Universidad de las Ciencias Informáticas

Facultad 3

Título:

Desarrollo de un módulo de Mantenimiento Centrado en Confiabilidad para la gestión de la información de los requerimientos de mantenimiento.

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

Autor: Yoan Danilo Lemus Becerra

Tutores: Ing. Miguel Angel Sánchez Palmero

Ing. Olga Yarisbel Rojas Grass

La Habana, Junio de 2014

Año 56 de la Revolución

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año_____.

_____

Yoan Danilo Lemus Becerra

Firma del Autor

_____

Ing. Miguel Angel Sánchez Palmero

Firma del Tutor

_____

Ing. Olga Yarisbel Rojas Grass

Firma del Tutor

.

Tutor:

- **Nombre y apellidos:** Ing. Miguel Ángel Sánchez Palmero
- **Correo electrónico:** masanchez@uci.cu
- **Situación laboral:** Especialista general
- **Institución:** Universidad de las Ciencias Informáticas (UCI)
- **Dirección:** Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros

Tutora:

- **Nombre y apellidos:** Ing. Olga Yarisbel Rojas Grass
- **Correo electrónico:** yarisbel@uci.cu
- **Situación laboral:** Especialista general
- **Institución:** Universidad de las Ciencias Informáticas (UCI)
- **Dirección:** Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros

.

´´Solo existen dos días en el año en que no se puede hacer nada.

Uno se llama ayer y el otro mañana.

Por lo tanto hoy es el día ideal para amar, creer, hacer y principalmente vivir´´

Dalai Lama

.

AGRADECIMIENTOS

A la Revolución por hacer de la educación un derecho de todos y no un privilegio de pocos.

A mis profesores que de forma tan gentil y profesional nos impartieron sus conocimientos y enseñanzas.

A mis tutores por brindarme su apoyo, sin el cual no hubiera sido posible la realización de esta tesis.

.

DEDICATORIA

A mi familia en especial a mis padres y hermanos por su amor incondicional, a mi novia a quien amo de forma exclusiva, a mis amigos con quienes siempre he podido contar y a todas las personas que han confiado en mí. Gracias.

RESUMEN

El Centro de Informatización de Entidades de la Universidad de las Ciencias Informáticas tiene la tarea de realizar la informatización de un proceso de mantenimiento para contribuir a la gestión de los requerimientos de información de mantenimiento. Para ello se apoya en la Confiabilidad Operacional, la cual propone un conjunto de buenas prácticas así como una metodología que facilita la toma de decisiones a la hora de realizarle el mantenimiento a los activos.

En el presente trabajo se realiza el análisis, diseño e implementación de un módulo de Mantenimiento Centrado en la Confiabilidad para un Sistema de Confiabilidad Operacional a raíz de los estudios realizados sobre la poca utilización de herramientas informáticas para la gestión del mantenimiento.

El desarrollo de este módulo se validó a partir de pruebas de caja blanca, caja negra y matrices de trazabilidad, donde se demostró el seguimiento de los requerimientos de información de mantenimiento desde el Modelado del negocio hasta la Implementación. El módulo facilita la toma de decisiones para el Ingeniero de Mantenimiento.

PALABRAS CLAVE

Análisis MCC, Confiabilidad Operacional, Hoja de Decisión, Hoja de Información, Mantenimiento Centrado en Confiabilidad.

ÍNDICE

ÍNDICE

INTRODUCCIÓN

El mantenimiento al igual que otras ciencias de la ingeniería, ha evolucionado a gran escala con el paso del tiempo, este cambio ha traído nuevas políticas e ideologías, que se han adaptado al ritmo de vida de las empresas de clase mundial. Las redes de automatización, los sistemas de control, la robótica, la sensórica, hacen parte de un gran número de ciencias innovadoras, que día a día son más comunes en la mayoría de procesos de manufactura, este tipo de tecnologías hacen que el mantenimiento haga parte de esta renovación, es por esto, que estrategias de última generación como el RCM, la Confiabilidad Operacional, el mantenimiento proactivo, entre otras, se hagan presentes al momento de realizar un estudio que determine las condiciones normales operativas de un sistema o equipo (Montaña Riveros, y otros, 2006).

El Mantenimiento Centrado en Confiabilidad[1] es una estrategia de mantenimiento según los investigadores Montañas y Rosas en su tesis de grado. El MCC tiene como principal objetivo mantener la función que realizan los equipos, el gran interés radica en lo que se quiere que haga la máquina desde el punto de vista productivo (Barrios Gárciga, 2005).

Los principios básicos del MCC se desarrollaron en los años 60 del pasado siglo para la industria aeronáutica norteamericana, en los 70 se generaliza su uso en el ejército y la marina estadounidense y a principio de la década de los 80, esta metodología se comienza a transferir a otros sectores industriales (Barrios Gárciga, 2005). En los últimos años, la aplicación de la estrategia MCC se generaliza a la práctica de la totalidad de los sectores industriales a nivel global.

El sistema empresarial cubano, bajo la implementación de los lineamientos de la política económica y social del Partido y la Revolución, debe ejecutar al menos 13 lineamientos relacionados con mantenimientos de las esferas: constructivo, tecnológico, industrial, entre otras. La importancia está dada debido a que el mantenimiento permite asegurar que los activos continúen desempeñando las funciones deseadas (Moubray, y otros, 2013).

Según esta definición de Moubray, se debe tomar en consideración que la industria en Cuba funciona con maquinarias y recursos de muchos años de explotación. Es entendible que el

[1] RCM por sus siglas en inglés y en lo adelante MCC

estado cubano comience a emplear estrategias que permitan garantizar el funcionamiento estable de sus activos.

En atención al papel rector del Ministerio de Industrias de Cuba para reactivar el mantenimiento industrial se creó un Grupo de trabajo temporal constituido por representantes de todos los ministerios. Estos representantes tienen incidencia en la elaboración e implantación de la política de mantenimiento, contando con la participación del Centro de Estudios en Ingeniería de Mantenimiento (CEIM)[2].

El CEIM ha comenzado la utilización del MCC. Esta estrategia determina lo que debe hacerse para asegurar que un elemento físico continúe desempeñando las funciones deseadas en su contexto operacional presente (Moubray, y otros, 2013). Este centro en colaboración con el Centro de Informatización de Entidades (CEIGE) de la Universidad de las Ciencias informáticas (UCI), han decidido buscar una solución que proporcione mayores facilidades para el trabajo de los especialistas de la ingeniería de mantenimiento, basado en los resultados del Grupo de trabajo temporal ya mencionado.

A partir del mes de febrero de 2013 se elaboró un cronograma por el Grupo de trabajo temporal, que comprendía en sus etapas el desarrollo de un proceso de diagnósticos del estado técnico de las instalaciones y de la gestión del mantenimiento en 91 empresas seleccionadas pertenecientes a 8 organismos. Los principales problemas identificados están relacionados con los siguientes aspectos: gestión del mantenimiento, carencia de recursos, de organización y dirección, de capacitación y formación de profesionales de la actividad. Este diagnóstico concluyó en el mes de junio de 2013, con resultados que afirman que el mantenimiento en la industria cubana carece de una política que asegure el estado técnico de las instalaciones industriales y que solo el 15,5 % de los problemas identificados, corresponden a la disponibilidad de recursos y falta de financiamiento, y el 84,5 % de los problemas son de planificación, organización, gestión del mantenimiento, capacitación y dirección (Cubadebate, 2013).

Por otra parte, en el estudio realizado por Raña González y colectivo en el 2010, se explica la evaluación de la función de mantenimiento en 10 empresas con características diferentes en cuanto a su funcionamiento y organización. Los resultados de esta evaluación evidenciaron de

[2] CEIM, centro del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE).

manera general que el peor resultado estuvo en el atributo relacionado con el soporte informático, o sea, que se usan pocas herramientas informáticas para la gestión de la función de mantenimiento (Raña González et al, 2010). De forma general, la carencia de herramientas informáticas que implementen estrategias de mantenimiento adecuadas, es otra de las dificultades identificadas que se asocia a la gestión del mantenimiento. Este resultado se incluye también en el estudio del Grupo de trabajo temporal.

A raíz de los estudios mencionados se infiere que la poca utilización de herramientas informáticas para la gestión del mantenimiento, requiere un mayor esfuerzo del personal que puede traer consigo errores en el funcionamiento de la entidad. El tiempo dedicado por los ingenieros de mantenimiento para la gestión de la información respecto a los activos, análisis y hojas de decisiones, se hace extenso y no se toman las decisiones correctas en el momento oportuno.

Los problemas ya mencionados impactan directamente en la economía del país. Con el fin de lograr en gran medida su mitigación, se formuló la investigación que ocupa este trabajo de diploma, en el que se plantea como **problema a resolver**: ¿cómo gestionar la información de los requerimientos de mantenimiento que sirva como fuente de datos del Sistema Informático para la Ingeniería de Mantenimiento?

El **objeto de estudio** se define en el proceso de Mantenimiento Centrado en Confiabilidad y el **campo de acción** se centra en los sistemas informáticos para el Mantenimiento Centrado en Confiabilidad.

Para solucionar el problema planteado se propone como **objetivo general** desarrollar un módulo de Mantenimiento Centrado en Confiabilidad para la gestión de los requerimientos de información de mantenimiento.

Para darle cumplimiento al objetivo general se plantearon los siguientes **objetivos específicos**:

- Realizar un estudio del estado del arte para la elaboración del marco teórico alrededor del objeto de estudio.
- Elaborar el análisis y diseño de la solución para el módulo de Mantenimiento Centrado en Confiabilidad.
- Realizar la implementación de la solución para la obtención del módulo de Mantenimiento Centrado en Confiabilidad.

- Validar la propuesta de solución a través de pruebas de caja blanca y caja negra.

Como **idea a defender** de este trabajo se plantea que el desarrollo del módulo de Mantenimiento Centrado en Confiabilidad permitirá gestionar la información de los requerimientos de mantenimiento que sirva como fuente de datos del Sistema Informático para la Ingeniería de Mantenimiento.

Para el correcto desarrollo de la investigación se utilizaron **métodos científicos** teóricos y empíricos. Entre los métodos teóricos utilizados se encuentra el análisis Histórico-Lógico para el estudio de estrategias y sistemas de mantenimiento implementados en diferentes empresas, determinando sus principales características, ventajas y desventajas para identificar aspectos a tener en cuenta durante la implementación. También se utiliza el método de Modelación para gestionar el modelo de dominio a desarrollar en la implementación, obtener los diferentes diagramas necesarios para el entendimiento del negocio y mantenimiento del producto; se trabaja además con el método Analítico-Sintético donde se realiza un estudio de las principales características del proceso de MCC. De los **métodos empíricos** se emplea la Observación para determinar los principales problemas que presenta el proceso de mantenimiento en las empresas cubanas.

El presente trabajo cuenta con tres capítulos:

Capítulo 1 Fundamentación teórica. En este capítulo se realiza un estudio de los principales conceptos asociados. Se analizan y describen diferentes sistemas de MCC para obtener indicadores que permitan su comparación, aportando funcionalidades necesarias en el producto final. Se definen las técnicas de captura y validación de requisitos, se describe el modelo de desarrollo utilizado y el flujo de trabajo a seguir en el trabajo de diploma. Se describen las herramientas y tecnologías necesarias definidas para obtener la solución propuesta, así como las métricas para validar el diseño y las pruebas de software a efectuar.

Capítulo 2 Propuesta de solución. En este capítulo se realiza el modelado del dominio, obteniendo el modelo conceptual. Se identifican y describen los requisitos funcionales y no funcionales. Seguidamente se realiza la validación, priorización y trazabilidad de los requisitos. Se realiza del modelo de datos mediante un diagrama de entidad relación. Se realizan los diagramas de clases del diseño, finalizando con la aplicación de métricas para evaluar el diseño propuesto.

Capítulo 3 Implementación y pruebas. En este capítulo se realiza el mapa de componentes y la descripción del estándar de codificación utilizado. Se realiza además, el diagrama de despliegue, así como la validación de la disciplina de implementación utilizando las pruebas de cajas blanca y negra. Finalmente se realiza la matriz de trazabilidad Requisito-Diseño de casos de prueba.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

1.1. Introducción

En este capítulo se realiza un estudio de los conceptos básicos relacionados con el MCC, así como un estudio de diferentes herramientas MCC existentes en el mundo. Se describe el modelo de desarrollo a utilizar con un flujo de trabajo específico incorporado. Se describen las tecnologías, lenguajes de programación, entorno de desarrollo, gestor de base de datos y marco de trabajo utilizado para el desarrollo del trabajo. Se identifican los tipos de pruebas a realizar en el desarrollo del producto.

1.2. Conceptos generales asociados a mantenimiento

En el estudio del marco teórico de la investigación se estudiaron varios conceptos: mantenimiento, confiabilidad operacional y MCC. Estos conceptos resultaron de relevancia para comprender los elementos del MCC.

1.2.1. Confiabilidad operacional

Los investigadores Amendola y Depool afirman que la confiabilidad operacional es la integración de la confiabilidad del equipo, de los proceso, además de la humana, de este modo se requiere centrar la atención del gestor de mantenimiento para garantizar la eficacia y calidad del trabajo. Basados en la integración de equipos de trabajo y la recopilación de datos se puede proporcionar acciones imprescindibles que se apoyan en sus observaciones (Amendola, y otros, 2005).

Según Aguinaga, la Confiabilidad Operacional (ver figura. 3) se define como una serie de procesos de mejora continua, que incorporan en forma sistemática, avanzadas herramientas de diagnóstico, metodologías de análisis y nuevas tecnologías, para optimizar la gestión, planeación, ejecución y control de la producción industrial. La Confiabilidad Operacional lleva implícita la capacidad de una instalación (procesos, tecnología, gente), para cumplir su función o el propósito que se espera de ella, dentro de sus límites de diseño y bajo un específico contexto operacional. Es importante, puntualizar que en un sistema de Confiabilidad Operacional es necesario el análisis de sus cuatro parámetros operativos: Confiabilidad

Humana, Confiabilidad de los Procesos, Mantenibilidad y Confiabilidad de los equipos (Aguinaga Barragán, 2007).

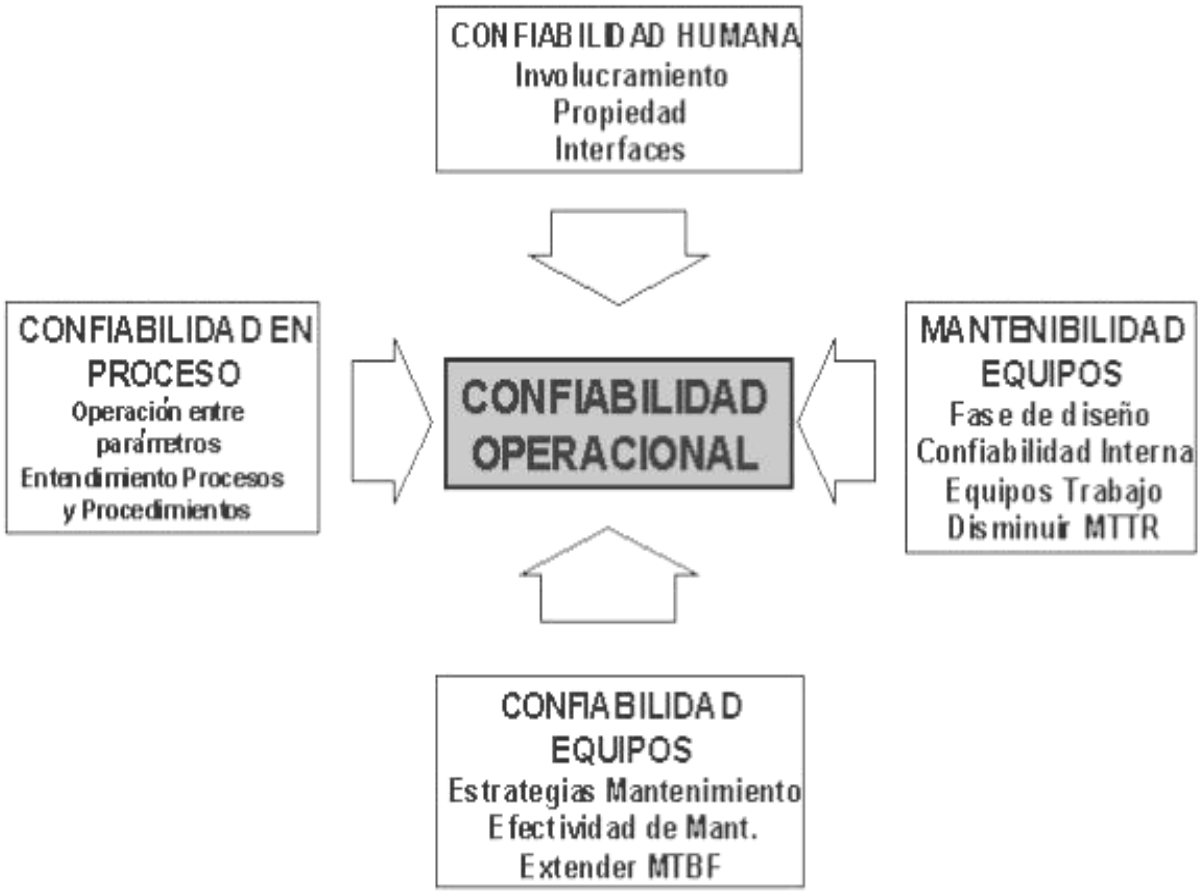

Figura 1 Confiabilidad Operacional

En los conceptos tratados sobre Confiabilidad Operacional se destacan 4 elementos: Confiabilidad de los Procesos, Confiabilidad Humana, Confiabilidad de los Equipos y su Mantenibilidad. La Confiabilidad Operacional no se debe enfocar solo a un área sino a toda la entidad, analizando desde el punto de vista de todo el sistema, para buscar mejoras al sistema con la ayuda de nuevas tecnologías. Para esta investigación se toma el concepto emitido por el investigador Aguinaga Barragán. En este concepto se aprecia que el mantenimiento de los activos constituye uno de los elementos principales en los que se basa la Confiabilidad Operacional.

1.2.2. Mantenimiento

En el artículo publicado por Barrios y Ortiz se entiende por Mantenimiento al conjunto de acciones oportunas, continuas y permanentes dirigidas a prever y asegurar el funcionamiento normal, la eficiencia y la buena apariencia de sistemas, edificios, equipos y accesorios. Entendiéndose por **acciones**: efectos de hacer algo. Las acciones más significativas del mantenimiento son: planificación, programación, ejecución, supervisión y control; por **continuas**: que duran o se hacen sin interrupciones, y **permanentes**: que son de duración firme, constante, y perseverantes (Barrios, y otros, 2012).

En el mismo artículo aparece el concepto que lo relaciona con el conjunto de técnicas destinadas a conservar equipos e instalaciones en servicio durante el mayor tiempo posible (buscando la más alta disponibilidad) y con el máximo rendimiento o la combinación de actividades mediante un equipo o un sistema para mantener o restablecer, a un estado en el que puede realizar las funciones designadas (Barrios, y otros, 2012).

Por su parte la norma venezolana COVENIN, define al mantenimiento como un conjunto de acciones que permite conservar o restablecer un sistema productivo a un estado específico, para cumplir un servicio determinado (SENCAMER, 2011).

Según lo que explica De León, el concepto de mantenimiento puede definirse de distintas maneras, atendiendo al enfoque que se le dé en cada caso. Incluso resulta insuficiente, hoy en día, realizar una definición basada simplemente en términos económicos. Resulta obvio que el punto de partida del mantenimiento es mantener el correcto estado funcional de los equipos e instalaciones, sin embargo las consecuencias que el desarrollo de este principio elemental puede tener sobrepasan ampliamente el objetivo inicial. El mantenimiento produce disponibilidad y constituye una actividad improductiva que forma parte íntegramente del sistema de producción (Gómez de León, 1998).

Amendola define además, que el rol del mantenimiento dentro del contexto actual se puede describir de la siguiente forma: preservar la función de los activos aplicando estrategias efectivas de mantenimiento, costo-riesgo-beneficio (Amendola, 2009).

A partir de los conceptos tratados se define en la investigación que el mantenimiento es un conjunto de acciones encaminadas a garantizar el correcto funcionamiento de los activos, con

la más alta disponibilidad y el mejor rendimiento posible. El conjunto de acciones permite conservar o restablecer un activo a un estado deseado, de manera que pueda seguir cumpliendo con su función; estas acciones deben estar asociadas a estrategias efectivas de mantenimiento, como pueden ser las de MCC.

1.2.3. Mantenimiento Centrado en Confiabilidad

Aguinaga en su investigación define al Mantenimiento Centrado en Confiabilidad[3], reconoce que el mantenimiento no puede hacer más que asegurar que los elementos físicos continúan consiguiendo su capacidad incorporada, confiabilidad inherente. Es un proceso que se usa para determinar los requerimientos del mantenimiento de los elementos físicos en su contexto operacional. Una definición más amplia de MCC podría ser "un proceso que se usa para determinar lo que debe hacerse para asegurar que un elemento físico continúa desempeñando las funciones deseadas en su contexto operacional presente" (Aguinaga Barragán, 2007).

Por su parte Da Costa plantea que el MCC es un proceso usado para determinar sistemática y científicamente qué debe ser hecho para asegurar que los activos físicos continúen haciendo lo que los usuarios quieren que hagan. Ampliamente reconocido por los profesionales del mantenimiento como la forma más "costo-eficaz" de desarrollar estrategias de mantenimiento de clase mundial, MCC lleva las mejoras rápidas sostenidas y sustanciales en la disponibilidad y confiabilidad de planta, calidad de producto, seguridad e integridad ambiental (Da Costa Burga, 2011).

Una definición más completa es la dada por el investigador Amendola, en su libro gestión de Proyectos de Activos Industriales, este autor plantea, que el MCC es una metodología utilizada para determinar sistemáticamente, que debe hacerse para que los activos físicos continúen haciendo lo requerido por los usuarios en el contexto operacional presente y que consiste en analizar las funciones de los activos, ver cuáles son sus posibles fallas, detectar los modos de fallas o causas de fallas, estudiar sus efectos y analizar sus consecuencias, para a partir de la evaluación de las consecuencias o riesgos, determinar las estrategias más adecuadas de operación, tanto técnicamente factibles, como económicamente viables (Amendola, 2006).

[3] RCM, siglas en inglés de Reliability Centered Maintenance.

De los conceptos mencionados se concluye que el MCC es una metodología para identificar estrategias adecuadas de mantenimiento, que permitan el cumplimiento de los estándares necesarios en los procesos productivos, teniendo en cuenta la evaluación de los riesgos que implican las fallas de los equipos.

1.3. Análisis de los principales sistemas

El uso de sistemas de MCC permite a las empresas:

- Mayor seguridad y protección del entorno.
- Mejores rendimientos operativos.
- Mayor control de los costos del mantenimiento.
- Más larga vida útil de los equipos, debido al aumento del uso de las técnicas de mantenimiento.
- Poseer una amplia base de datos de mantenimiento.
- Mayor motivación de las personas.
- Mejor trabajo en equipo.

A continuación se relacionan algunas de las herramientas más usadas en el mundo para aplicar el MCC.

RELEX

El software RELEX integra un total de doce módulos, centralizados en una misma interfaz desde la que se puede acceder de manera rápida a todos ellos, con posibilidad de simultanearlos dos a dos. Estos módulos permiten hacer un análisis amplio y preciso de la fiabilidad de un equipo o sistema. Cada uno de ellos puede operar de forma totalmente independiente, pero a su vez, están integrados en una misma plataforma común (Relex Architect) que permite sincronizar los cálculos integrándolos en una misma interfaz. Esta estructura posibilita a cada módulo mostrar datos de forma dinámica, lo que asegura que los resultados de los cálculos realizados en un módulo estén inmediatamente disponibles en los otros. Este único nivel de interactividad proporciona consistencia e incrementa la exactitud en todos los análisis (Parra Márquez, y otros, 2013).

El módulo de análisis FMEA[4] realiza un análisis de los modos de fallo y sus efectos, es decir, analiza los potenciales modos de fallo de un sistema y los efectos resultantes de esos fallos. Soporta la mayoría de estándares industriales y permite realizar el diseño y la ejecución de análisis FMEA a nivel sistema (bloque funcional), grupos de piezas y a nivel componente. Las tasas de fallo calculadas por el módulo de Predicción de Fiabilidad se actualizan constantemente, y de forma instantánea en Relex FMEA. Combinando el módulo de Relex Fault Tree con el de FMEA se puede generar un árbol de fallos mostrando todos los modos de fallo que contribuyen a producir un efecto (Parra Márquez, y otros, 2013).

Relex permite la utilización de la herramienta por parte de distintos usuarios simultáneamente. Únicamente se puede acceder al software por autentificación de los usuarios autorizados.
El software posee una guía para instruir al usuario paso a paso y enseñarle a introducir datos de forma sencilla y ordenada, haciendo un recorrido muy básico sobre cada uno de los módulos. Una vez se realizan los diferentes análisis, el software elabora un informe de resultados. La aplicación para elaborar gráficos que posee Relex permite elegir entre una amplia variedad de plantillas ya confeccionadas, con la posibilidad de adaptarlas según requerimientos del usuario y visualizar mejor los resultados que se deseen (Parra Márquez, y otros, 2013).

Item Toolkit

El software Item Toolkit integra un total de 8 módulos relacionados con técnicas de Fiabilidad, Mantenibilidad, Disponibilidad, Seguridad y Riesgo. Estos módulos permiten hacer un análisis amplio y preciso de la fiabilidad de un equipo o sistema. Es válido destacar que este software no tiene módulo de RCM. Item Toolkit permite la utilización de la herramienta por parte de distintos usuarios simultáneamente. Permite configurar las aplicaciones de cada módulo en varios niveles. A nivel de sistema y a nivel de usuario, con distintas opciones de aplicación para facilitar su uso. Permite exportar los resultados a base de datos como Excel y Access (Parra Márquez, y otros, 2013).

[4] Failure Mode Effects Analysis

El módulo FMECA[5] proporciona una interfaz gráfica intuitiva con múltiples opciones para construir y realizar un análisis. Posee un árbol de amplia jerarquía y tablas diseñadas para una navegación agradable y una fácil entrada de datos. Su diseño permite identificar y analizar los potenciales modos fallos dentro de un sistema e identificar los riesgos que estos pueden provocar dentro del contexto operacional. La herramienta provee de un modelo de análisis de criticidad basado en la combinación de los factores de probabilidad de fallos por severidad (Parra Márquez, y otros, 2013).

RCMO Software

El sistema RCMO[6] fue realizado por la compañía privada Meridium. Es un módulo de MCC que está integrado y se gestiona a través de la propia interfaz de SAP PM. Las características principales del software son las siguientes:

- Integración de las recomendaciones de mantenimiento (RCMO) con los planes de mantenimiento de SAP (creación de órdenes de trabajo).
- Reevaluación automática de las estrategias de mantenimiento, medida de efectividad y búsqueda de mejoras.
- Matriz de criticidad configurable.
- Plantillas MCC y FMEA.
- Elaboración de informes que resumen los resultados MCC.
- Alerta de excepción.
- Rastreo de modos de fallo del catálogo de SAP PM y estimación de frecuencias óptimas.
- Posee guía o constructor de decisión lógica (soporte a la toma de decisión).
- Historial de revisiones automatizado, que rastrea los cambios en los análisis en función del tiempo.
- Integración con el módulo de Manejo de Documentos de SAP para facilitar el manejo de documentos de referencia al análisis.

[5] Failure Mode, Effects and Criticality Analysis

[6] Reability Centered Maintenance and Optimization

- Una vez el plan de estrategias es optimizado, él mismo puede enviarse a SAP (previa aceptación) como un plan de mantenimiento propuesto (Barberá Martínez, y otros, 2010).

El software RCMO sigue la lógica MCC y su estructura es acorde con esta metodología. Permite definir el sistema, realizar FMEA, diseñar políticas y estrategias de mantenimiento y optimizarlas mediante reevaluación automática, todo ello teniendo en cuanta el contexto operacional. Está totalmente integrado a SAP en todas sus funciones y el rendimiento en las operaciones del software es equivalente al de SAP. El hecho de que esté integrado en SAP, simplifica su utilización por parte de la organización además de numerosas ventajas derivadas de esta característica (no se hace necesario el exportar/importar datos) Universidad de Sevilla, 2010.

RCM++

Sistema desarrollado por ReliaSoft, líder en Ingeniería de la Confiabilidad, ha diseñado un conjunto de 12 herramientas relacionadas con el área de Fiabilidad y Análisis de Fallos. RCM++ incluye los estándares industriales de RCM más conocidos a nivel mundial: SAE[7] JA1011, SAE JA1012 y SAE J1739. El software posibilita la creación de paquetes de tareas de mantenimiento y facilita la agrupación de tareas individuales en paquetes basados en intervalos o equipos. Este módulo desarrolla todas las etapas propuestas por la metodología RCM y está integrado de forma directa con distintos módulos de Reliasoft. En términos generales, este módulo permite:

- Incluir la información básica sobre el contexto operacional de los equipos.
- Describir la configuración del equipo o sistema a evaluar.
- Desarrollar el análisis FMEA.
- Evaluar la criticidad de los modos de fallos del sistema evaluado.
- Introducir datos relacionados con el historial de fallos, con el objetivo de realizar evaluaciones básicas de indicadores de fiabilidad.

[7] Siglas en inglés para las normas emitidas por la Sociedad de Ingenieros Automotrices.

- Informes: resumen de funciones, fallos funcionales, modos de fallos, efectos y acciones a ejecutar por equipo (Barberá Martínez, y otros, 2010).

Renovegem 3.0 Standard Edition

El software Renovegem 3.0 Standard Edition ha sido desarrollado por Renovetec para distribuir de forma gratuita. El software está orientado a pequeñas y medianas empresas que prefieren invertir en herramientas y bienes de equipo antes que en programas informáticos de dudosa rentabilidad. Se ha desarrollado para que pueda manejarse de forma intuitiva y sencilla, pero a la vez se ha dotado con una potente herramienta de búsqueda que permite filtrar y exportar cualquier información contenida en su base de datos permitiendo la confección de informes a medida, que pueden ser realizados por el propio usuario (Renove Tecnología S.L, 2009-2014).

Entre sus principales funciones se encuentran:

- Gestión de activos, con su árbol jerárquico.
- Gestión de personal.
- Gestión del mantenimiento programado y de las gamas de mantenimiento. Incluye la creación automática del plan de mantenimiento programado.
- Control de la programación de mantenimiento.
- Gestión de órdenes de trabajo.
- Gestión del repuesto.
- Gestión económica del mantenimiento: costes de O.T., costes de un periodo, compras.
- Determinación de los indicadores más usuales en mantenimiento (Fuente de elaboración propia, consultado en el sistema).

El software incluye otras funcionalidades que no están disponibles para esta versión que se distribuye de forma gratuita.

PMX pro PLUS

El software PMX pro Plus fue desarrollado por CWORKS y se distribuye de forma gratuita. Está basado en Access, por ello admite algunas interesantes posibilidades de personalización y

configuración de informes; permite acceder a la base de datos, recuperar los datos directamente desde allí, cambiar informes y pantallas, y todo ello con unos conocimientos muy básicos. La versión gratuita no trabaja en red.

Permite implementar el árbol jerárquico de equipos y sistemas, personal, costes y control de stocks. Permite una parametrización completa del programa de forma sencilla y versátil (Renove Tecnología S.L, 2009-2014).

1.4. Valoración de los sistemas estudiados

La valoración de los sistemas estudiados se realizó a través de varios indicadores relacionados a las funciones que debe cumplir un sistema MCC. Estas funciones se identificaron a partir del estándar SAE-JA1011 que establece los criterios que debe cumplir un proceso MCC y el SAE-JA1012 que es la guía para la aplicación de la metodología MCC. En la tabla 1 se muestra este análisis.

Tabla 1. Análisis de las herramientas MCC

Indicadores/ Sistemas	Relex	Item Toolkit	RCMO	RCM++	RENOVEGEM	PMX pro Plus
Tratamiento de funciones	Sí	Sí	Sí	Sí	No	No
Tratamiento de fallos funcionales	Sí	Sí	Sí	Sí	No	No
Tratamiento de modos de fallos	Sí	Sí	Sí	Sí	No	No
Tratamiento del efecto de fallo	Sí	Sí	Sí	Sí	No	No
Políticas y estrategias MCC	No	No	Sí	Sí	No	No
Exportar/Importar datos	Sí	Sí	No	Sí	No	No

Licencia privativa	Sí	Sí	Sí	Sí	Sí	Sí

Tomando en consideración la información que se muestra en la tabla 1 se concluye que de los sistemas estudiados el RCM++ es la herramienta que mejor se ajusta a la metodología MCC. Esta solución puede resolver gran parte de los problemas de informatización que necesitan los procesos de mantenimiento en Cuba pero tiene como principal desventaja que su licencia es privativa. Su licencia no se ajusta a las políticas de software libre que impulsa el país, sin contar además los elevados gastos anuales por licencias de uso y mantenimiento que implicaría su elección. En una situación casi muy similar se encuentra el RCMO pero este además no permite importar o exportar datos debido a su alto nivel de integración con SAP.

Por otro lado los sistemas Relex e Ítem Toolkit, además de poseer licencia privativa, no soportan metodologías de MCC, pero sí realizan análisis FMEA.

El Toolkit se estudió en mayor profundidad debido a que la herramienta fue entregada por el cliente como apoyo al desarrollo de la propuesta de solución. De esta herramienta salieron como propuestas candidatas la gestión de los nomencladores: especialidad e intervalo, la gestión de las funciones, fallos funcionales, modos de fallo, efectos de fallo y la hoja de decisión.

Los sistemas Renovegem y PMX pro Plus son sistemas para la gestión de mantenimiento pero que no contienen módulo para el MCC. Se analizaron porque son de distribución gratuita pero después de probarlos no se encontraron funciones afines al análisis FMEA o a la metodología MCC.

La información mostrada en la tabla 1 manifiesta la necesidad que hay en Cuba de desarrollar una herramienta propia pues las que hoy existen en el mundo no satisfacen las necesidades del cliente.

1.5. Modelo de desarrollo de software

Para el desarrollo de cualquier producto de software se realizan una serie de tareas entre la idea inicial y el producto final, un modelo de desarrollo establece el orden en el que se harán las cosas en el proyecto, provee de requisitos de entrada y de salida para cada una de las actividades. Dado que cada proyecto es único, no existe un modelo que se aplique al 100% a

todos los proyectos de una organización. Una organización puede contar con uno o más modelos de desarrollo para ser utilizados dependiendo del tipo de proyecto (Pérez Cortés, 2010).

En el caso del trabajo desarrollado se utilizó el Modelo de Desarrollo de Software v1.1 del CEIGE. Este modelo cuenta con 2 fases: Inicio o Estudio preliminar y Desarrollo. La fase de Inicio o Estudio preliminar realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto. En la fase de Desarrollo se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. En esta fase se ejecutan las disciplinas Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de liberación (CEIGE, 2013).

1.6. Tecnologías y herramientas utilizadas

Una vez conocidas las características del modelo de desarrollo que guiaron el proceso de desarrollo de software para este trabajo, se hace necesario describir las tecnologías y herramientas definidas por los arquitectos del proyecto para el desarrollo de la aplicación.

1.6.1. Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es un lenguaje basado en la metodología orientada a objetos creado por el Object Management Group (OMG) con la intención de convertirse en un estándar para el modelado de aplicaciones software, tanto a efectos de servir de base de la ingeniería software, servir de método de intercambio de ideas entre los equipos que desarrollan software y como base a las herramientas que utilizan (Drake, y otros, 2009).

UML es un lenguaje de modelado universal con capacidad de describir cualquier tipo de sistema, sin embargo, su desarrollo semántico ha sido específicamente desarrollado para el modelado de aplicaciones software. El objetivo de UML es permitir al diseñador formular un modelo del sistema y por modelo se entiende la definición de un conjunto de abstracciones

semánticas que permiten describir simbólicamente los conceptos estructurales, de comportamiento y funcionales que se proponen para el sistema (Drake, y otros, 2009).

UML es un lenguaje que es soportado por Visual Paradigm, debido a su característica de descripción simbólica de los conceptos.

1.6.2. Visual Paradigm 8.0

Visual Paradigm es una herramienta de Ingeniería de Software Asistida por Computación (CASE, por sus siglas en inglés) que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. Visual Paradigm se caracteriza por ser un sistema multiplataforma, distribuido bajo una licencia comercial y gratuita. Su diseño se centra en los casos de usos y enfocado al negocio que generan un software de mayor calidad además de ser capaz de ofrecer ingeniería directa e inversa (Piñero González, y otros, 2013).

1.6.3. Patrón arquitectónico Modelo-Vista-Controlador

Para conformar el esquema fundamental del software se utilizó el patrón Modelo-Vista-Controlador (MVC). El MVC separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres partes diferentes. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). La vista maneja la visualización de la información. El controlador interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado. Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual (Ballester Jiménez, 2012).

1.6.4. AJAX

Se utilizó AJAX, acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML (JavaScript asíncrono y XML[8]), porque es una técnica de desarrollo web para dotar a las aplicaciones de interactividad. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página (Vallejo Moreno, y otros, 2012).

1.6.5. Lenguaje de programación JavaScript

JavaScript es un lenguaje de programación interpretado, que se utiliza principalmente en el lado del cliente. Se define como orientado a objetos, basado en prototipos, imperativo y dinámico. No obstante, se pueden realizar implementaciones en JavaScript en el lado del servidor utilizando por ejemplo Node.js. Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las aplicaciones web. El DOM (Document Object Model) es una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo sobre como pueden combinarse dichos objetos y una interfaz estándar para acceder a ellos y manipularlos. Por ello, a través del DOM, es como las aplicaciones web pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML de manera dinámica utilizando el lenguaje JavaScript (Hernández Carballido, y otros, 2013).

1.6.6. Lenguaje de programación PHP 5

PHP es un acrónimo recursivo que significa **P**HP **H**ypertext **P**re-processor. Según el sitio PHP.net, es un lenguaje de script incrustado dentro del HTML[9]. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta

[8] eXtensible Markup Language

[9] HyperText Markup Language

del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas (Torres, y otros, 2012). El marco de trabajo Sauxe, en el que se implentará la solución, está desarrollado sobre este lenguaje.

1.6.7. Marco de trabajo Sauxe 2.2

Para el desarrollo de este trabajo se utilizará el marco de trabajo Sauxe en su versión 2.2. Este marco de trabajo fue desarrollado en el CEIGE como plataforma tecnológica para sus propios proyectos. El marco de trabajo Sauxe contiene un conjunto de componentes reutilizables que proveen la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor agilidad en el proceso de desarrollo. Está basado en Zend Framework[10], utiliza como marco de trabajo de persistencia a Doctrine y ExtJS en la capa de presentación, por la gran gama de componentes que se pueden reutilizar y para mostrarle al usuario una interfaz más amigable (Velazquez Osorio, y otros, 2013).

1.6.8. ExtJs 2.2

Como marco de trabajo de presentación del lado del cliente se utilizó ExtJs 2.2. Ofrece múltiples opciones para trabajar con las validaciones así como el manejo de errores en el cliente. Es multiplataforma y adaptable a diferentes marcos de trabajo. Cuenta con dos licencias: comercial y open source (Capdevila Camacho, y otros, 2013).

ExtJs ofrece al desarrollador un gran conjunto de widgets (componentes como por ejemplos grids, ventanas de diálogo, etc) plenamente integrados y una API (Application Program Interface) para conseguir interfaces web más dinámicas e interactivas con el usuario (Correa Lozano, 2010).

Una de las grandes ventajas de ExtJs es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno. Además la ventaja flotante que provee ExtJs es excelente por la forma en que funciona. Al moverla o

[10] Término del inglés para referirse al marco de trabajo.

redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le da una ventaja tremenda frente a otros (Rivera Calero, y otros, 2012).

1.6.9. Zend Framework 1.11

Se trata de un marco de trabajo para el desarrollo de aplicaciones y servicios web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Este marco de trabajo está formado por una serie de métodos estáticos y componentes que usarán estos métodos (González Brito, y otros, 2013).

Por las bondades que ofrece se integró al marco de trabajo Sauxe, permitiendo:

- Contar con una amplia documentación para el desarrollo.
- El desarrollo es completamente orientado a objetos pues está basado en PHP5.
- Implementación del patrón modelo-vista-controlador.
- Bajo acoplamiento de sus componentes por lo que se pueden usar de forma independiente.
- Contar con adaptadores para bases de datos diferentes (Guerra López, 2012).

1.6.10. Doctrine Framework 1.2.2

Es un Mapeador de Objetos Relacionales (ORM) para PHP5 que se encuentra en la parte superior de una capa de abstracción de base de datos de gran alcance. Una de sus principales características es la posibilidad de escribir consultas de base de datos en un lenguaje orientado a objetos de propiedad SQL llamada Doctrine Query Lenguaje, inspirado en Hibernates HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad sin necesidad de duplicar código innecesario (Santiesteban Pérez, y otros, 2011).

1.6.11. Sistema de gestión integral de seguridad Acaxia

Para la seguridad se utilizará el sistema de gestión integral de seguridad Acaxia, desarrollado en el CEIGE. Este sistema está integrado al Sauxe, marco de trabajo utilizado, gestionando la seguridad de forma centralizada.

El modelo de seguridad del sistema, se caracteriza por gestionar los conceptos rol, permiso y recurso, garantiza que determinados roles tengan las atribuciones o permisos que le

corresponden solamente sobre recursos específicos. El componente provee además un sistema de trazas que registra la información de las acciones que realiza el usuario en todo momento (Nuñez Sanz, y otros, 2013).

Un aspecto a tener en cuenta que incorpora el sistema es la administración de conexiones donde se restringe el acceso de los sistemas y usuarios a los servidores, bases de datos, esquemas y otras estructuras de datos, con esto no se tendrá que almacenar las configuraciones de conexión en ficheros físicos que pueden ser objetos de ataques (Gómez Baryolo, 2011).

1.6.12. Servidor de aplicaciones Apache 2.2

El servidor web Apache es un servidor web gratuito desarrollado por el Proyecto Servidor Apache. Según los datos publicados por Netcraft, es hoy en día más usado que todos los demás servidores web juntos. Entre sus características que lo hacen tan popular cuenta que su licencia es de código abierto del tipo BSD[11] que permite el uso comercial y no comercial. Cuenta con una talentosa comunidad de desarrolladores siguiendo un proceso abierto de desarrollo. Los usuarios de Apache pueden adicionar fácilmente funcionalidad a sus ambientes específicos. Apache trabaja sobre todas las versiones recientes de UNIX y Linux, Windows, BeOs, mainframes. Es robusto y seguro (Márquez Díaz, y otros, 2013).

1.6.13. Servidor de Base de Datos PostgreSQL 9.1

PostgreSQL es un servidor de base de datos objeto-relacional libre, liberado bajo licencia BSD. Es una alternativa a otros sistemas de bases de datos de código abierto, como MySQL, Firebird y MaxDB. Concurre en el mercado de sistemas gestores de bases de datos (SGBD) con sistemas propietarios, tales como Oracle, MS SQL y DB2 (Persegona, y otros, 2012).

PostgreSQL por ser el SGBD de código abierto más avanzado del mundo ya que soporta la gran mayoría de las transacciones SQL, control concurrente, ofrece modernas características como consultas complejas, disparadores, vistas, integridad transaccional y permite agregar extensiones de tipo de datos, funciones, operadores y lenguajes procedurales. A partir de la versión 9.1, PostgreSQL brinda facilidades para que los usuarios puedan crear, cargar,

[11] Berkeley Software Distribution (en español, Distribución de Software Berkeley)

actualizar y administrar extensiones utilizando el objeto de base de datos EXTENSION (Robles Aranda, y otros, 2013).

1.6.14. Navegador Firefox 24.0 o superior

Firefox es un navegador de Internet con interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios. Se emplea por las ventajas que ofrece:

- Es de código libre y multiplataforma.
- Posee función de auto completamiento en la barra de navegación, lo cual facilita la navegación por internet.
- Guarda la contraseña para entrar a un sitio determinado.
- Permite la restauración de las ventanas y pestañas que se estuvieran usando antes de cerrar el navegador.
- Seguridad avanzada en la navegación pues permite la identificación de sitios web de forma instantánea, posibilita navegar de forma privada y sin registro de lo accedido en internet. (Guerra López, 2012).

1.6.15. Entorno de desarrollo integrado Netbeans

El entorno integrado de desarrollo elegido fue el *Netbeans* 8.0 debido a que es un entorno de desarrollo integrado libre. Soporta lenguajes como PHP y C con C++. Más que un entorno de desarrollo, Netbeans es una comunidad en crecimiento compuesta por cientos de socios en el mundo. Existe un gran número de módulos que permiten su extensión y las aplicaciones creadas con la plataforma pueden ser construidas a partir de módulos, los cuales pueden ser añadidos para extender las funcionalidades de las aplicaciones. Entre las características más relevantes, resalta la detección y reparación de errores, capacidad de realizar sugerencias de cambios y reportar al usuario de posibles pérdidas de información. Incluye un depurador y un selector de vistas que posibilita navegar por las líneas de código de un programa, siendo esto la base de la detección de errores en la lógica del programa (Piñero González, y otros, 2013).

1.6.16. Biblioteca TCPDF v2.1

El proyecto creador de la biblioteca TCPDF fue iniciado en 2002; es ahora uno de los proyectos Open Source más activos del mundo, que se utiliza a diario por millones de usuarios y se incluye en miles de CMS y aplicaciones web. TCPDF es una biblioteca PHP utilizada para generar documentos PDF (Asuni, 2013). Entre las potencialidades que ofrece se encuentran:

- No se necesitan bibliotecas externas para las funciones básicas.
- Admite los formatos JPEG, PNG y SVG[12].
- Encabezado y pie de página de gestión automática.
- Documento cifrado de hasta 256 bits y certificados de firma digital.
- Utilización del modo de columnas múltiples, regiones de la página de no escritura, marcadores y tabla de contenido, la partición de palabras de texto, salto de página automático, salto de línea y las alineaciones de texto, incluyendo la justificación; grupos automáticos de numeración de página, mover y eliminar páginas (Asuni, 2013).

1.7. Pruebas de software

La prueba es una actividad fundamental de ingeniería, y es responsable de una porción significativa de los costos del desarrollo y el mantenimiento del software (Serna, y otros, 2012).
Realizar pruebas al software durante el proceso de desarrollo es garantía para la puesta en explotación de los sistemas; además de corroborar el grado de confiabilidad antes de ser entregado a sus usuarios finales; disminuyendo los defectos al utilizar técnicas apropiadas que posibiliten procesos de desarrollo de software eficaz, minimizando tiempo y costo (Pentón Saucedo, y otros, 2012).
Para realizarle las pruebas al módulo MCC se tuvo en cuenta que los investigadores del tema han desarrollado dos técnicas fundamentales: las pruebas de caja blanca o caja de cristal y la prueba de caja negra o prueba de comportamiento (Pressman, 2002).

[12] Gráficos vectoriales escalables, del inglés *Scalable Vector Graphics*.

Pruebas de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que utiliza la estructura de control del diseño procedimental para obtener los casos de prueba (Pressman, 2002).

La prueba de caja blanca aplicada a la solución desarrollada es la prueba del camino básico, que permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (Pressman, 2002).

Pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2002).

El método de caja negra que se utiliza para asegurar la calidad de la aplicación desarrollada es el de partición equivalente.

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico (Pressman, 2002).

1.8. Conclusiones parciales

Se realizó un estudio de los principales conceptos relacionados con el MCC, recopilando información de utilidad para comprender el dominio del negocio. Se estudiaron sistemas MCC desarrollados en el mundo, que aportaron un conjunto de elementos que cumplen con las características deseadas del módulo que se propone. Se describieron las tecnologías y herramientas propuestas permitiendo sentar las bases para el desarrollo del componente MCC.

CAPÍTULO 2 PROPUESTA DE SOLUCIÓN

2.1 Introducción

En este capítulo se describe la propuesta de solución partiendo del modelado del dominio, en el que se obtiene el modelo conceptual. Se describen técnicas utilizadas para obtener el modelo de dominio, para identificar y validar los requisitos. Se identifican y describen los requisitos funcionales y no funcionales. Seguidamente se realiza la validación de los requisitos, la matriz de trazabilidad para mostrar la relación conceptos-requisitos y se realiza el modelo de datos mediante un diagrama de entidad relación. Se realizan los diagramas de clases del diseño y la descripción de las clases del diseño, finalizando con la aplicación de métricas para evaluar el diseño propuesto.

2.2 Requerimientos de información de mantenimiento

Los requerimientos de información de mantenimiento se determinaron a partir de los criterios de especialistas del CEIM. En los encuentros realizados definieron que la primera necesidad era la informatización de la Hoja de decisión, a partir de la cual se deriva el trabajo con los activos y el análisis MCC. El análisis necesita de la gestión de las funciones, fallos funcionales, modos de fallo y efectos de fallo para crear su estructura.

Asociado a la gestión de la Hoja de decisión, se realizan varios reportes como el de la Hoja de información por análisis, listado de activos con último análisis realizado, listado de activos, análisis asociado a un activo y Hoja de decisión por análisis. Esta información, según los ingenieros de mantenimiento, les permite acceder con mayor rapidez a la información exacta que necesitan para trabajar.

Tomando como base los requerimientos de información de mantenimiento, se comenzó con el modelado del dominio para identificar los conceptos de la propuesta de solución.

2.3 Modelado del dominio

Para comenzar el desarrollo de la disciplina de Modelado de negocio, se hizo un estudio de la documentación entregada por el cliente, que consiste en tesis de maestrías y doctorados en las que definen el MCC. A partir del estudio se realizó el glosario de términos que permitió ir anotando la terminología de difícil manejo.

Para modelar el negocio se utilizó el modelado de dominio, por no existir en el negocio procesos claramente definidos. Para el modelado se utilizaron las reglas de UML por su facilidad de uso y comprensión.

Modelo conceptual

El modelo de dominio (conceptual) se utilizó para capturar y expresar el entendimiento alcanzado acerca del negocio. Es un modelo sencillo en el que se identificaron las clases u objetos del dominio, así como sus atributos y relaciones. El modelo se puede apreciar a continuación:

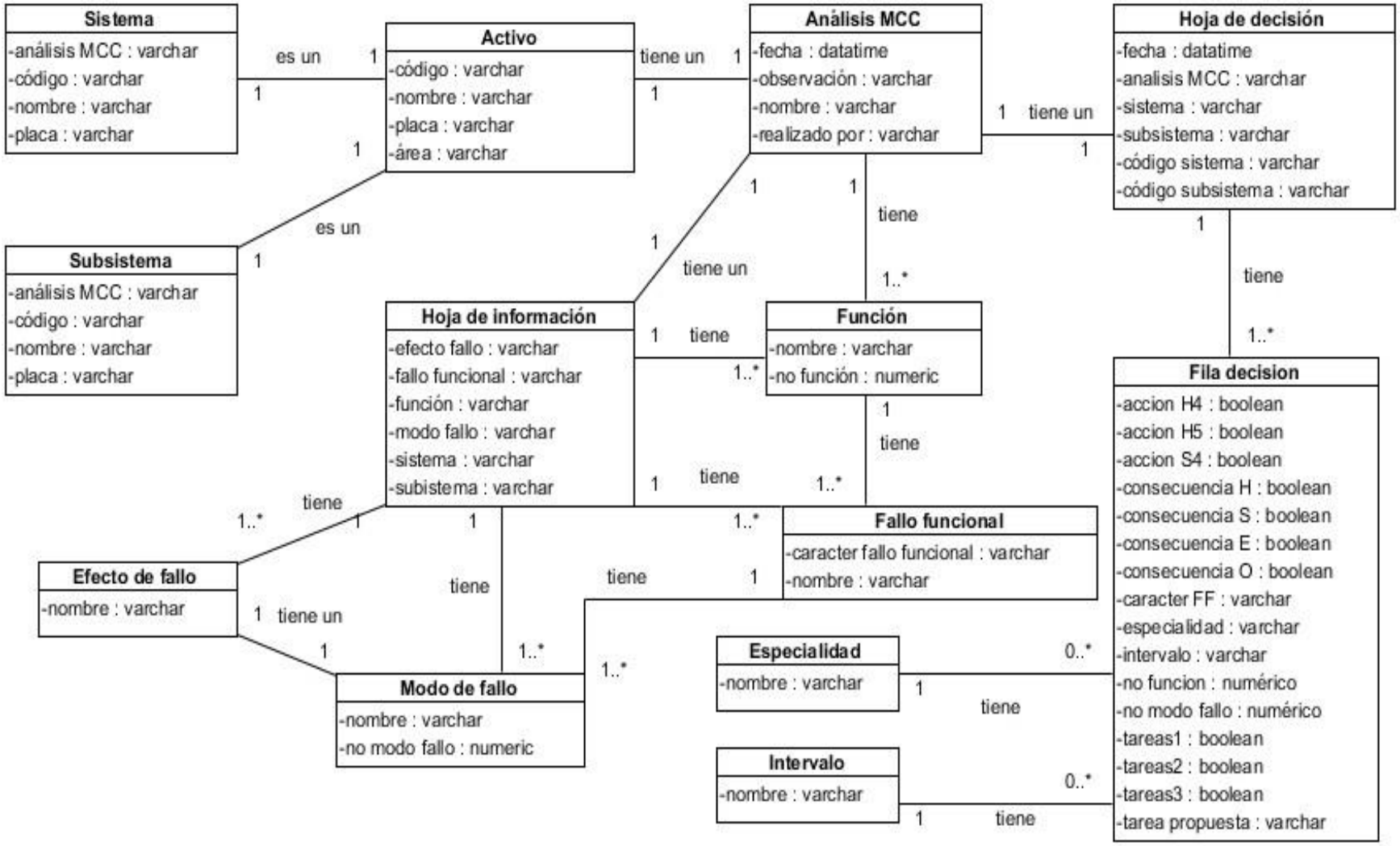

Figura 2. Modelo conceptual

Técnicas utilizadas para el modelado del dominio

- **Tormenta de ideas**

Participaron en la aplicación de esta técnica especialistas del ISPJAE, los tutores y el jefe de departamento. Se realizaron debates en el grupo escuchando la opinión de cada miembro y se arribaron a conclusiones, donde se refleja el concepto y sus relaciones.

- **Utilización de una lista de categorías de clases conceptuales**

Se comienza la creación de un modelo del dominio haciendo una lista de clases conceptuales candidatas (Craig, 2003). Para el desarrollo de esta técnica se tuvo en cuenta un grupo de clases conceptuales obtenidas en la tormenta de ideas, permitiendo identificar otras y refinar las existentes.

- **Identificación de frases nominales**

Otra técnica útil (debido a su simplicidad) es el análisis lingüístico: identificar los nombres y frases nominales en las descripciones textuales del dominio, y considerarlos como clases conceptuales o atributos candidatos (Craig, 2003).

Esta técnica permitió hacer un análisis detallado de la información entregada por el cliente. Se identificaron los nombres y frases nominales en las descripciones textuales de la documentación y se incorporaron nuevas clases conceptuales o atributos candidatos.

2.4 Requisitos

Una vez conocidas las técnicas utilizadas para el modelado del dominio fue posible pasar a identificar y describir los requisitos funcionales del sistema. Se identificaron los conceptos y sus atributos, permitiendo la utilización de técnicas para comenzar la captura de requisitos.

Técnicas para la captura de requisitos

- **Tormenta de ideas**

Es una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la acumulación de ideas o información sin evaluar las mismas, donde un miembro del grupo asume el rol de moderador (Escalona, y otros, 2002).

En la aplicación de esta técnica participaron especialistas del ISPJAE, los tutores y el jefe de departamento; se realizaron debates, escuchando el criterio de cada uno de los participantes para lograr el entendimiento común de cada uno de los requisitos capturados, permitiendo determinar los posibles cambios o errores en dicha captura.

- **Análisis de documentación**

Cuando existen sistemas anteriores similares, estos se convierten en fuente de información respecto a los procedimientos de las operaciones. Además, existen otros tipos de documentos en las organizaciones de las cuales es posible obtener información sobre prácticas corrientes de las actividades que el sistema debe hacer y que serán útiles al escribir una especificación de requisitos (Aranda Amaolo, 2009).

Esta técnica se utilizó para analizar distintos sistemas ya desarrollados, relacionados con el mantenimiento. En ellos se observaron las interfaces de usuario, el tipo de información que se maneja y cómo es manejada, permitiendo seleccionar características y funcionalidades que se incluyeron al sistema en desarrollo.

Requisitos funcionales

Los requisitos funcionales son las condiciones que el software para cumplir con las necesidades del cliente los cuales se identificaron a partir de las técnicas definidas. Cada requisito funcional tiene asociada una descripción textual y en los casos requeridos se acompaña de prototipos de interfaz de usuario. A continuación se muestran los requisitos identificados, algunos de ellos contenidos en agrupaciones:

RF 1. Gestionar especialidad
- RF 1.1 Adicionar especialidad
- RF 1.2 Modificar especialidad
- RF 1.3 Buscar especialidad
- RF 1.4 Listar especialidad

RF 2. Asociar especialidad a la fila de decisión
RF 3. Listar intervalo
RF 4. Buscar intervalo
RF 5. Listar activo
RF 6. Buscar activo
RF 7. Asociar análisis MCC al activo
RF 8. Gestionar análisis MCC
- RF 8.1 RF 8.1 Adicionar análisis MCC
- RF 8.2 Modificar análisis MCC

RF 8.3 Buscar análisis MCC

RF 8.4 Listar análisis MCC

RF 9. Asociar función al análisis MCC

RF 10. Gestionar función

RF 9.1 Adicionar función

RF 9.2 Modificar función

RF 9.3 Eliminar función

RF 9.4 Listar función

RF 11. Asociar fallo funcional a la función

RF 12. Gestionar fallo funcional

RF 11.1 Adicionar fallo funcional

RF 11.2 Modificar fallo funcional

RF 11.3 Eliminar fallo funcional

RF 11.4 Listar fallo funcional

RF 13. Asociar modo de fallo al fallo funcional

RF 14. Gestionar modo de fallo

RF 14.1 Adicionar modo de fallo

RF 14.2 Modificar modo de fallo

RF 14.3 Eliminar modo de fallo

RF 14.4 Listar modo de fallo

RF 15. Asociar efecto de fallo al modo de fallo

RF 16. Gestionar hoja de decisión

RF 16.1 Adicionar hoja de decisión

RF 16.2. Modificar hoja de decisión

RF 16.3. Buscar hoja de decisión

RF 16.4. Listar hoja de decisión

RF 17. Gestionar fila de decisión

RF 17.1 Adicionar fila de decisión

RF 17.2 Modificar fila de decisión

RF 17.3 Eliminar fila de decisión

RF 17.4 Listar fila de decisión

RF 17.5 Buscar fila de decisión

RF 18. Asociar fila de decisión a la hoja de decisión

RF 19. Recuperaciones

RF 19.1. Generar listado de activos

RF 19.2. Generar listado de activos con último análisis MCC realizado

RF 19.3. Generar hoja de información por análisis MCC

RF 19.4. Generar hoja de decisión por análisis MCC

RF 19.5. Generar análisis MCC de un activo en un período

Especificación de requisitos funcionales

Al concluir la identificación de los requisitos se procedió con la descripción de los requisitos del cliente. Entre los requisitos más significativos se encuentra la agrupación Gestionar hoja de decisión. A continuación se presenta el requisito Listar hoja de decisión. Para observar el resto de los requisitos se deben consultar los entregables de la investigación en el departamento de Soluciones Empresariales.

Listar hoja de decisión

En el requisito se lista la hoja de decisión que se realiza por cada análisis MCC realizado, con el objetivo de conocer la hoja de decisión asociada al dicho análisis.

Descripción del requisito Listar hoja de decisión

Tabla 2. Descripción del requisito Listar hoja de decisión

Precondiciones	Se ha registrado al menos una hoja de decisión en el sistema.
Flujo de eventos	
Flujo básico Adicionar competencia	
1.	Se insertan los criterios de búsqueda: análisis mcc realizar por fecha
2.	El sistema muestra un listado de las hojas de decisión que cumplen los criterios de búsqueda especificados. Se muestran los siguientes campos por cada hoja de decisión:

	fecha análisis mcc código sistema sistema código subsistema subsistema
3.	Concluye el requisito.
Pos-condiciones	
1	Se registró en el sistema una competencia.
Flujos alternativos	
Flujo alternativo 2.a No existen datos que cumplan con los criterios especificados	
1	El sistema notifica que no existen datos que cumplan con los criterios especificados.

Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual.	
Conceptos	**Hoja de decisión**	**Visibles en la interfaz:** fecha análisis mcc código sistema sistema código subsistema subsistema **Utilizados internamente:** N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

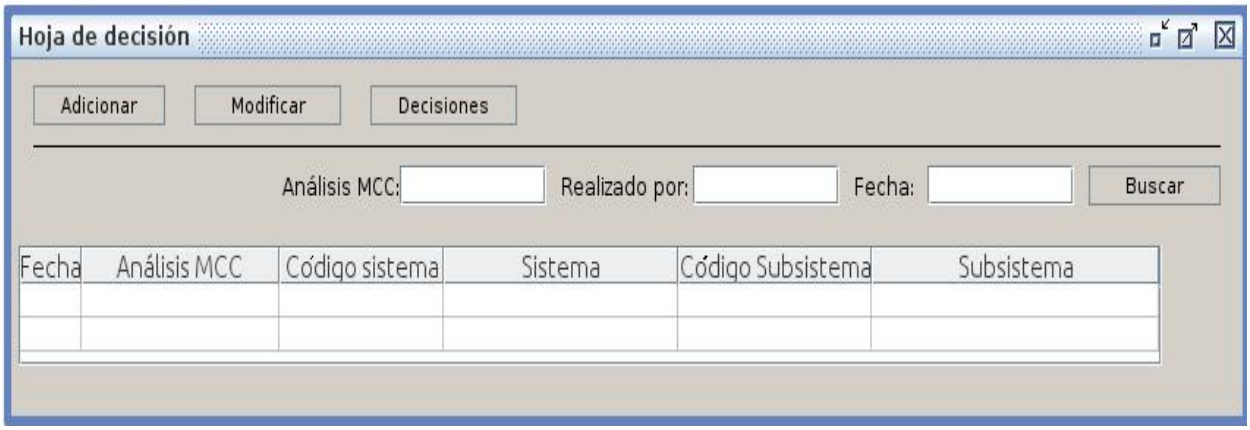

Figura 3 Prototipo de interfaz de usuario del requisito Listar hoja de decisión

Para más información sobre la descripción de requisito Listar hoja de decisión del módulo MCC para un sistema de Confiabilidad Operacional, consultar el documento entregable Especificación de requisitos. Las demás descripciones de los requisitos del módulo MCC para un sistema de Confiabilidad Operacional se encuentran especificados en el documento entregable Especificación de requisitos.

Requisitos no funcionales

Los requisitos no funcionales son propiedades que el software debe cumplir para hacerlo más atractivo, rápido y confiable. Están relacionados con las características que de una u otra forma pueden limitar el sistema, como por ejemplo rendimiento, fiabilidad y seguridad (Sommerville, 2005).

Requisitos no funcionales identificados:

Usabilidad:

- El idioma de todas las interfaces de la aplicación será el español.
- El sistema será consistente en el uso de abreviaturas, usará la misma abreviación siempre para la misma palabra y nunca en un elemento de selección o menú.
- Los flujos de navegación para la gestión de cualquier concepto del negocio no excederán las 3 interfaces.

Fiabilidad:

- El sistema tendrá un respaldo de la información en un servidor aparte del concebido para el almacenamiento regular de los datos, permitiendo la recuperación ante la pérdida parcial o total de la información.

Seguridad:

- El sistema manejará la seguridad de acceso y administración de usuarios mediante el otorgamiento de privilegios y roles, asignación de perfiles.
- Se concederá acceso al sistema a partir de un nombre de usuario y una contraseña.
- El sistema concederá acceso a cada usuario autenticado solo a las funciones que le estén permitidas, de acuerdo a la configuración del sistema. Los menús serán generados a partir del propio perfil del usuario.

2.5 Priorización de requisitos

Para la priorización de los requisitos identificados se utilizó la planilla Evaluación de requisitos definida en el expediente de proyecto de la UCI, para la gestión de los proyecto de desarrollo. Siguiendo los criterios de complejidad que en ella se establecen se determinaron que los requisitos de mayor prioridad son aquellos que obtuvieron complejidad Alta. Consultar documento entregable priorización de requisitos.

Criterios de complejidad:

- Diferentes comportamientos: un mismo requisito se comporta de manera diferente ante determinadas situaciones.
- Complejidad por interfaces: el criterio interfaz se le aplica a requisitos que presentan algún tipo de complejidad en su interacción con el elemento humano (formulario e informe).
- Consultas a fuentes de almacenamientos: los requisitos pueden presentar diversidad en la cantidad y complejidad de la interacción con la fuente de datos (base de datos, ficheros, otros).
- Restricciones de validación: complejidad de todas las validaciones que lleve un requisito, tanto las validaciones en el lado del cliente, como en el servidor.
- Grado de reutilización: complejidad de un requisito, para poder ser reutilizado por otros.

- Lógica de negocio: los requisitos pueden presentar diferentes niveles de complejidad para la implementación de la lógica de negocio que contienen; Ej. operaciones, métodos matemáticos, CRUD (Consultar entregable Priorización de requisitos).

La Tabla 3 muestra los resultados arrojados de la evaluación de los requisitos de acuerdo a los criterios de complejidad a los que se hacen referencia anteriormente.

Tabla 3. Resultados de la evaluación de requisitos

Resumen	
Cantidad. de requisitos con complejidad Alta	4
Cantidad. de requisitos con complejidad Media	10
Cantidad. de requisitos con complejidad Baja	31

2.6 Validación de requisitos

La validación de los requisitos se realiza con la finalidad de comprobar que los requerimientos identificados sean precisos, consistentes, realistas, verificables, definan lo que el usuario desea del producto final, que los errores que hayan sido detectados sean corregidos y el resultado del trabajo cumpla con los estándares establecidos para el proceso, el proyecto y el producto. (Pressman, 2002). Para la validación se emplearon las siguientes técnicas:

Las **revisiones técnicas formales**: una vez terminadas las descripciones de los requisitos, se revisaron nuevamente las descripciones para comprobar que no existieran errores en el contenido o malas interpretaciones, información incompleta, inconsistencias y que los requisitos no fueran contradictorios, imposibles o inalcanzables, dando como resultado que fueran aprobados los que estaban descritos de forma correcta, clara y consistente. En estas revisiones a las descripciones de requisitos se identificaron incoherencias, errores de conceptos y ortográficos, falta de algún atributo o poca claridad en algunas de estas descripciones, pero en todos los casos fueron resueltos.

Prototipos: a partir de las especificaciones de requisitos fueron conformados prototipos de interfaz de usuario. Los prototipos validaron que los requerimientos estaban en concordancia con las necesidades plasmadas por los proveedores de requisitos. El empleo de esta técnica

ofreció como resultados que el cliente tuviera una idea de la estructura de la interfaz de usuario y se favoreciera la comunicación con el mismo.

2.7 Administración de los requisitos

La administración de requisitos es una parte esencial para controlar la complejidad, riesgo, alcance del proyecto, y definir los roles y criterios para un software. La administración de requisitos es efectiva para mejorar los flujos de trabajo a través del ciclo de vida del proyecto, desde el diseño de software hasta la estimación de costo (Sommerville, 2005).

Para la administración de los requisitos se utilizó la herramienta Visual Paradigm, para ello se definieron los siguientes elementos de trazabilidad:

- Modelo conceptual
- Requisitos
- Diseños de casos de pruebas

Se realizó el diagrama de requisitos con el objetivo de definir la relación entre los elementos de trazabilidad mencionados anteriormente, representándose las siguientes matrices de trazabilidad:

- Requisitos - Modelo Conceptual
- Requisitos - Diseños de casos de pruebas

En la Figura 4 se muestra la matriz de trazabilidad Requisito-Modelo Conceptual, comprobando que a cada concepto del dominio se le asoció al menos un requisito funcional. Esta matriz permite conocer las relaciones de dependencias de cada una de las clases del dominio de los requerimientos de información con los requisitos funcionales. La realización de la matriz permite estudiar el impacto de posibles cambios en los requisitos a partir de transformaciones que puedan tener los requerimientos de información.

La matriz de trazabilidad Requisito-Diseño de casos de prueba se encuentra en la Figura 22 del capítulo 3.

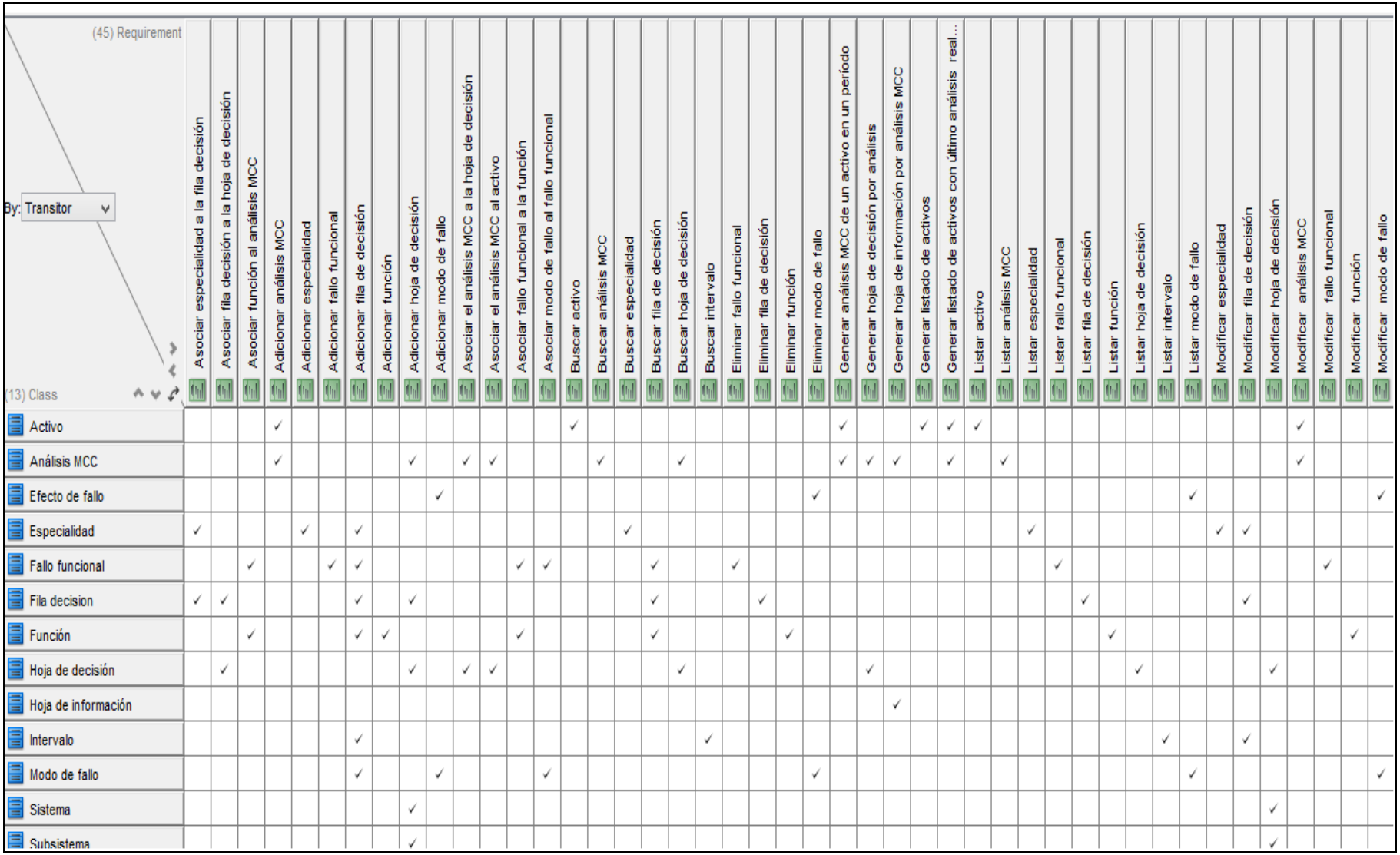

Figura 4 Matriz de trazabilidad Requisito-Modelo Conceptual

2.8 Diseño del sistema

El diseño es el sitio donde manda la creatividad, donde los requisitos del cliente las necesidades de negocio y las consideraciones técnicas se unen en la formulación de un producto o sistema. Crea una representación o modelo del software, pero a diferencia del modelo de análisis, el modelo de diseño proporciona detalles acerca de las estructuras de datos, las arquitecturas, las interfaces y los componentes del software que son necesarios para implementar el sistema (Pressman, 2006).

2.8.1 Mecanismos de diseño

Los mecanismos de diseño se utilizan con el objetivo de simplificar los diagramas de clases. Cada diseñador acorde a sus necesidades establece sus propios mecanismos de diseño, teniendo en cuenta los patrones y estilos seleccionados (Cabrera Pereira., y otros, 2009).

Mecanismo de diseño para las clases controladoras

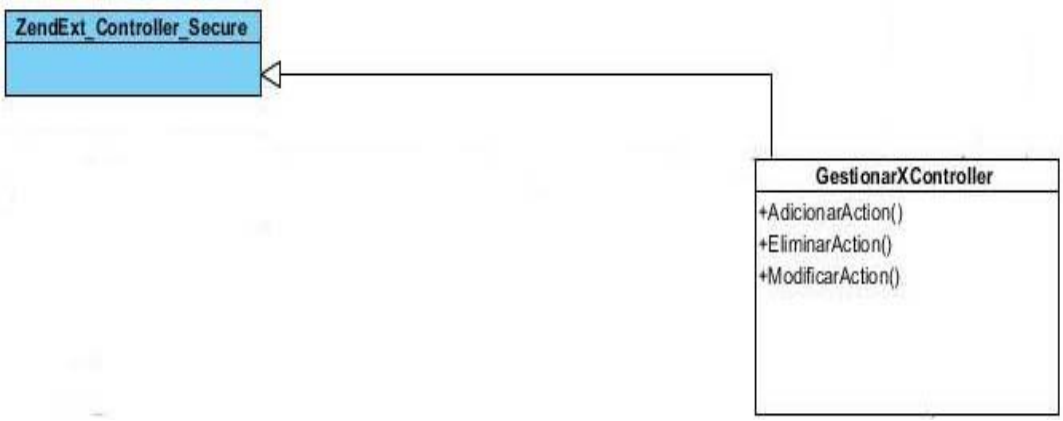

Figura 5 Mecanismo de diseño para las clases controladoras

Todas las clases controladoras definidas en el diseño propuesto heredan de la clase ZendExt_Controller_Secure, ya que en ella se incluyen numerosas funcionalidades comunes en todas las controladoras.

Para el módulo se definieron otros mecanismos de diseño como: inclusión del marco de trabajo ExtJS en las páginas clientes, inclusión del formato UCID para las páginas clientes, inicialización de páginas clientes, construcción y actualización de páginas clientes, mecanismo de diseño para las clases modelos, y mecanismo de diseño para las clases del dominio. Estos

mecanismos se pueden consultar en el artefacto Modelo de diseño del proyecto Mantenimiento UCI.

2.8.2 Diagrama de clases del diseño

El diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. En el caso de las aplicaciones web, el diagrama de clases representa las colaboraciones que ocurren entre las páginas, donde cada página lógica puede ser representada como una clase. Al tratar de utilizar el diagrama de clases tradicional para modelar aplicaciones Web surgen varios problemas, por lo cual los especialistas del Rational plantearon la creación de una extensión al modelo de análisis y diseño que permitiera representar el nivel de abstracción adecuado y la relación con los restantes artefactos de UML (Montes de Oca Hernández., y otros, 2011).

A continuación se muestra un ejemplo de diagrama de clases del diseño:

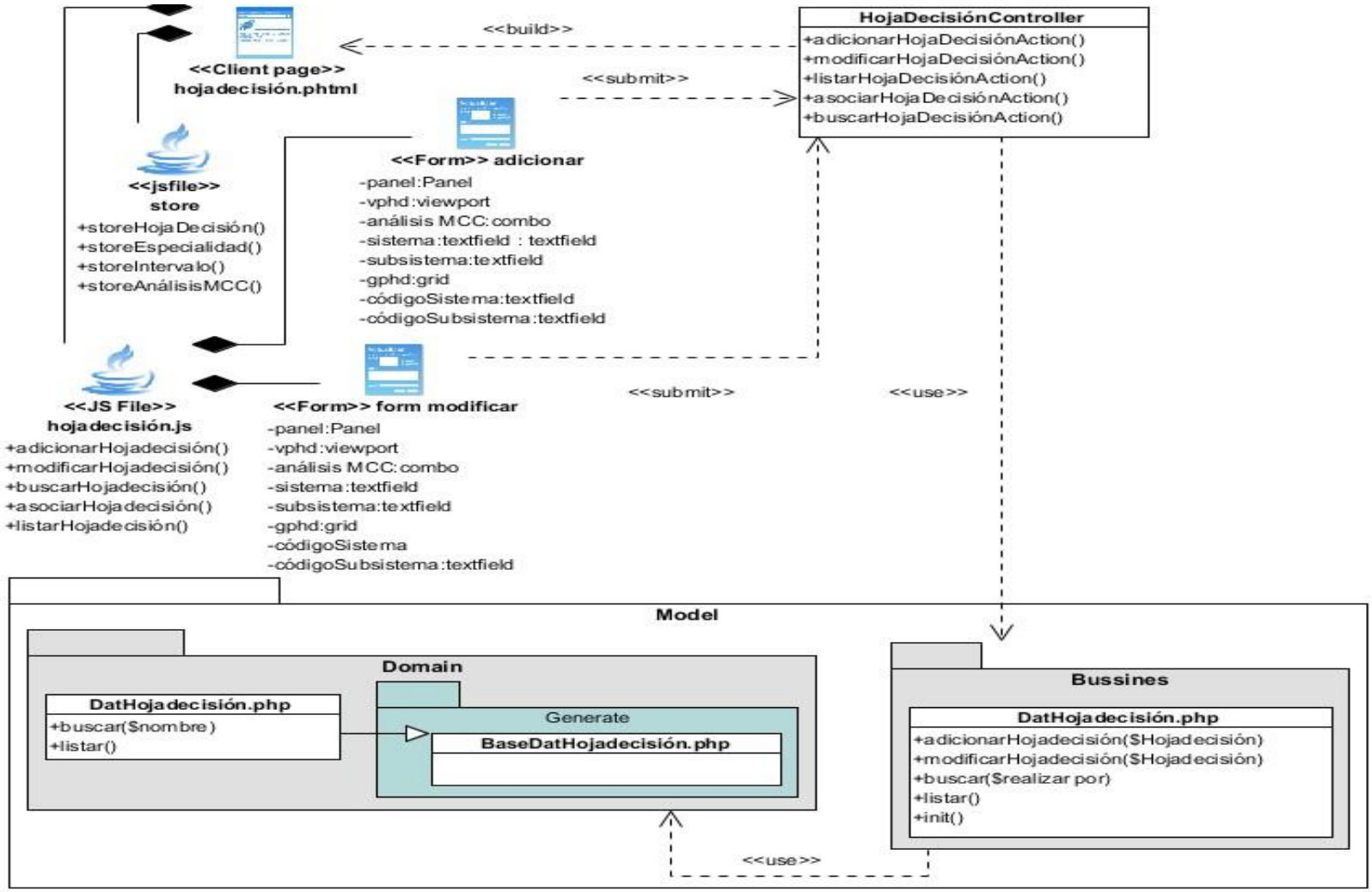

Figura 6 Diagrama de clases del diseño para la agrupación de requisitos Gestionar hoja de decisión

2.8.3 Patrones de diseño

Un patrón de diseño describe un problema que ocurre frecuentemente en el campo de la construcción de software y su respectiva solución; constituye una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular (Sommerville, 2005).

En el diseño del módulo MCC se emplearon los siguientes patrones de diseño.

Patrones GRASP:

Los patrones GRASP[13] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos (Gracia, 2005).

En el diseño del módulo MCC se emplearon los siguientes patrones de diseño.

- **Creador**

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. Un ejemplo más claro de su aplicación se encuentra en los diagramas de clases del diseño realizados, donde las clases “modelo” son las encargadas de la creación de los objetos en el sistema.

- **Controlador**

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. El empleo de este patrón se puede observar en las clases controladoras para las diferentes acciones que se realizan en el módulo MCC. Ejemplo: HojadecisionController.

[13] General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades)

- **Experto**

El patrón Experto se puso en práctica partiendo de que es el principio fundamental de asignación de las responsabilidades durante el diseño, pues siempre se le debe atribuir determinada responsabilidad al experto en la información, por ello el patrón fue usado en todas las clases definidas ya que estas cuentan con la información necesaria para cumplir dicha responsabilidad.

- **Alta cohesión**

En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Su empleo se puede observar en el desarrollo del módulo MCC, pues se le asignaron responsabilidades a las clases de forma tal que cada una se encarga de realizar solamente las funciones que estén en correspondencia con su responsabilidad.

- **Bajo acoplamiento**

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas, por lo tanto una clase con bajo acoplamiento no depende de muchas otras. Este patrón soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. Este patrón se utilizó al asignar una responsabilidad de modo que la misma no incremente la fuerza con que están conectadas entre sí las clases, posibilitando así una mayor reutilización y minimizando el impacto que podría ocasionar la necesidad de eliminar una clase.

2.8.4 Modelo de datos

El modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Describe la representación lógica y física de los datos persistentes.

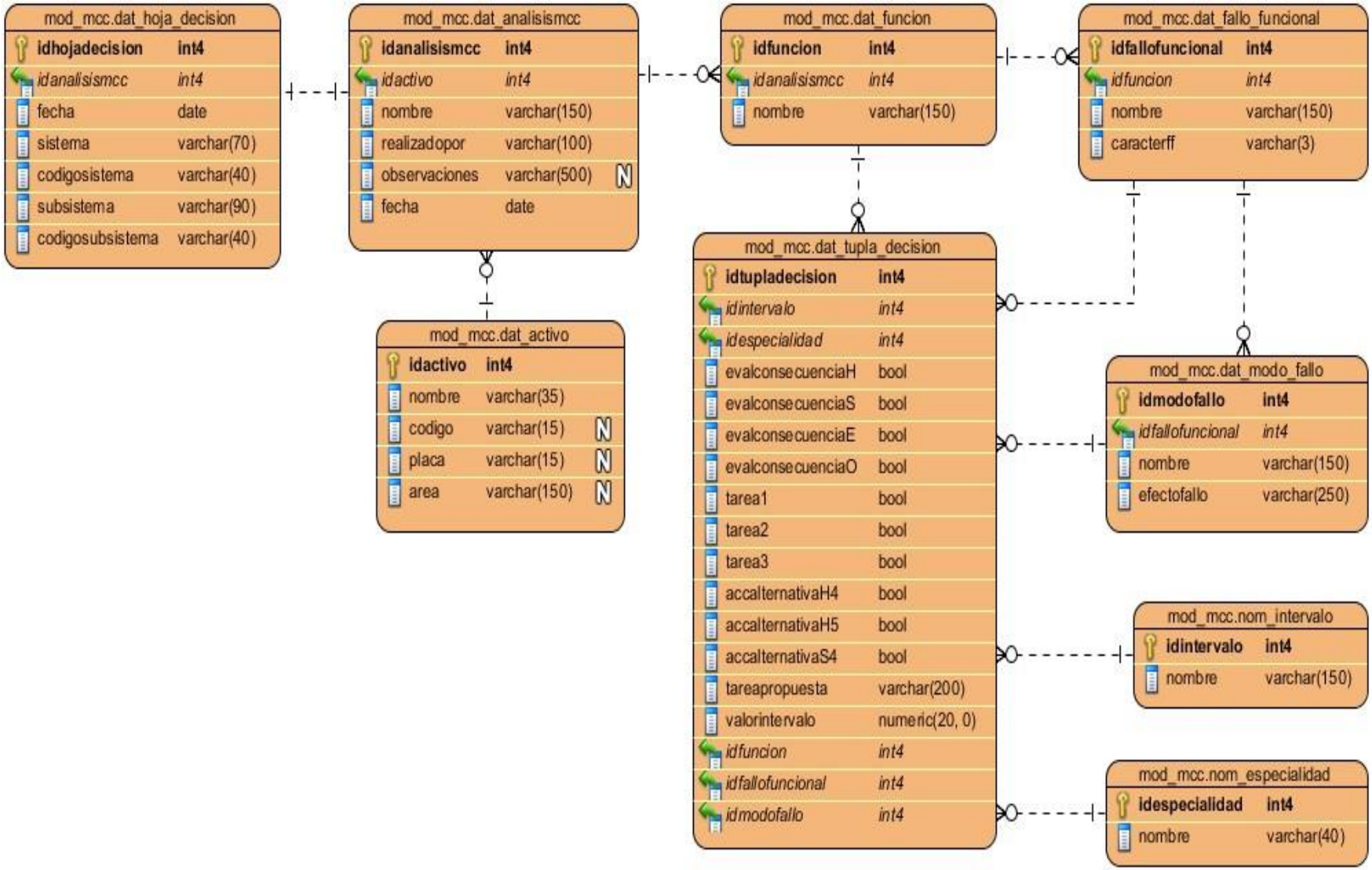

Figura 7 Diagrama Entidad Relación

2.9 Métricas para evaluar el diseño propuesto

Las métricas del software son una medida cuantitativa de evaluar la calidad de los atributos internos de un sistema. Se emplean con el objetivo de llevar el control de la calidad del producto que se está desarrollando, evaluar la efectividad del proceso y mejorar la calidad del trabajo. Las métricas proporcionan los conocimientos necesarios para crear modelos efectivos de análisis y diseño, un código sólido y pruebas exhaustivas (Pressman, 2006).

Una vez realizado el diseño del módulo MCC, se dio paso a su evaluación, para esto se utilizaron las métricas orientadas a objetos, específicamente a las clases, ya que las medidas y métricas para una clase individual, la jerarquía y las colaboraciones de estas permiten al ingeniero de software evaluar la calidad del diseño propuesto.

Métricas orientadas a objetos propuestas por Lorenz y Kidd:

En su libro sobre métricas, Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías, cada una con un diseño al nivel de componentes: tamaño, herencia, valores internos y valores externos. Las orientadas al tamaño aplicadas a una clase orientada a objetos se centran en el conteo de atributos y operaciones de una clase individual y los valores promedio para el sistema orientado a objetos como un todo (Pressman, 2006). Estas métricas se han introducido para ayudar a un ingeniero del software a usar el análisis cuantitativo para evaluar la calidad en el diseño antes de que un sistema se construya.

Las métricas empleadas fueron Tamaño operacional de clase (TOC) para lo cual se tuvo en cuenta la cantidad de procedimientos de las clases identificadas, luego se realizó el cálculo del promedio de los procedimientos y mediante un criterio se obtuvo las categorías (baja, media, alta) para la Responsabilidad, Complejidad y Reutilización. La otra métrica utilizada fue las Relaciones de clases (RC), para aplicarlas se tuvo en cuenta el Número de dependencias entre dichas clases, se calculó del promedio de las dependencias usando un criterio obteniendo las categorías siguientes (baja, media, alta) para el Acoplamiento, Complejidad del mantenimiento, Reutilización y Cantidad de pruebas.

Tamaño Operacional de Clase (TOC): se refiere al número de métodos pertenecientes a una clase. Está determinado por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

Tabla 4. Tamaño operacional de clase (TOC)

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Tabla 5. Rango de valores para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC

	Categoría	Criterio
Responsabilidad	Baja	< =Prom. (5.1)
	Media	Entre Prom. Y 2* Prom
	Alta	> 2* Prom
Complejidad de implementación	Baja	< =Prom
	Media	Entre Prom. y 2* Prom
	Alta	> 2* Prom
Reutilización	Baja	> 2*Prom
	Media	Entre Prom. y 2* Prom
	Alta	<= Prom

Tabla 6. Umbrales para el TOC

Tamaño operacional de la clase	Criterio
Pequeña	<= Prom.(5.1)
Media	Entre Prom. y 2 * Prom.
Grande	> 2* Prom.

Resultados de la evaluación de la métrica Tamaño Operacional de Clase (TOC)

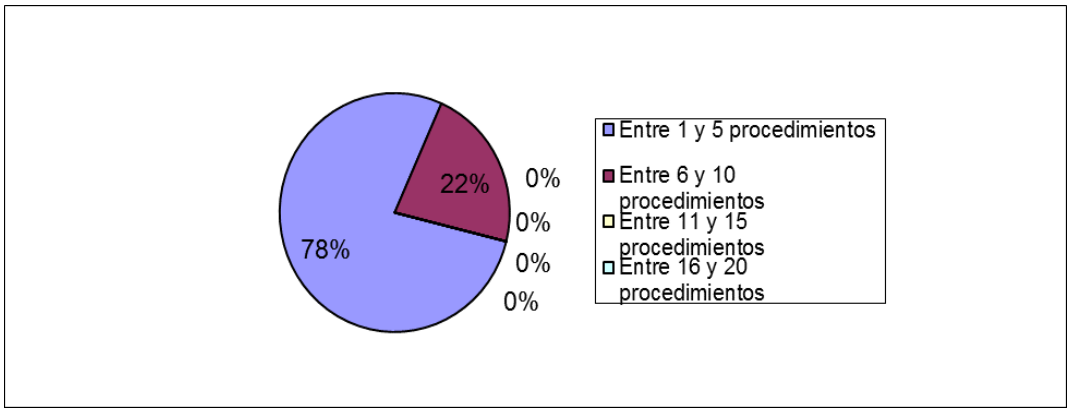

Figura 8 Representación en % de los resultados obtenidos por la aplicación de la métrica (TOC), agrupados en intervalos para la cantidad de procedimientos por clases.

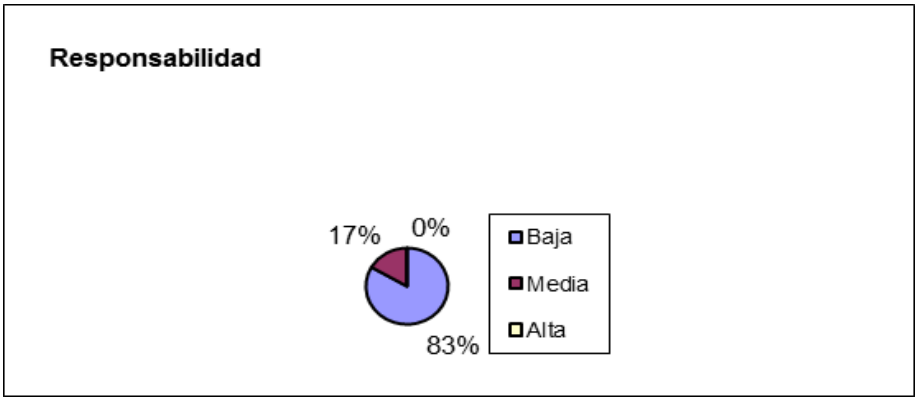

Figura 9 Representación en % de los resultados obtenidos por la aplicación de la métrica (TOC), agrupados en intervalos para el atributo Responsabilidad.

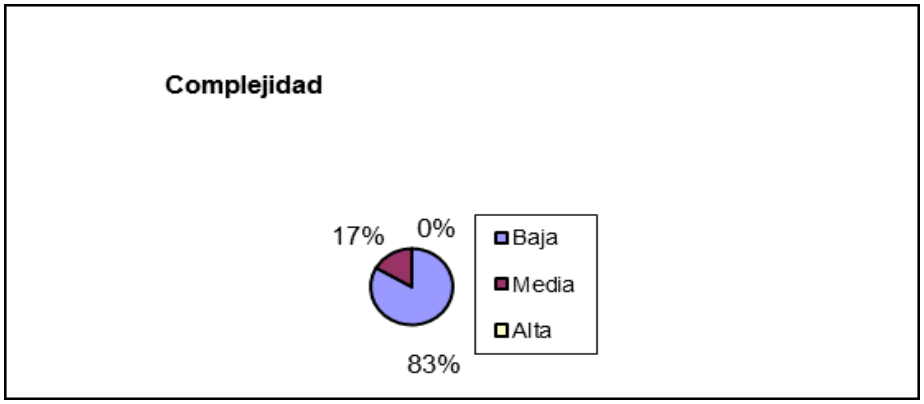

Figura 10 Representación en % de los resultados obtenidos por la aplicación de la métrica (TOC), agrupados en intervalos para el atributo Complejidad.

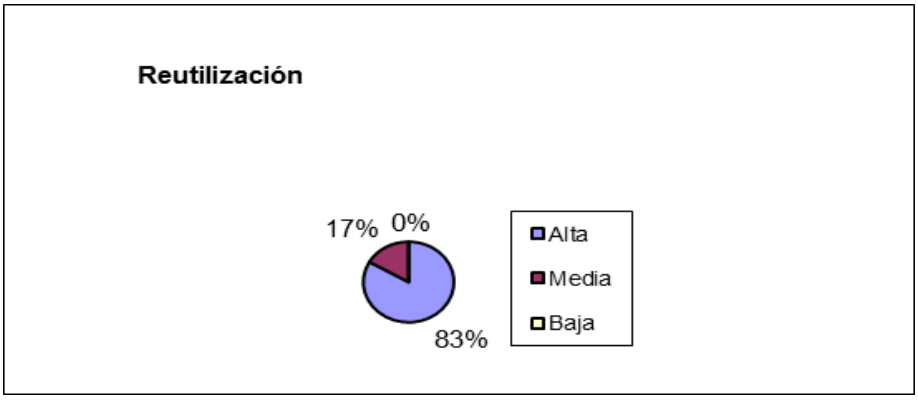

Figura 11 Representación en % de los resultados obtenidos por la aplicación de la métrica (TOC), agrupados en intervalos para el atributo Reutilización.

Examinado los resultados obtenidos mediante el uso de la métrica (TOC) se obtiene que el 78% de las clases poseen entre 1 y 5 procedimientos; se considera que el sistema tiene un diseño simple. Además el 83% de las clases poseen una Responsabilidad y Complejidad baja y también el 83% de ellas son Reutilizables, en base a esto el sistema cuenta con una calidad aceptable.

Relaciones entre Clases (RC): se refiere al número de relaciones de uso de una clase. Determinado por los atributos siguientes: Acoplamiento, Complejidad de mantenimiento, Cantidad de pruebas y Reutilización, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

Tabla 7. Atributos de calidad evaluados por la métrica RC

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.

Tabla 8. Criterios de evaluación para la métrica RC

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	> 2
Complejidad de mantenimiento	Baja	< =Prom.
	Media	Entre Prom y 2* Prom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom y 2* Prom
	Alta	<= Prom.
Cantidad de pruebas	Baja	< =Prom.
	Media	Entre Prom y 2* Prom.
	Alta	> 2* Prom.

Resultados de la evaluación de la métrica Relaciones entre Clases (RC).

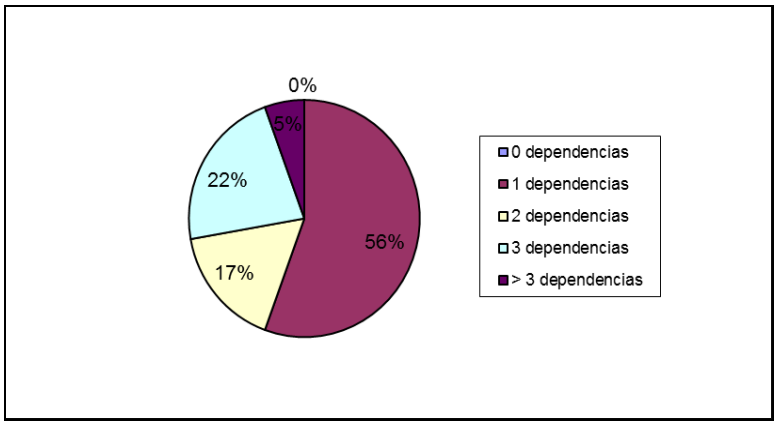

Figura 12 Representación en % de los resultados obtenidos por la aplicación de la métrica (RC), agrupados en intervalos.

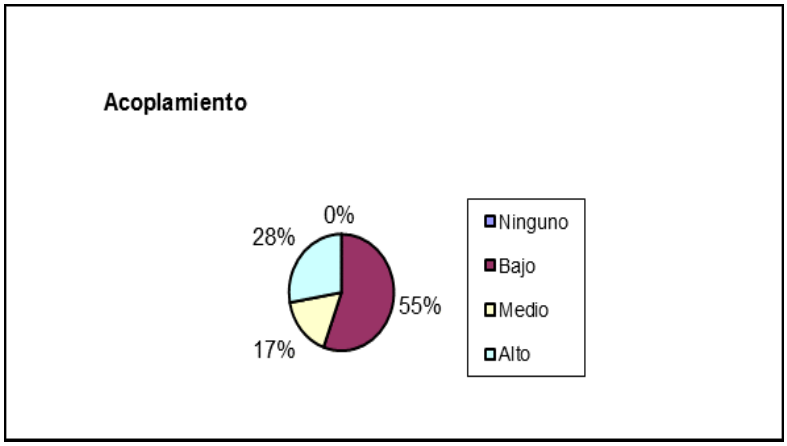

Figura 13 Representación en % de los resultados obtenidos por la aplicación de la métrica (RC), agrupados en intervalos para el atributo Acoplamiento.

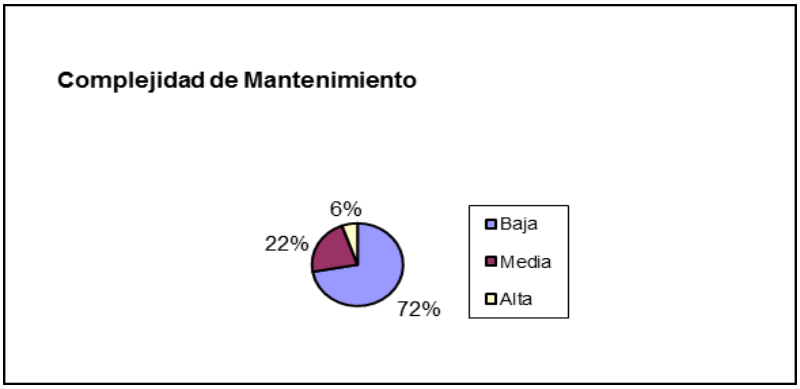

Figura 14 Representación en % de los resultados obtenidos por la aplicación de la métrica (RC), agrupados en intervalos para el atributo Complejidad de mantenimiento.

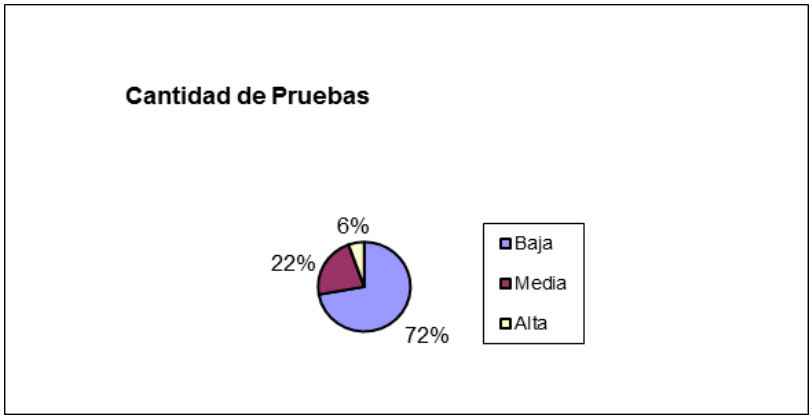

Figura 15 Representación en % de los resultados obtenidos por la aplicación de la métrica (RC), agrupados en intervalos para el atributo Cantidad de pruebas.

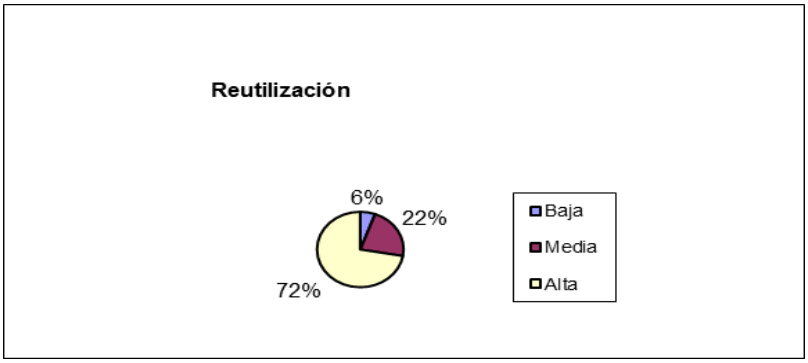

Figura 16 Representación en % de los resultados obtenidos por la aplicación de la métrica (RC), agrupados en intervalos para el atributo Reutilización.

Examinado los resultados obtenidos mediante el uso de la métrica (RC) se obtiene que el 56% de las clases poseen una dependencia, el nivel de la Cantidad de pruebas y Complejidad de mantenimiento es bajo para un 72% de las clases. El Acoplamiento se comporta satisfactoriamente para un 55% de las clases, mientras que el grado de Reutilización se mantiene alto para un 72% de ellas. Por tanto se puede concluir que el sistema posee un diseño simple.

2.10 Conclusiones parciales

Con la modelación del negocio se obtuvo como principal artefacto el Modelo de dominio, alcanzando un mayor entendimiento de la investigación. Las descripciones de requisitos brindaron una mayor claridad de las funcionalidades que debían cumplir las necesidades del cliente y la aplicación de las técnicas de validación probaron que los requisitos identificados estaban descritos de forma correcta, clara y consistente. El diseño se validó a través de las métricas TOC y RC, arrojando resultados satisfactorios debido a que los atributos de calidad alcanzaron niveles favorables. El diseño realizado es simple, debido a que el 78% de las clases resultaron ser pequeñas por lo que la implementación de forma general es sencilla, disminuyendo en gran medida la responsabilidad de las clases y aumentando la reutilización. El modelo de datos que se obtuvo describió la representación lógica y física de los datos persistentes.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS

3.1. Introducción

En este capítulo se realiza el diagrama de componentes con el objetivo de visualizar la estructura general del sistema. Se describe el estándar de codificación utilizado, así como el diagrama de despliegue para capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. También se realiza la validación de la disciplina de implementación, utilizando para ello las pruebas de cajas blanca y negra, con el objetivo de garantizar que el software posea la máxima calidad posible.

3.2. Diagrama de componentes

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Muestra la organización y las dependencias entre un conjunto de componentes. En él se situarán bibliotecas, tablas, archivos, ejecutables y documentos que formen parte del sistema (Schmuller, 2000).

Uno de sus usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema. A continuación en la Figura 17 se representa el diagrama general de componentes del módulo de MCC para la gestión de la información de los requerimientos de mantenimiento en el cual están contenidos los componentes definidos en el subsistema, así como las distintas dependencias existentes entre ellos. El módulo MCC comprende un componente con este mismo nombre que hace uso de la biblioteca TCPDF para la generación de reportes, así como de un componente externo denominado Activo.

A continuación se muestra el diagrama de componentes:

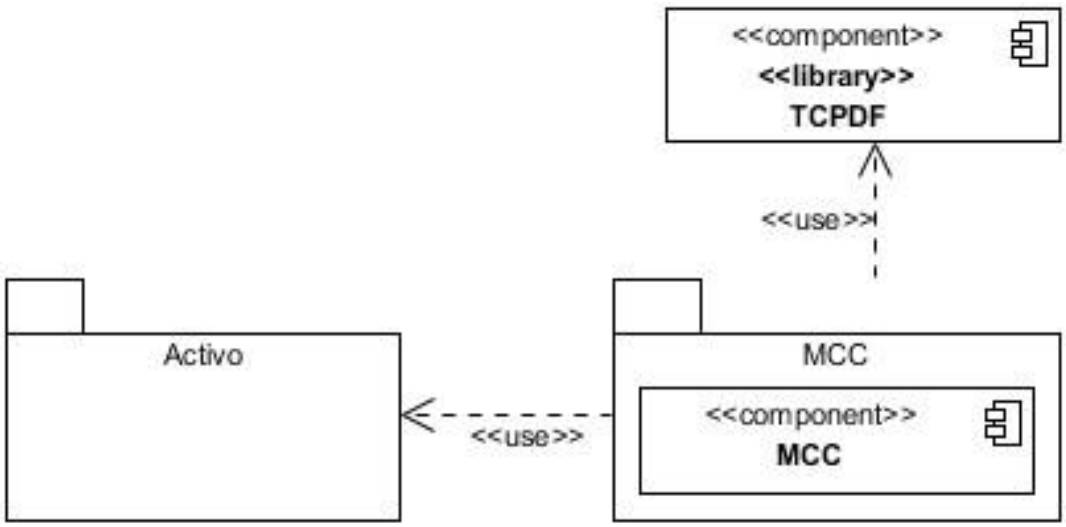

Figura 17 Diagrama de componentes

3.3. Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. El empleo de los estándares de codificación conllevan a lograr un código más legible y reutilizable, de tal forma que pueda aumentar su mantenibilidad a lo largo del tiempo (Microsoft, 2003).

A continuación se especifican los estándares de codificación que fueron utilizados en el desarrollo del módulo MCC para un sistema de Confiabilidad operacional:

Nomenclatura de las clases:

- Para las clases controladoras el nombre comenzará con la primera letra en mayúscula y el resto en minúscula, además estará dado según la función que realiza y se le adicionará al final “Controller”. Ejemplo: HojadecisionController.
- El nombre de las clases del modelo que se encuentran dentro de la carpeta “bussines” comenzará con la primera letra en mayúscula y el resto en minúscula. A este nombre se le agregará al final “Model” y al inicio se le colocará “Nom”, en caso de que sea un nomenclador, de lo contrario será Dat, por ejemplo: DatActivoModel, NomEspecialidadModel.

- Las clases que se encuentran dentro de ¨domain¨ el nombre que reciben es el de la tabla en la Base de Datos. Ejemplo: DatFalloFuncional, NomIntervalo.
- Las clases contenidas en la carpeta “generated” son generadas mediante un mapeo a la base de datos realizado por la herramienta Doctrine Generator, la cual fue desarrollada en el CEIGE y al inicio del nombre se le coloca “Base”. Ejemplo: BaseDatModoFallo, BaseDatActivo.

Nomenclatura de las funciones:

El nombre de los métodos o funciones de una clase comenzará en minúscula, en caso de que sea compuesto se utilizará la notación CamelCasing, o sea, seguido del primer nombre, comenzará el segundo con mayúscula. En caso de que sea una función de una clase controladora se le agregará al final la palabra “Action”. Ejemplo: cargarActivoAction, guardarHojadecisionAction.

Nomenclatura de las variables:

Las variables serán nombradas comenzando en minúscula, en caso de que el nombre de estas sea compuesto, se utilizará la notación CamelCasing, o sea, seguido del primer nombre, comenzará el segundo en mayúscula. En la capa de la Vista, las variables que contengan componentes que formen parte de la interfaz, se les colocará delante un sufijo que indique el tipo de componente que contiene. Ejemplo: idfuncion, idmodoFallo.

3.4. Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Está compuesto por nodos, dispositivos y conectores. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. Permite el mapeo de procesos dentro de los nodos, asegurando la distribución del comportamiento a través de aquellos nodos que son representados. Se definirá una arquitectura cliente-servidor como se muestra en la Figura 16.

A continuación se muestra el diagrama de despliegue del sistema:

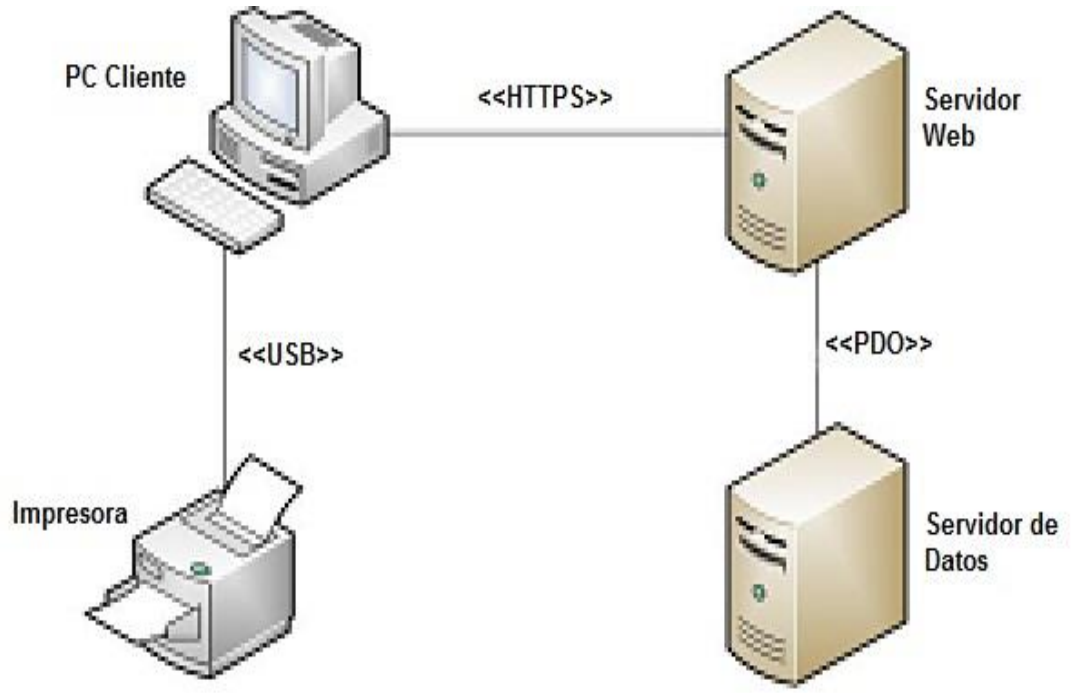

Figura 18 Diagrama de despliegue del sistema

PC Cliente: computadora en la cual la aplicación se ejecutará a través de un navegador. En este caso Navegador Firefox 24.0 o superior.

Servidor Web: radica la lógica de negocio de la aplicación. Servidor web Apache 2.2.

Servidor de base de datos: servidor de datos PostgreSQL 9.1, donde se encuentra la base de datos que utiliza el sistema, este puede estar instalado en la misma computadora donde se encuentra el servidor web.

Impresora: utilizada para imprimir los reportes del sistema en caso de ser necesario.

3.5. Pruebas de software

Uno de los instrumentos adecuados para determinar el status de la calidad de un software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos (Sommerville, 2005).

3.5.1. Pruebas de caja blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usan la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de caja blanca el ingeniero del software puede obtener casos de prueba que garanticen que se ejecuten por lo menos una vez todos los caminos independientes de cada módulo, programa o método. Es por ello que se considera la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplican al software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad (Pressman, 2002).

Se empleará como prueba de caja blanca la técnica **Prueba del Camino Básico** propuesta por Tom McCabe en 1976. Para aplicar esta técnica se hace necesario:

- Enumerar las sentencias del código de la funcionalidad a analizar (Figura 19).
- Elaborar el grafo de flujo de la funcionalidad (Figura 20).
- Calcular la complejidad ciclomática del grafo.
- Determinar el conjunto básico de caminos independientes.
- Preparar los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

```
function obtenerEstructuraFuncionalAction() {                          /1
    $nodo = $this->_request->getPost('node');

if (preg_match("/fallofuncional/i", $nodo) > 0) {                       /2
   $idfallo = str_replace("fallofuncional", '', $nodo);                 /3
   $estructura = $this->obtenerModosFalloPorFalloFuncional($idfallo);   /3
 } else if (preg_match("/funcion/i", $nodo) > 0) {                      /4
    $idfuncion = str_replace("funcion", '', $nodo);                     /5
    $estructura = $this->obtenerFallosFuncionalesPorFuncion($idfuncion); /5
 } else {                                                               /6
      $idanalisismcc = $this->_request->getPost('idanalisismcc');       /7
        $estructura = $this->obtenerFuncionesPorAnalisisMcc($idanalisismcc); /7
    }

    echo json_encode($estructura);                                      /8
}
```

Figura 19 Fragmento de Código de la clase Función Controller

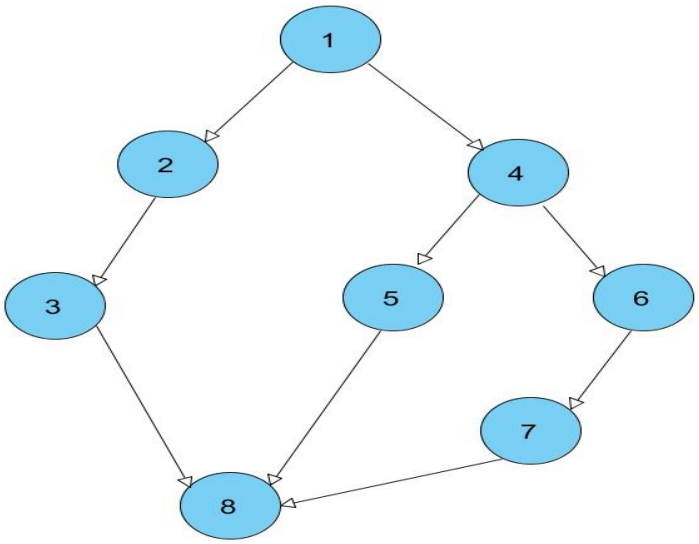

Figura 20 Grafo de flujo asociado a la funcionalidad de obtener estructura funcional

Cálculo de la complejidad ciclomática

Para determinar la complejidad ciclomática existen tres formas diferentes que serán utilizadas para calcular la complejidad del método obtenerEstructuraFuncionalAction () de FuncionController, de esta forma se verificará que el cálculo se haya efectuado de forma correcta si coinciden los tres resultados. Las fórmulas para realizar la operación se definen a continuación:

V(G)= A-N+2 =9-8+2=3

V(G)= 3

Siendo A: aristas y N: nodos

V (G)= P+1= 2+1= 3

V(G)=3

Siendo P: nodo predicado

V(G)= #Regiones= 3

Siendo Regiones: la cantidad de regiones en el grafo.

La complejidad ciclomática es 3, por tanto existen 3 caminos independientes en el grafo de flujo.

Camino 1: 1-2-3-8

Camino 2: 1-4-5-8

Camino 3: 1-4-6-7-8

Para cada uno de los caminos obtenidos se realiza un caso de prueba. Los casos de prueba realizados son los siguientes:

Caso de prueba para el camino básico #1.

Camino1: 1-2-3-8

Entrada: idnodo

Resultados esperados: Se espera que sea del tipo fallo funcional y se cargan los modos de fallo de ese fallo funcional.

Resultados obtenidos: Satisfactorio. Se muestra los modos de fallo de ese fallo funcional.

Caso de prueba para el camino básico #2.

Camino 2: 1-4-5-8

Entrada: idnodo

Resultados esperados: Se espera que sea del tipo función y se cargan los fallos funcionales de dicha función.

Resultados obtenidos: Satisfactorio. Se muestra los fallos funcionales de dicha función.

Caso de prueba para el camino básico #3.

Camino 3: 1-4-6-7-8

Entrada: idanalisismcc

Resultados esperados: Se espera que se obtenga en la estructura las funciones asociadas al análisis MCC.

Resultados obtenidos: Satisfactorio. Se muestra las funciones asociadas al análisis MCC.

3.5.2. Pruebas de caja negra

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.

Caso de prueba: Modificar hoja de decisión

Condiciones de ejecución

- Se debe seleccionar la opción del menú: **Inicio/MCC/Hoja Decisión.**
- Se ha insertado al menos una hoja de decisión en el sistema.

Tabla 9. Caso de prueba: Modificar hoja de decisión

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Modificar hoja de decisión.	Todas las variables son introducidas.	EC 1.1 Escenario modificar hoja de decisión correctamente.	1. Se introducen los datos de la hoja de decisión correctamente. 2. Se presiona el botón **Aceptar**. 3. Se muestra un mensaje de información. 4. Se presiona el botón **Aceptar**.
		EC 1.2 Escenario modificar hoja de decisión incorrectamente.	1 Se introducen datos incorrectos en campos obligatorios. 2 Se presiona el botón **Aceptar**. 3 Se muestra un mensaje informando del error.

		4	Se presiona el botón **Aceptar**.
	EC 1.3 Escenario incompleto modificar hoja de decisión.	1. Se deja campos obligatorios sin llenar. 2. Se presiona el botón **Aceptar**. 3. Se muestra un mensaje informando del error. 4. Se presiona el botón **Aceptar**.	

Para más información sobre el caso de prueba Modificar hoja de decisión, consultar el documento entregable diseño de casos de prueba del módulo MCC. Los casos de prueba elaborados para los restantes requisitos funcionales se encuentran en los documentos entregables, diseño de casos de prueba del módulo MCC.

Después de aplicar los casos de prueba se identificaron un total de 9 no conformidades, las mismas están recogidas en el documento entregable: Resumen de NC del módulo MCC.

En la primera iteración se identificaron 7 no conformidades: 6 de validación y 1 de funcionalidad; en la segunda iteración se identificaron 2 nuevas no conformidades de validación. Todas las no conformidades fueron corregidas al término de cada iteración. Luego se realizó una tercera iteración para identificar cualquier otra no conformidad que pudiese existir comprobando nuevamente que las funciones del software son operativas, que las entradas se aceptan correctamente y las salidas se producen adecuadamente, además que la integridad de la información externa se mantiene. En los resultados obtenidos en la tercera iteración el sistema quedó en total funcionamiento y acorde a las necesidades funcionales requeridas por el cliente. (Consultar documentos entregables No conformidades).

En la figura se pueden observar los resultados obtenidos en las iteraciones.

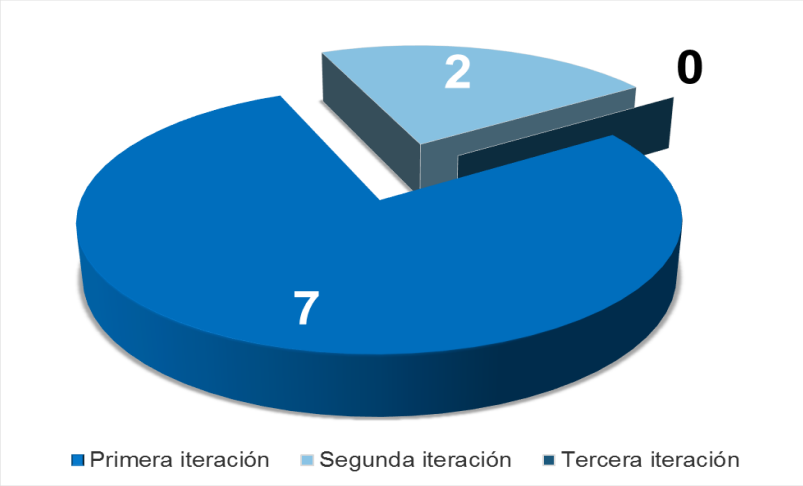

Figura 21 Resultados de las Pruebas Funcionales

3.6. Trazabilidad de los requerimientos de información

La matriz de trazabilidad Requisito-Diseño de casos de prueba representada en la Figura 22, muestra el seguimiento de los requerimientos de información debido a que cada requisito funcional tiene asociado un diseño de caso de prueba. Esta representación permitió comprobar que cada requisito se revisó a través de su diseño de caso de prueba correspondiente, arrojando como resultado que el módulo de Mantenimiento Centrado en Confiabilidad cumplió con las expectativas del cliente.

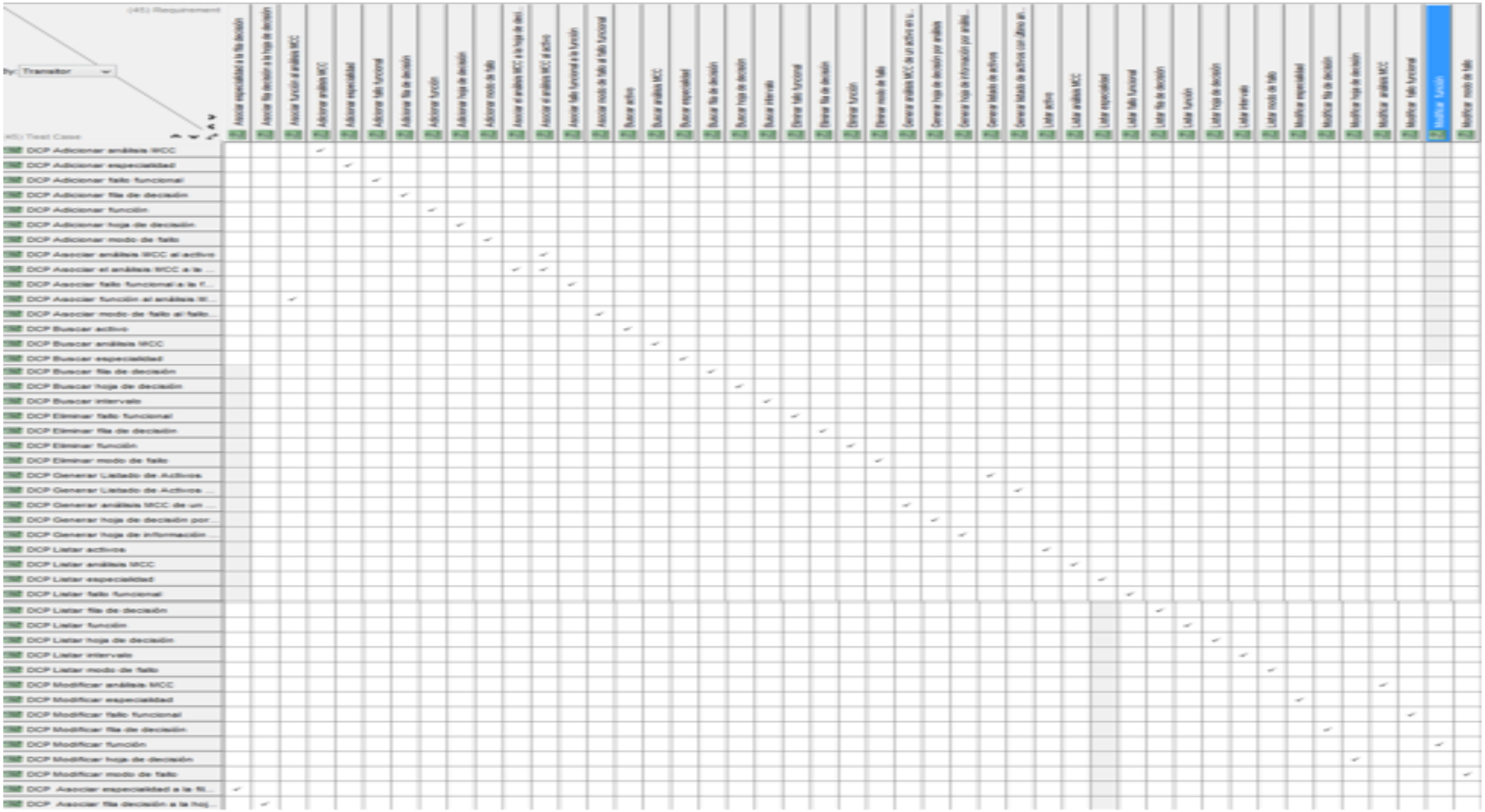

Figura 22 Matriz de trazabilidad Requisito-DCP

3.7. Conclusiones parciales

En el capítulo se describieron los diagramas de componente y despliegue el primero para mostrar las relaciones entre los componentes y el segundo para describir los nodos físicos y protocolos a tener en cuenta para el despliegue del sistema. Las pruebas de caja blanca aplicadas al módulo resultaron ser correctas pues los caminos independientes se ejecutaron al menos una vez y se usaron todas las estructuras de datos internas definidas en la implementación. Se realizaron tres iteraciones de pruebas al módulo MCC empleando las pruebas de caja negra donde las no conformidades identificadas fueron resueltas, demostrando así el cumplimiento de las necesidades funcionales requeridas por el cliente, la integridad de la información externa y la calidad requerida en el mismo.

CONCLUSIONES

Con el desarrollo del módulo de MCC se logró el cumplimiento de los objetivos propuestos en el trabajo de diploma, manifestando que:

- El estudio de diferentes sistemas relacionados al mantenimiento permitió sentar las bases para el desarrollo del módulo MCC, identificando conceptos y funcionalidades candidatas para la propuesta de solución.
- La realización del análisis y diseño permitió obtener los artefactos necesarios para la implementación del módulo, los cuales fueron validados a través de las métricas TOC y RC. Se comprobó además que el diseño realizado es simple, debido a que el 78% de las clases resultaron ser pequeñas por lo que la implementación de forma general es sencilla, disminuyendo en gran medida la responsabilidad de las clases y aumentando la reutilización.
- Con la implementación del módulo MCC se logró gestionar los requerimientos de información, obteniéndose los análisis MCC y la Hoja de decisión mediante los reportes (salidas del sistema) identificados como parte de la propuesta de solución, utilizando la biblioteca de clases TCPDF para su impresión.
- Se aplicaron pruebas de caja blanca y caja negra, las cuales mostraron que el módulo contaba con un correcto funcionamiento, manifestando así el cumplimiento de las necesidades del cliente y la calidad requerida en el mismo.
- Se realizó la trazabilidad de los requerimientos de información mediante las matrices de trazabilidad Concepto-Requisito y Requisito-Diseño de caso de prueba, comprobando que cada concepto se asoció al menos a un requisito y que cada requisito se probó mediante un diseño de caso de prueba.

RECOMENDACIONES

De acuerdo con los resultados obtenidos se recomienda:

- Utilizar el presente trabajo de diploma como guía para el desarrollo de futuros sistemas de MCC para la gestión de los requerimientos de información de mantenimiento.
- Realizar la integración del módulo de MCC para la gestión de los requerimientos de información de mantenimiento con otros módulos y herramientas informáticas para así contribuir a mejorar la Ingeniería de Mantenimiento.

REFERENCIAS BIBLIOGRÁFICAS

Aguinaga Barragán, Álvaro. 2007. *Confiabilidad Operacional para la Ingeniería del Mantenimiento.* Quito : s.n., 2007.

Amendola, Luis José. 2006. *Gestión de Proyectos de Activos Industriales.* Valencia : Universidad Politécnica de Valencia, 2006. ISSN: 84-8363-052-4.

Amendola, Luis José. 2009. Aplicación de la Confiabilidad en la Gestión de Proyectos en Paradas de Plantas Químicas. [En Papers VI Internacional Congreso on ProjectEngineering]. 2009. pág. 184. 84.

Amendola, Luis José y Depool, Tibaire. 2005. *Modelo de Confiabilidad Humana en la Gestión de Activos.* Asociación Española de la Calidad. Madrid : s.n., 2005.

Aranda Amaolo, Gabriela Noemí. 2009. *Marco para la elitización de requisitos software en procesos de desarrollo global.* 2009.

Asuni, Nicola. 2013. *TCPDF.* [En línea] 2013. [Citado el: 29 de Abril de 2014.] www.tcpdf.org.

Ayala, Beatriz, Ramírez , Claudia Marcela y Ocampo, Lina María. 2006. *La Ingeniería de Requeriemientos aplicada al desarrollo de sistema de información.* 2006.

Ballester Jiménez, Lianet. 2012. Arquitectura de software para el producto “Mis Mejores Cuentos”. La Habana : s.n., 2012. Vol. 5, 5.

Barberá Martínez, Luis y Crespo Márquez, Adolfo. 2010. *Revisión de herramientas de software actuales de soporte a la metodología RCM. Una visión estructural y funcional.* Universidad de Sevilla. Sevilla : s.n., 2010.

Barrios Gárciga, Alfredo. 2005. *Análisis de criticidad de los subsistemas objetos de mantenimiento en una instalación hotelera.* Centro de Estudios en Ingeniería de Mantenimiento, Instituto Superior Politécnico José Antonio Hecheverría. La Habana : s.n., 2005. Tesis de grado.

Barrios, Aracelis y Ortiz, Maritza. 2012. *El Mantenimiento en el Desarrollo de la Gestión Empresarial, Fundamentos Teóricos.* Observatorio de la Economía Latinoamericana. 2012.

Cabrera Pereira., Leisniel Ignacio y Hernández Gómez, Maylin. 2009. *Análisis y Diseño del Módulo de Cobros y Pagos del sistema integral de gestión CEDRUX*. La Habana: s.n., 2009.

Capdevila Camacho, Rodrigo Alberto, Muñoz Casals, Velmour y del Valle Román, Luismel. 2013. *SLD252 Capa de getión de servicios web para el Sistema de Gestión para la Ingeniería Clínica y Electromedicina.* La Habana : s.n., 2013. Memoria de Evento Informática 2013. ISBN: 978-959-7213-02-4.

CEIGE. 2013. *Modelo de Desarrollo de Software v1.2.* CEIGE. La Habana : s.n., 2013.

Consultoría y Sistemas, S.L. 2011. Canarytek Consultoría y Sistemas, S.L. *Consultoría y Sistemas, S.L.* [En línea] 2011. [Citado el: 21 de Abril de 2014.]

http://www.canarytek.com/tutoriales/alfresco/explorador-documental-de-alfresco/trabajar-con-flujos-.

Correa Lozano, Pablo Ramiro. 2010. *Análisis comparativo de los Frameworks Adobe Flex, Java Rich Faces y Extjs para el desarrollo de aplicaciones enriquecidas en internet (RIA).* Escuela politécnica Nacional de Ecuador. Quito : s.n., 2010. Tesis de grado.

Craig, Larman. 2003. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2003.

Cubadebate. 2013. La Industria Cubana se reorganiza buscando productividad y eficiencia. *Cubadebate.* [En línea] 24 de Octubre de 2013. [Citado el: 21 de Abril de 2014.] http://www.cubadebate.cu/especiales/2013/10/24/la-industria-cubana-se-reorganiza-buscando-productividad-y-eficiencia/.

Da Costa Burga, Martín. 2011. *Aplicación del mantenimiento centrado en la confiabilidad a motores a gas de dos tiempos en pozos de alta producción.* 2011.

Drake, José M y Barros, Laura. 2009. *Programación Concurrente y Distribuida.* 2009.

Escalona, María José y Koch, Nora. 2002. *Ingeniería de Requisitos en Aplicaciones para la Web–Un estudio comparativo.* 2002.

Gómez Baryolo, Oiner. 2011. Sistema de gestión integral de seguridad Acaxia. La Habana : s.n., 2011. Vol. 4, 7.

Gómez de León, Félix Cesáreo. 1998. *Tecnología del mantenimiento industrial.* Murcia : EDITUM, 1998.

González Brito, Henry Raúl, González Cádiz, Yennis y Linares Pons, Naryana. 2013. *Sistema informático para la gestión de seminarios científicos en un centro de desarrollo de software.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2013. ISBN: 978-959-286-022-3.

Guerra López, Alejandro. 2012. *Análisis, Diseño e Implementación del proceso Administración de Recursos Humanos del Sistema de Gestión de Mantenimiento Vehicular v 1.0.* Centro de Informatización de la Gestión de Entidades, Universidad de las Ciencias Informáticas. La Habana : s.n., 2012. Tesis de grado.

Hernández Carballido, José y Talló Sendra, Marc. 2013. *Portal web para la empresa Ruben´s Instalaciones y Excavaciones.* Universidad Autónoma de Barcelona. 2013. Tesis de grado.

Mariñán, Martín Pérez. 2002. *Design Patterns.* 2002.

Márquez Díaz, José, Sampedro, Leonardo y Vargas, Félix. 2013. Instalación y configuración de Apache, un servidor Web. 2013.

Montaña Riveros, Leonardo y Rosas Niño, Elkin Gustavo. 2006. *Diseño de un sistema de mantenimiento con base en análisisde criticidad y análisis de modos y efectos de falla en la*

planta de coque de fabricación primaria en la empresa Acerías Paz del Río S.A. Facultad Seccional Duitama, Escuela de Ingeniería Electromecánica Duitama. 2006. Tesis de grado.

Montes de Oca Hernández, Yanirys y Brito D, Yuliesky. *Biblioteca virtual Eumed.net. s.l.*: Fundación Universitaria Andaluza Inca Garcilaso, 2011.

Moubray, John y Maintenance, Certified. 2013. *Mantenimiento Centrado en Confiabilidad (RCM)*. [trad.] Carlos Mario Pérez Jaramillo. Junio de 2013.

Nuñez Sanz, Claudia, y otros. 2013. Sistema de Captura de Información para la Toma de Decisiones e Inteligencia de Negocio. La Habana : s.n., 2013. Vol. 6, 4.

Parra Márquez, Carlos, y otros. 2013. *Revisión de herramientas informáticas para el análisis de la fiabilidad, disponibilidad, mantenibilidad y seguridad (RAMS) de equipos industriales.* Departamento de Organización Industrial y Gestión de Empresas, Escuela Superior de Ingenieros, Universidad de Sevilla. 2013.

Pentón Saucedo, Ángel Eduardo y Castillo Serpa, Alfredo. 2012. Aplicación de la Tabla Ortogonal en el diseño de los Casos de prueba de Software. Application of the orthogonal arrays in the design of the Cases of test of Software. 2012. Vol. 15, 2, págs. 1-12.

Pérez Cortés, Alejandro. 2010. Implementación de tableros de control (indicadores) en el área de mejora continua en una empresa de manufactura. *Eumed.* [En línea] Junio de 2010. [Citado el: 22 de Abril de 2014.] http://www.eumed.net/libros-gratis/2011a/896/MODELOS%20DE%20DESARROLLO%20DE%20SOFTWARE.htm.

Persegona, Marcelo Felipe, y otros. 2012. Sistema de Información Geográfica Aplicada al Estudio de los Datos de Los Profesionales de Enfermería del Brasil. 2012. Vol. 3, 4. ISSN: 2357-707X.

Piñero González, Maidelyn, Jiménez Morales, Yenier y Kubota López, Emilio. 2013. Infraestructura de datos espaciales para la Universidad de las Ciencias Informática. La Habana : s.n., 2013. Vol. 6, 7.

Piñero González, Maidelyn, Jiménez Morales, Yenier y Kubota López, Emilio. 2013. Infraestructura de datos espaciales para la Universidad de las Ciencias Informática. La Habana : s.n., 2013. Vol. 6, 7.

Pressman, Roger S. 2002. *Ingeniería del software. Un enfoque practico.* Quinta edición. Madrid : MacGraw-Hill, 2002.

Pressman, Roger S. 2006. *Ingeniería del Software Un enfoque práctico.* 2006.

Raña González, Luz del Alba. 2010. *Evaluación de la función mantenimiento en empresas transportistas.* 2, 2010, Revista Ciencias Técnicas Agropecuarias, Vol. 19, págs. 10-15.

Renove Tecnología S.L. 2009-2014. [En línea] 2009-2014. [Citado el: 6 de Mayo de 2014.] http://www.renovetec.com/index.php/86-renovetec-ingenieria/331-renove-gem.

Renove Tecnología S.L. 2009-2014. [En línea] 2009-2014. [Citado el: 10 de Mayo de 2014.] http://www.renovetec.com/590-mantenimiento-industrial/110-mantenimiento-industrial/307-software-de-mantenimiento-gratuito-pmx-pro .

Rivera Calero, Álvaro Enmanuel y Rodríguez Almache, Martha Gudelia. 2012. *Estudio de factibilidad, desarrollo e implementación de un sistema integrado de gestión educativa aplicable a cualquier nivel de educación (SIGA)–Proceso a automatizar Planificación Curricular.* Facultad de Ingenierías, Universidad Politécnica Salesiana. Guayaquil : s.n., 2012. Tesis de grado.

Robles Aranda, Yadira y Sotolongo, Anthony R. 2013. Integración de los algoritmos de minería de datos 1R, PRISM e ID3 a PostgreSQL. 2013. Vol. 10, 2, págs. 389-406.

Santiesteban Pérez, Irina Ivis, y otros. 2011. Desarrollo de funcionalidades que faciliten al docente su preparación y el control del aprendizaje de los estudiantes en la plataforma educativa Zera. La Habana : s.n., 2011. Vol. 4, 11.

SENCAMER. 2011. COVENIN. [En línea] 2011. [Citado el: 2 de Mayo de 2014.] http://www.sencamer.gob.ve/sencamer/normas/3049-93.pdf.

Serna, Edgar y Arango, Fernando. 2012. Desafíos y estrategias prácticas de los estudios empíricos sobre las técnicas de prueba del software Challenges and practical strategies of the empirical studies on software testing techniques SYSTEMS ENGINEERING. 2012. Vol. 13, 1.

Sommerville, Ian. 2005. Ingeniería de Software. 2005. ISBN:8478290745.

Torres, Victoria y Urdaneta, Nancy. 2012. *Diseño de la Página Web del Canal Juvenil RNV Activa 103.9 FM.* 2012. Tesis de grado.

Universidad de Sevilla. 2010. *Análisis de sistemas RCM.* Sevilla : s.n., 2010.

Vallejo Moreno, Diego Alejandro y Florez Castillo, Diego Fernando. 2012. *Localizador táctil para productos de supermercado.* Corporación Universitaria Minuto de Dios. Bogotá : s.n., 2012. Tesis de grado.

Velázquez Osorio, Inoelkis, Cordero Estrada, Lisandra y Bauta, René Rodrigo. 2013. *Procedimiento para actualización de la capa de acceso a datos del marco de trabajo Sauxe Doctrine 1.2.2 A Doctrine 2.2.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2013.