollo del trabajo, es decir, se evaluaron la Especificación de requerimientos, los artefactos que componen el Modelo de casos de uso del sistema y los que componen el Modelo del diseño, arribándose a las siguientes conclusiones:

- Las actas de liberación por el grupo de calidad CEGEL y de aceptación por parte de los clientes así como las métricas aplicadas confirmaron que la Especificación de requerimientos y el Modelo de casos de uso del sistema poseían una calidad aceptable.
- La liberación realizada por el grupo de calidad CEGEL y las métricas aplicadas a los elementos del diseño del módulo Acciones Centralizadas validan la calidad del diseño elaborado.

- Los valores de 4 y 5.33 para la complejidad estructural y de datos respectivamente indican que el sistema no es muy complejo.
- El diseño de las clases del módulo Acciones Centralizadas posee una cohesión funcional con un 71% de fortaleza, lo que demuestra que sus elementos están altamente cohesionados.
- El 100% de las clases diseñadas están consideradas como pequeñas, lo que facilitará el proceso de construcción del módulo Acciones Centralizadas.

CONCLUSIONES GENERALES

Al concluir el presente trabajo se arriba a las siguientes conclusiones:

- La aplicación de técnicas para la identificación y análisis de requerimientos propició que se obtuvieran resultados satisfactorios en la especificación del módulo Acciones Centralizadas.
- La elaboración del Modelo de casos de uso del sistema facilitó un mayor entendimiento y un acuerdo común entre los clientes y los desarrolladores, con respecto a las funcionalidades que el módulo Acciones Centralizadas debe brindar.
- La construcción del Modelo de diseño permitió obtener los elementos que deberán ser implementados para alcanzar un sistema que cumpla con los requerimientos de software del mismo.
- El análisis de los artefactos obtenidos como resultado de la especificación y diseño del módulo Acciones Centralizadas reveló una calidad aceptable en los mismos.

RECOMENDACIONES

Se recomienda:

- Elaborar los artefactos restantes pertenecientes al diseño, que son necesarios para continuar con el desarrollo del módulo y que no los realiza el diseñador.
- Continuar con el resto de los flujos de trabajo que propone la metodología RUP (implementación y prueba).
- Continuar perfeccionando la especificación y el diseño del módulo mediante la actualización de los cambios que sean necesarios durante las etapas de implementación y pruebas.
- Realizar la administración de requerimientos.

BIBLIOGRAFÍA

Alvarez, Miguel Angel. 2002. Desarrollo Web. [En línea] 2002. http://www.DesarrolloWeb.com.

Beck, Kent. 2002. *Una explicación de la Programación extrema: aceptar el cambio.* s.l. : Addison-Wesley Iberoamericana Espanya, S.A. - 2002, 2002. pág. 216. 8478290559.

Bieman, J. M. y Ott, L. M. 1994. *Measuring Functional Cohesion.* 1994. págs. 308-320. Vol. 20.

Davis, Alan, y otros. 1993. *Identifyng and Measuring Quality in a Software Requirements Specification.* 1993.

Duran, A., y otros. 1999. *A Requirements Elicitation Approach Based in Templates and Patterns.* 1999.

G.S.I. 2007. Grupo Soluciones GSInnova. [En línea] 2007. http://www.rational.com.ar/herramientas/roseenterprise.html.

grupo INTERCOM. 1997-2010. SOFTONIC. [En línea] 1997-2010. http://axure-rp.softonic.com/.

IBM. 1997. *Developing Object Oriented Software.* s.l. : IBM Object Oriented Technology Center. Prentice-Hall, 1997.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El proceso unificado de desarrollo de software.* 1ra edicion en español. 2000.

Koch, N. 2001. *Software Engineering for Adaptative Hypermedia Applications.* s.l. : Uni-Druck Publishing Company, Munich. Germany, 2001.

Larman, Craig. 1999. *UML y Patrones, introduccion al análisis y diseño orientado a objetos.* 1999.

Microsoft Corparation. 2010. Microsoft Office Online. [En línea] 2010. http://office.microsoft.com/es-hn/visio/HA101656403082.aspx.

Microsoft Corporation. 2010. msdn Visual Studio. [En línea] 2010. http://msdn.microsoft.com/es-es/library/72bd815a(VS.80).aspx.

—. 2000. Servicios de Consultoría de Microsoft. [En línea] 2000. http://www.microsoft.com/colombia/portafolio/msf.htm.

Olsina, L. 1999. *Metodología cualitativa para la evaluacióny comparación de la calidad de sitios web.* 1999.

Oracle Corporation. 2010. Oracle Corporation. [En línea] 2010. http://java.sun.com/applets/.

Pan, D., Zhu, D. y Johnson, K. 2001. *Requirements Engineering Techniques.* 2001.

Peña, R. 1998. *Diseño de programas. Formalismo y abstracción.* 1998.

Pressman, Roger S. 2005. *Ingenieria de Software: Un enfoque práctico.* 2005.

Raghavan, S. y Zelesnik, Ford, G. 1994. *Lectures Notes of Requirements Elicitacitation. Educational Materials.* 1994.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. *El Lenguaje Unificado de Modelado.Manual de referencia.*

SPARX System Pty L tb. 2000-2007. SPARX System. *SPARX System.* [En línea] SPARX System, 2000-2007. http://sparxsystems.com.ar/products/ea.html.

Sparx Systems Pty Ltd. 2000-2007. Enterprise Architect - Herramienta de diseño UML. [En línea] 2000-2007. http://www.sparxsystems.com.ar/products/ea.html.

—. 2000 - 2010. Sparx Systems. [En línea] 2000 - 2010. http://www.sparxsystems.com.ar/about.html.

the PHP Documentation Group. 1997-2008. Manual de PHP. [En línea] 1997-2008. http://www.php.net/docs.php.

UDELAB. 2009. UDELAB. *Universidad de las americas Puebla.* [En línea] 2009. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/capitulo4.pdf.

Universidad de las Ciencias Informáticas. 2009. Entorno Virtual de Aprendizaje. [En línea] 2009. http://eva.uci.cu.

Villaverde, Pablo. 2006-2007. ASP.Recursos y características. [En línea] 2006-2007. http://tgp0607.awardspace.com/Recursos_ASP.pdf.

Visual Paradigm. 1999-2010. Visual Paradigm. [En línea] 1999-2010. http://www.visual-paradigm.com/product/vpuml/provides/.

Weidenhaupt, K., y otros. 1999. *Scenarios in Systems Development: Current Practice.* 1999.

ANEXOS

Anexo 1. Acta de liberación de la Especificación de requerimientos de software, el Modelo de casos de uso del sistema y el Modelo de diseño.

Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Martes, 04 de mayo de 2010.

Luego de haber efectuado 2 iteraciones de revisiones a los artefactos: Especificación de Requisitos No Funcionales, Especificación de Requisitos, Modelo de Sistema y Modelo de diseño del módulo Acciones Centralizadas del proyecto SINAPSIS del Centro CEGEL de la Facultad 15 y haberse detectado un promedio de 6 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que se considera que los artefactos están correctamente y listos para ser utilizados.

Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarez

Anexo 2. Acta de aceptación.

PDVSA

Caracas, 07 de Diciembre de 2009.

Señor **Juan Carlos Montané Izaguirre**
Jefe del Proyecto **"SISTEMA NACIONAL PÚBLICO PARA EL SEGUIMIENTO DE INVERSIONES Y SECTORES".**

Estimado: **Juan Carlos Montané Izaguirre**

Después de analizado los documentos:

- Especificación de requisitos de software. Módulo de Proyectos.
- Especificación de requisitos de software. Módulo de Acciones Centralizadas.
- Especificación de requisitos de software. Módulo de Seguimiento y Control.
- Especificación de requisitos de software. Módulo de Administración de Usuario.
- Especificación de requisitos de software. Módulo de Configuración.
- Especificación de requisitos de software. Módulo de Reporte.
- Especificación de requisitos de software. Módulo de Migración de Datos.
- Modelo de Sistema. Módulo de Proyectos.
- Modelo de Sistema. Módulo de Acciones Centralizadas.
- Modelo de Sistema. Módulo de Seguimiento y Control.

PETROLEOS DE VENEZUELA
Avenida Libertador, edificio Petróleos de Venezuela, Torre Este, La Campiña. Apartado Postal 61373. Caracas 1060-A. Venezuela
Teléfono: (02) 708 1111 Fax: (02) 708 3257 www.pdvsa.com

PDVSA

- Modelo de Sistema. Módulo de Administración de Usuario.
- Modelo de Sistema. Módulo de Configuración.
- Modelo de Sistema. Módulo de Reporte.
- Modelo de Sistema. Módulo de Migración de Datos.
- Arquitectura de Software.

Visión del Proyecto concretado en el **Anexo 3 al Convenio PDVSA-ALBET**, manifestamos nuestra conformidad.

Atentamente,

Saludos,

07/12/09

Saúl Chirinos Gutiérrez
Gerente del Centro de Servicios Comunes-Occidente.
Jefe de Proyecto Sistema Nacional Público para el
Seguimiento de Inversiones y Sectores

PETROLEOS DE VENEZUELA
Avenida Libertador, edificio Petróleos de Venezuela, Torre Este, La Campiña. Apartado Postal 61373. Caracas 1060-A. Venezuela
Teléfono: (02) 708 1111 Fax: (02) 708 3257 www.pdvsa.com

GLOSARIO

Actividades: Son una unidad más atómica que permite desagregar aún más la acción centralizada permitiendo así una planificación más exacta y un control más eficiente.

Acciones Centralizadas: Consisten en la realización de peticiones de financiamientos para sustentar las operaciones empresariales del organismo que las procura.

EE Entes Ejecutores: Un Ente Ejecutor puede ser una empresa o institución, es uno de los responsables de proyecto.

ERD: Acrónimo de Entity Relationship Diagram (Diagrama de Entidad Relación), es un mecanismo para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

ERWin: Herramienta para el diseño de base de datos, que brinda productividad en su diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la base de datos diseñada. Genera automáticamente las tablas.

Especificación del módulo: Conjunto de artefactos generados durante el flujo de trabajo Requerimientos.

Ejecución Financiera: Realización financiera del proyecto, constituido por los pagos y las transacciones financieras referentes al proyecto.

Ejecución Física: Realización física del proyecto, lo constituyen las actividades que se deben realizar en el proyecto.

Fuentes de financiamientos: Fuentes por las cuales le llegan los recursos a los entes.

Hibernate: Herramienta de Mapeo objeto-relacional.

IEEE: Corresponde a las siglas de The Institute of Electrical and Electronics Engineers (el Instituto de Ingenieros Eléctricos y Electrónicos), es la asociación técnica y profesional, sin fines de lucro, más grande del mundo formada por profesionales de todas las disciplinas de la ingeniería. Su trabajo es promover la creatividad, el desarrollo y la integración, compartir y aplicar los avances en las tecnologías de la información, electrónica y ciencias en general para beneficio de la humanidad y de los mismos profesionales.

OMG: Object Management Group es una asociación sin fines de lucro formada por grandes corporaciones, muchas de ellas de la industria del software, como IBM, Apple, Sun Microsystems y HP. Se encarga de la definición y el mantenimiento de estándares para aplicaciones de la industria de la computación, como UML, CORBA y otros.

Paquetes de diseño: Es una colección de clases, relaciones, realizaciones de casos de uso, diagramas y otros paquetes que estén de alguna forma relacionados. No definen un comportamiento o semántica bien definida.

Programación Orientada a Objetos (POO): Es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (propiedades o datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). Expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

Release: Producto final preparado para lanzarse como versión definitiva a menos que aparezcan errores que lo impidan. Implementa todas las funciones del diseño y se encuentra libre de cualquier error que suponga un punto muerto en el desarrollo.

XML: Acrónimo de Extensible Markup Language («lenguaje de marcas ampliable»), es un metalenguaje extensible de etiquetas, aunque no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.