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

Módulo de Gestión de problemas para el Centro de Soporte de la Universidad de las Ciencias Informáticas

Autores:

Yorgüy Antonio Batista Desdin

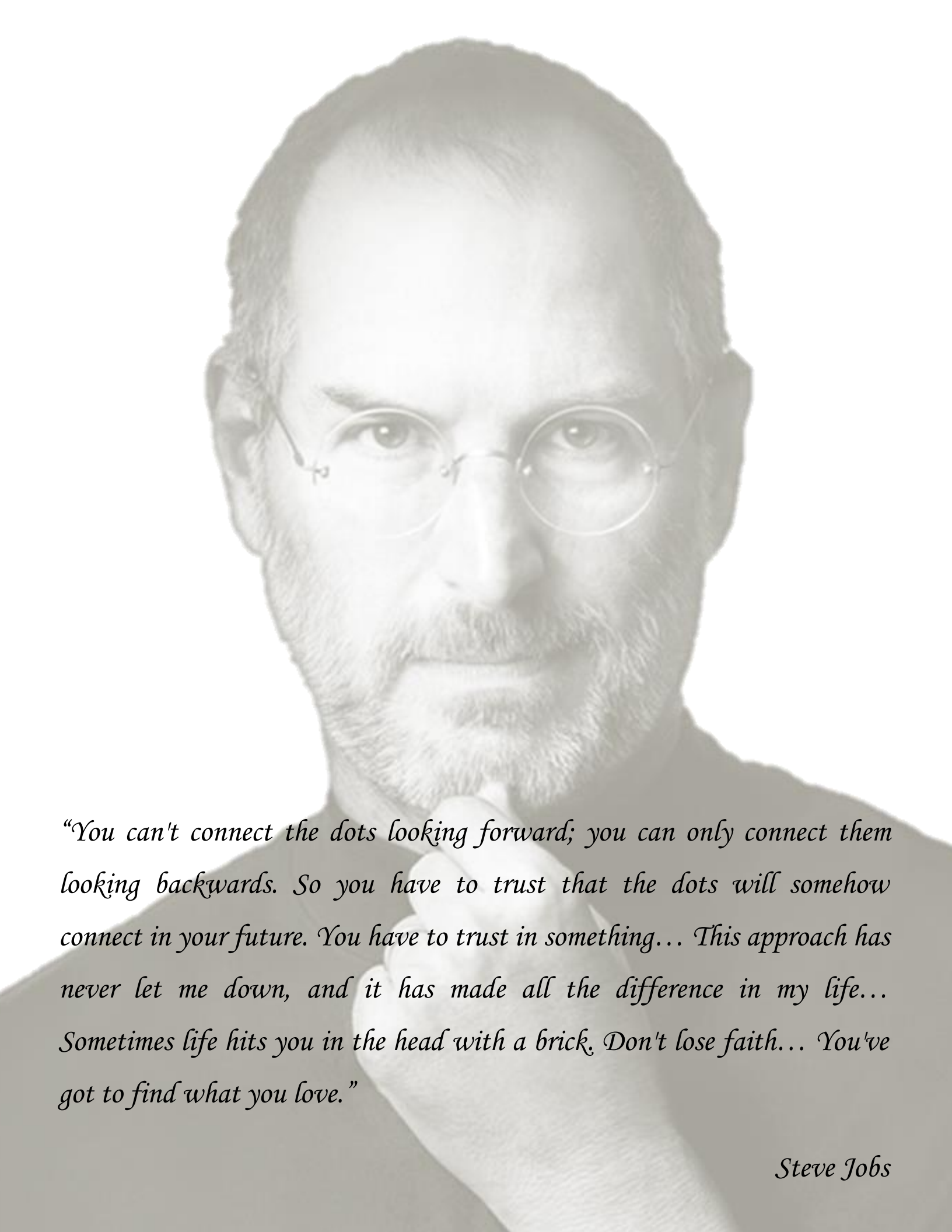
Andy Daniel Meriño Coronado

Tutores:

MSc. Yulio Seriocha García Gallardo

Ing. Neybis Lago Clara

La Habana, 2014



"You can't connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future. You have to trust in something… This approach has never let me down, and it has made all the difference in my life… Sometimes life hits you in the head with a brick. Don't lose faith… You've got to find what you love."

Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaramos que somos los autores de este trabajo titulado: “Módulo de Gestión de problemas para el Centro de Soporte de la Universidad de las Ciencias Informáticas”, y otorgamos al Centro de Soporte de la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año_____.

Yorgüy Antonio Batista Desdin

_____

Autor

Andy Daniel Meriño Coronado

_____

Autor

MSc. Yulio Seriocha García Gallardo

_____

Tutor

Ing. Neybis Lago Clara

_____

Tutor

DATOS DE CONTACTO

Autor: Yorgüy Antonio Batista Desdin

Correo: yabatista@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Autor: Andy Daniel Meriño Coronado

Correo: admerino@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Tutor: MSc. Yulio Seriocha García Gallardo

Correo: ygarciag@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Tutor: Ing. Neybis Lago Clara

Correo: nlago@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

DEDICATORIA

De Yorgüy:

A mi abuelo Felipe, que Dios lo llamó a su lado antes de que pudiese regalarle este título.

A mi madre Oralis por ser la guía de mi vida y la fuerza de mí alma.

A mi padre José por ser la visión objetiva de mis decisiones.

A mi padrastro Luis por acogerme como un hijo y estar a mi lado siempre.

A mi Tata por ser mi compañera, mi amante y mi confidente.

A mi hermanita Adriana por dejarme ser su ejemplo y guía.

De Andy:

A mi pequeño gigante por quitarme las pocas horas de sueño.

A mi Reina por estar presente dándome ánimo en los buenos momentos, siendo mi sostén en los malos y por ser la persona a la que tanto quiero.

A mis padres por apoyarme en todas las locuras y alentarme a alcanzar las metas propuestas.

A mi flaca por pedir un hermano y soportarme la vida entera.

A mis abuelos por quererme y malcriarme tanto durante todos estos años.

A mis tíos y tías que me han enseñado tantas cosas en esta vida.

A mi suegra por quererme como si fuera un hijo más y apoyarme en estos años tan difíciles.

AGRADECIMIENTOS

De Yorgüy:

Gracias a mi madre por darme todo el amor del mundo, su apoyo en los momentos difíciles, sus consejos, sus palabras de aliento y sobre todo por traerme a la vida e inculcar en mí los valores que hoy me hacen la persona que soy.

Gracias a mis dos padres José y Luis por ayudarme cada día a cruzar con firmeza el camino de la vida, porque con su apoyo y aliento hoy he logrado uno de mis más grandes anhelos.

Gracias a mis abuelos Regina, Paco, Felipe por darme todo el amor del mundo en vida y cuidarme ahora desde el cielo.

Gracias a mi abuela Dulce por ser el aire que respiro, por representar en mi vida la cúspide de la dedicación a la familia.

Un agradecimiento especial a mi novia Idelmis (Tata) por acompañarme a lo largo de toda mi carrera, por estar a mi lado en todos los momentos buenos o malos, y sobre todo por darme su amor incondicional.

Gracias a mi hermanita Adriana porque desde su inocencia me ha enseñado humildad y sinceridad, porque no existe amor más sincero que el de mi niña.

Gracias a mi amigo y compañero Andy con el cual comparto no solo esta tesis sino además una gran amistad.

Gracias a mi suegra Marina por preocuparse por mí como si fuese su hijo.

Gracias a Yanet que más que una amiga es una hermana, gracias por inculcarme el deseo de la superación.

Gracias a Eudel, Yunior y Orlando por ser mis hermanos en la universidad, mis bastones en los momentos difíciles y sobre todo porque su amistad no tiene precio.

Agradecer también aquellas personas que han demostrado su valía en los momentos difíciles: Wilfredo, Reynaldo, Rainer, Juniel, Victor, Odette, Mailyn y Yudelyn.

Gracias a la gente del proyecto por soportarme durante tres cursos: Maylevis, Leo, Eddie, Luis Manuel, Irene, Yoendry, Idel, Yordanis y Yorlen.

Gracias a todos mis compañeros de brigada.

De Andy:

*Gracias a **mi familia**, en especial a mi compañera, **madre de mi hermoso hijo** a la que le debo ese regalo tan especial. Un infinito agradecimiento a **mis padres** por ser ejemplo y guía en la interminable aventura que es vivir, por darme todo su amor, ayudarme en todo momento e inculcar en mí los valores de una persona honrada, honesta que no tiene miedo a vivir.*

*Gracias a **mi hermana**, por todos los consejos que me ha dado, por todo lo que me ha ayudado.*

*Gracias a **mi suegra** por todo lo que ha tenido que luchar conmigo estos últimos años.*

*Gracias a **mi compañero de tesis** por entender mis prioridades y ayudarme en todo momento a cumplir con ellas.*

*Gracias a **mi cuñado** por buscar tanta comida para este estómago insaciable.*

*Gracias a **mis amigos** más cercanos Abel, Wilfredo, Mailyn, Lily, Héctor, Brayan, Juan Carlos, Rolando, Luis.*

*Gracias a mi primo **Pepe,** de no ser por él no hubiera llegado nunca a un turno de clases temprano.*

*Gracias a **mi tío** Eduardo por enseñarme a ensuciarme las manos de grasa, llenarme la cara de sudor y amar el trabajo. Gracias a **mi tía** Yelina por todos esos barcos de comida que pone en la mesa cuando yo estoy. Gracias a **todos mis tíos** y primos.*

*Gracias a la **Dra. Navarrete** por dedicar tanto tiempo a la corrección de los errores de esta tesis.*

*Gracias a los **profesores** que han puesto a nuestra disposición sus conocimientos y de una forma u otra nos han brindado su ayuda incondicional en todo momento, gracias a Juniel, Zénel, Boris, Yordanis.*

Gracias a todos aquellos que han estado presente a lo largo de la carrera y por azares de la vida no menciono.

Resumen

La Gestión de problemas es uno de los procesos que se llevan a cabo como parte del servicio de soporte en la Universidad de las Ciencias Informáticas. La aplicación informática con la que se tramitan los problemas surgidos en los sistemas desarrollados, no satisface los requerimientos particulares de la universidad. Además, es un sistema de Gestión de servicios de tecnología propietaria, lo cual impide modificar el flujo de actividades que hasta el momento se desarrollan para cumplir con la labor del centro. Otra de las desventajas de este sistema es que no se ajusta a la actualización del estándar Information Technology Infrastructure Library (ITIL) en su versión 2011.

Por tales motivos se decide desarrollar una aplicación que cumpla con el proceso de Gestión de problemas como parte de la Gestión de servicios en el Centro de Soporte de la Universidad de las Ciencias Informáticas, basado en el estándar ITIL 2011. La creación de esta nueva aplicación permitirá ahorrar recursos monetarios a la Universidad y la obtención de una solución ajustada a los requerimientos del centro y las necesidades de sus trabajadores.

Palabras claves: Gestión de servicios, Gestión de problemas, servicios de soporte técnico.

Contenido

Introducción

Las instituciones, empresas y gobiernos cada vez son más dependientes de la información, por ello se necesitan mecanismos de control que contribuyan a mejorar el funcionamiento interno y sistémico que aseguren la eficiencia y la competitividad con las demás instituciones. A esto pueden contribuir las llamadas Tecnologías de la Información (TI), las cuales han evolucionado y están cambiando el modo tradicional de realizar las actividades. Ejemplo de esto son la Internet, las tarjetas de crédito, el pago electrónico de la nómina, entre otros servicios que han potenciado el incremento de la utilización de esas tecnologías. Los servicios que brindan las TI están enfocados a suplir en gran medida las necesidades particulares de los usuarios finales, como es el caso del soporte técnico. Este sirve para ayudar a resolver los problemas que puedan presentar los usuarios mientras hacen uso de servicios, programas o dispositivos (APM Group Ltd, 2013).

El manejo de los servicios de TI se conoce como Gestión de Servicios de TI (GSTI). La GSTI es una disciplina de gestión basada en procesos horizontales diseñados para facilitar una metodología orientada al cliente, que mejora considerablemente la alineación entre la organización de TI (Proveedora de Servicios de TI) y los clientes (usuarios responsables del uso de estos servicios para el cumplimiento de los objetivos del negocio) (SICELCA IT Systems, 2013). Los estándares de Gestión de servicios más utilizados son COBIT [1], Information Technology Infrastructure Library (ITIL), CMMI [2] e ISO [3] 20000 (IT Governance Institute, 2008). Dentro de estos estándares el más aceptado a nivel mundial es ITIL, el cual proporciona un conjunto coherente de buenas prácticas, derivado de los sectores públicos y privados a nivel internacional (OGC ITIL, 2011).

En Cuba el estado potencia la utilización de las TI mediante la creación de centros que fortalecen el conocimiento sobre las nuevas tecnologías. Entre ellos se encuentra la Universidad de las Ciencias Informáticas (UCI), la cual juega un papel importante en la informatización de la sociedad cubana. Esta universidad cuenta con una red de centros que realizan actividad de desarrollo–producción, 14 son centros de desarrollo y 2, centros que brindan servicios transversales: el Centro de Calidad de Software y el Centro de Soporte.

[1] Del inglés: *Control Objectives for Information and related Technology.*
[2] Del inglés: *Capability Maturity Model Integration.*
[3] Del inglés: *International Organization for Standardization.*

El Centro de Soporte UCI presta servicios de soporte técnico a los usuarios de las aplicaciones desarrolladas por la red de centros de desarrollo, y de este modo centraliza la atención al cliente. Además, existe una estrecha comunicación con los centros de la universidad, lo cual es fundamental para la prestación de un servicio con calidad. Las actividades que se realizan en este centro son ejecutadas a través de una plataforma Service Desk[4] basada en ITIL versión 2, que constituye una tecnología propietaria, lo que trae como consecuencia que no pueda modificarse el flujo de actividades que sigue.

Las aplicaciones liberadas por la universidad en ocasiones presentan dificultades en su funcionamiento, las que son reportadas al servicio de soporte técnico como una incidencia. Se debe señalar que el Centro de Soporte UCI tiene implantado un sistema que cuenta con una base de soluciones en la que se almacenan las respuestas a las incidencias. Esta base se nutre de tres fuentes fundamentales: el trabajo proactivo, la Gestión de incidencias y la Gestión de problemas.

El trabajo proactivo consiste en crear soluciones, antes de que ocurran las incidencias, a partir del análisis que realizan especialistas y técnicos o de sus experiencias personales. La Gestión de incidencias realiza un tratamiento directo a los incidentes y crea soluciones temporales. La Gestión de problemas tramita aquellos incidentes que por su frecuencia de aparición o por su impacto en el servicio se transforman en un problema. Procesar un problema significa determinar una causa y proponer una solución definitiva.

En el Centro de Soporte UCI la Gestión de problemas se encuentra restringida debido a la ausencia del registro de errores conocidos, los cuales son definidos como aquellos problemas sobre los cuales se conoce su causa y se tiene una solución. La restricción está dada por el hecho de que la base de soluciones del Centro citado no permite consultar la relación que existe entre la solución y los incidentes que les dieron origen. La falta de información sobre la relación que se establece entre los problemas, sus causas y soluciones provoca demora en el tratamiento de las nuevas incidencias.

La existencia de más de 1500 entradas dentro de la base de soluciones provoca demora en la búsqueda. Los parámetros de búsqueda están restringidos a palabras claves definidas para cada una de las soluciones y por el nombre del sistema del cual proviene el incidente. La relación entre la solución y el problema tiene que ser interpretada a partir del nombre de la solución. Las soluciones cuentan además con una descripción, la que es redactada por los

[4]Es una solución de resolución de problemas y respuesta ante incidentes automatizada para una reparación efectiva y rápida de incidentes de usuario final.

técnicos y especialistas y está sujeta a posibles errores humanos. Estos errores pueden ser ortográficos, de concordancia, coherencia o relacionados con imágenes y otros elementos descriptivos. Los errores deben ser corregidos de forma manual por un especialista asignado para esta tarea. Esta corrección se realiza revisando el total de soluciones y comprobando cuales presentan errores. Las soluciones con errores son analizadas como soluciones válidas en el tratamiento de los problemas.

La demora en la búsqueda de soluciones a los problemas afecta de manera directa el rendimiento laboral de los técnicos y especialistas. Este rendimiento viene dado por el número de problemas solucionados en un tiempo determinado. La disminución de este rendimiento provoca insatisfacción por parte del cliente, quien espera recibir una respuesta en el menor tiempo posible. Los técnicos y especialistas realizan otras tareas además del tratamiento a los problemas, la dificultad que existe en este proceso afecta las demás responsabilidades laborales de estos trabajadores. Una menor cantidad de problemas atendidos se traduce en un mayor costo por concepto de servicios profesionales.

Teniendo en cuenta la problemática anteriormente descrita se plantea el siguiente **problema a resolver:** El modo de gestionar los problemas en el Centro de Soporte UCI disminuye el rendimiento laboral de los técnicos y especialistas de soporte.

Teniendo como objeto de estudio: La Gestión de servicios de soporte de TI.

Definiendo como campo de acción: El proceso de Gestión de problemas en el Centro de Soporte UCI según el estándar ITIL 2011.

Objetivo general: Desarrollar un módulo que aumente el rendimiento laboral de técnicos y especialistas a través de la Gestión de problemas en el Centro de Soporte UCI.

Objetivos específicos:

1. Realizar el marco teórico y el estado del arte de la investigación para obtener información sobre el funcionamiento de sistemas que realicen la Gestión de problemas.
2. Realizar el estudio de las herramientas, tecnologías y metodologías a utilizar para el desarrollo del módulo.
3. Diseñar el módulo para la Gestión de problemas.
4. Implementar el módulo para la Gestión de problemas.
5. Diseñar y aplicar pruebas de aceptación, unitarias y de sistema para validar funcionalmente el módulo.

Para cumplir con el objetivo general propuesto y darle solución al problema planteado, se proponen las siguientes tareas de investigación:

1. Elaboración del diseño teórico de la investigación.
2. Análisis de los procesos y las fases de ITIL.
3. Análisis del estado del arte sobre los sistemas que realizan la Gestión de problemas.
4. Realización del estudio de las herramientas y tecnologías a utilizar en el proceso de desarrollo.
5. Realización del análisis y diseño de la solución.
6. Realización del modelo de implementación.
7. Implementación de los componentes del módulo.
8. Diseño de los casos de prueba de las funcionalidades implementadas.
9. Ejecución de las pruebas a partir de los casos de prueba.
10. Validación de la solución a partir del objetivo propuesto.

Como idea a defender se plantea: Con el desarrollo de un módulo que realice la Gestión de problemas basado en ITIL 2011 aumentará el rendimiento laboral de técnicos y especialistas en el Centro de Soporte UCI.

Para darle cumplimiento al objetivo general y respuesta al problema a resolver planteado fueron aplicados los siguientes métodos científicos de investigación:

- Análisis histórico–lógico: este método se utilizó para entender los antecedentes del tema de investigación, así como otras temáticas relacionadas a lo largo de la historia del estándar ITIL.
- Analítico–sintético: este método se utilizó para analizar y comprender la teoría y documentación relacionada con el tema de investigación, permitiendo así, extraer los elementos relacionados con el objeto de estudio.
- Observación: este método ha permitido percibir a partir de la situación real que se está investigando, cómo se desarrolla el proceso que constituye el objeto de estudio. Se utilizó para realizar el diagnóstico sobre el sistema implantado en el Centro de Soporte UCI.
- Entrevista: este método fue aplicado durante la realización de entrevistas a los especialistas del Centro de Soporte UCI, quienes han aportado elementos significativos a la investigación.

El documento está constituido por tres capítulos estructurados de la siguiente forma:

Capítulo 1 Fundamentación Teórica: se definen elementos teóricos que sustentan la investigación. Además se realiza un estudio del estado del arte de soluciones existentes que permiten realizar la Gestión de problemas, y se definen las herramientas y tecnologías a utilizar.

Capítulo 2 Análisis y Diseño: se especifican los distintos artefactos a generar por cada una de las etapas de la metodología de desarrollo escogida. Se especifica el diseño de la propuesta de solución. Este diseño es validado a través de distintas métricas. Se realiza el plan de entrega de la solución.

Capítulo 3 Implementación y pruebas: se implementa el módulo para darle solución a la situación problemática planteada. Se realizan las pruebas unitarias, de aceptación y de sistema para validar funcionalmente la propuesta de solución. Se valida que la propuesta de solución cumple con el objetivo planteado.

Capítulo 1: Fundamentación Teórica

1.1. Introducción

En el presente capítulo se analizará el proceso de Gestión de problemas de la metodología ITIL 2011. Además se profundizará en la necesidad de la solución propuesta y se abordarán los elementos y características fundamentales de ITIL 2011. Se realizará un análisis de los diferentes sistemas de Gestión de problemas existentes. También se realizará en este capítulo un estudio de las metodologías de desarrollo de software, tecnologías y herramientas a utilizar.

1.2. Estándares para la Gestión de servicios de TI

Los estándares de Gestión de servicios más utilizados son COBIT, ITIL, CMMI e ISO 20000 (IT Governance Institute, 2008). COBIT no está orientado a los procesos y es un conjunto de buenas prácticas para ayudar a la administración de los gobiernos de TI. Sirve para planear, organizar, dirigir y controlar toda la función informática dentro de una empresa. Actúa sobre la dirigencia y ayuda a estandarizar la organización. Las buenas prácticas de COBIT están enfocadas fuertemente en el control y menos en la ejecución. Es un estándar enfocado a la Gobernanza de la Seguridad de la Información y no a la Gestión de servicios (von Solms, 2005).

El modelo CMMI es una fusión de modelos de mejora de procesos para ingeniería de sistemas, ingeniería del software, desarrollo de productos integrados y adquisición de software. La gran ventaja de CMMI es que ha demostrado ser una metodología de gran eficacia, que ha permitido mejoras de gran impacto en procesos de desarrollo de productos software, tales como reducción del coste de desarrollo, localización y resolución de defectos, entre otras. El gran problema de CMMI es su falta de adecuación al enfoque de los servicios que está experimentando el sector de las TI en todas sus líneas de actividad, así como el alto esfuerzo de implantación que exige (Sánchez, 2014).

La ISO 20000 ofrece la posibilidad de implantar en su organización los requerimientos de este prestigioso estándar internacional en gestión de servicios de TI, y obtener la certificación correspondiente que otorgará a la organización una ventaja competitiva en el mercado. Sin embargo este estándar no consta de un modelo de arquitectura de información de los datos corporativos y los sistemas de información asociados a ellos. La ISO 20000 está enfocada a medir la calidad de los servicios a partir de su certificación, basándose en ITIL para realizar la Gestión de los servicios.

ITIL está orientado a los procesos de la Gestión de servicios y abarca todas las áreas de su desarrollo. Las fases de estrategia y de diseño se enfocan en el área de planificación, las fases de operación y transición al área del desarrollo y la fase de mejora apoya la calidad del servicio. ITIL por medio de procedimientos, roles, tareas, y responsabilidades que se pueden adaptar a cualquier organización de TI, genera una descripción detallada de mejores prácticas, que permitirán tener mejor comunicación y administración en la organización de TI. Proporciona los elementos necesarios para determinar objetivos de mejora y metas que ayuden a la organización a madurar y crecer. Los demás estándares aunque son los mejores en cada área no permiten ser usados de manera completa para gestionar un servicio. La utilización de ITIL en el Centro de Soporte UCI para realizar los procesos de Gestión de incidencias y Gestión de cambios condiciona su empleo para realizar la Gestión de problemas.

Ilustración 1: Interacción entre procesos.

1.3. ITIL

En la década de los 80 el gobierno británico brindaba un conjunto de servicios de TI. Con el fin de aumentar el nivel de calidad de los servicios, le encargó a la Agencia Central de Informática y Telecomunicaciones (CCTA[5]) el desarrollo de un marco de referencia para el uso eficiente de recursos de TI. Este marco debería ser aplicable tanto en la administración como en las empresas privadas. La primera versión se denominó Gobierno de Tecnología de la Información de Gestión de Infraestructuras (GITIM [6]). Esta versión se utilizó ampliamente por la administración y grandes empresas en Europa, con lo cual a principios de los años 90 se desarrolló una nueva versión, ya denominada ITIL (APM Group Ltd, 2013). En el 2001, se comienza a liberar la versión 2 de ITIL. En junio de 2007 se publica la versión 3, que adopta un enfoque orientado al ciclo de vida del servicio, con un mayor énfasis en la integración con el negocio de TI. En el año 2011 se publica una nueva versión donde se mantienen características similares con la versión 3 pero adicionando algunas mejoras y cambiando

[5] Del inglés: *Central Computer and Telecommunications Agency.*
[6] Del inglés: *Government Information Technology Infrastructure Management.*

mucho la parte comercial de la biblioteca. En la Ilustración 2 se muestra la evolución que ha tenido ITIL.

Ilustración 2: Evolución de ITIL (OSIATIS S.A, 2011)

ITIL es un conjunto de publicaciones de mejores prácticas para la gestión de servicios de TI. Proporciona guías de calidad para la prestación de servicios de TI y los procesos, las funciones y otras competencias necesarios para sustentarlas. El marco de trabajo ITIL se basa en el ciclo de vida del servicio que consta de cinco etapas (Estrategia del servicio, Diseño del servicio, Transición del servicio, Operación del servicio y Mejora continua del servicio), cada una de ellas tiene su propia publicación de apoyo. También hay una serie de publicaciones complementarias de ITIL que proporcionan orientación específica para sectores de la industria, tipos de organización, modelos operativos y arquitecturas de tecnología (APM Group Ltd, 2011).

“Las mejores prácticas de la Gestión de servicios de TI han evolucionado desde 1989, año en que se publicaron las primeras librerías de ITIL. Su utilización se soporta mediante una certificación y una estructura de formación que han sido adoptadas a nivel mundial para reconocer la competencia profesional en Gestión de servicios de TI.” (Sun Microsystems, 2013).

“ITIL es el enfoque más aceptado en relación a la GSTI en todo el mundo. Proporciona un conjunto coherente de buenas prácticas, derivado de los sectores públicos y privados a nivel internacional. Está acompañado por un amplio esquema de calificación, por organizaciones de

formación acreditadas y herramientas de implementación y evaluación. Las necesidades que presentan las empresas y su enfoque directo en los procesos de negocio convierten a ITIL en el marco que les permite identificar, planificar, entregar y apoyar los servicios de la TI, persiguiendo como meta facilitar los cambios estructurales y el crecimiento en los beneficios." (OGC ITIL, 2011).

Según lo expuesto por (itSMF Ltd, 2007) la adopción de ITIL puede ofrecer a los usuarios una amplia gama de beneficios que incluyen:

- Aumento de la satisfacción de usuarios y clientes de los servicios TI.
- Mejora de la disponibilidad del servicio, lo que lleva directamente a un aumento de los beneficios y los ingresos del negocio.
- Ahorros financieros atendiendo a la reducción del doble trabajo y la pérdida de tiempo, la mejora del uso y gestión de recursos.
- Mejora del tiempo dedicado al marketing de nuevos productos y servicios.
- Mejora de la toma de decisiones y el riesgo optimizado.

ITIL se centra en brindar servicios de alta calidad para lograr la máxima satisfacción del cliente a un costo manejable. "Para ello, parte de un enfoque estratégico basado en el triángulo procesos-personas-tecnología." (Sun Microsystems, 2013). En otras palabras: determina la forma de ejecución de procesos estándares apoyados en la tecnología para lograr la satisfacción de los usuarios de los servicios de TI.

ITIL estructura la gestión de los servicios TI sobre el concepto de Ciclo de Vida de los Servicios. Este enfoque tiene como objetivo ofrecer una visión global de la vida de un servicio desde su diseño hasta su eventual abandono, sin por ello ignorar los detalles de todos los procesos y funciones involucrados en la eficiente prestación del servicio.

Este ciclo consta de cinco fases, dentro de las cuales recae el mayor nivel de criticidad sobre la fase de Operación de servicio la cual es la encargada de la implementación de nuevos servicios y la mejora de los ya existentes. En esta fase se persiguen como objetivos fundamentales la coordinación e implementación de todos los procesos, actividades y funciones. Consta de distintos procesos donde se monitorizan los eventos que acontezcan en la infraestructura de las TI. Durante esta fase se lleva a cabo el control sobre las incidencias que por su frecuencia o impacto degradan la calidad del producto. Además ofrece soluciones capaces de mitigar los efectos no deseados provocados por las incidencias, el proceso de Gestión de problemas lleva a cabo el progreso de esta actividad.

1.3.1. La nueva versión desde 2011

En 2011 se editó una nueva versión de ITIL, totalmente revisada y mejorada: “ITIL 2011”. ITIL 2011 recoge las experiencias de las versiones anteriores. Se centra en apoyar el negocio base de las empresas e intentar que consigan a largo plazo ventajas sobre la competencia, mejorando la labor de la organización. Consta de cinco publicaciones básicas que reproducen conjuntamente el Ciclo de Vida del Servicio (Molina, 2011):

- Estrategia del servicio (Service Strategy).
- Diseño del servicio (Service Design).
- Transición del servicio (Service Transition).
- Operación del servicio (Service Operation).
- Mejora continua del servicio (Continual Service Improvement).

En ITIL 2011, los procesos ya conocidos de ITIL v3 se complementan con numerosos procesos nuevos. Estas novedades se caracterizan por una mayor orientación al cliente a la hora de ofrecer servicios de TI. Sin embargo los principios sobre los cuales se basa ITIL se mantienen prácticamente inalterados en este enfoque.

Ilustración 3: Ciclo de vida de los servicios (Tecnofor, 2013)

Todas las fases del Ciclo de Vida del Servicio tienen como objetivo último que los servicios sean correctamente prestados aportando el valor y la utilidad requerida por el cliente con los niveles de calidad acordados (OSIATIS S.A, 2011). Según la perspectiva empresarial, los

servicios de TI se encuentran condicionados a un ciclo de vida típico, que comienza con la introducción del servicio al mercado y finaliza con su exclusión del portafolio de servicios. Cada una de las cinco disciplinas principales de ITIL está enfocada a una fase específica dentro del Ciclo de Vida del Servicio (APM Group Ltd, 2013):

- En el marco de la Estrategia del servicio se determina qué clase de servicios deben ofrecerse a determinados clientes y/o mercados.
- En la fase del Diseño del servicio se determinan los requisitos concretos, se ocupa de desarrollar soluciones adecuadas a estos requisitos, de proyectar nuevos servicios y de modificar y/o mejorar los ya existentes.
- En la fase de la Transición del servicio se amplían y extienden los servicios nuevos o modificados.
- La Operación del servicio se encarga de realizar todas las tareas operacionales que se vayan presentando.

Según el sitio (OSIATIS S.A, 2011) la fase de Operación del servicio es la más crítica entre todas. La percepción que los clientes y usuarios tengan de la calidad de los servicios prestados depende en última instancia de una correcta organización y coordinación de todos los agentes involucrados. Los principales procesos asociados directamente a la fase de Operación del servicio son:

- **Gestión de eventos:** es el responsable de monitorizar todos los eventos que acontezcan en la infraestructura de la TI con el objetivo de asegurar su correcto funcionamiento y ayudar a prever futuras incidencias.
- **Gestión de incidencias:** es el responsable de registrar todas las incidencias que afectan la calidad del servicio y restaurarlo a los niveles acordados de calidad en el menor tiempo posible.
- **Petición de servicios de TI:** es el responsable de gestionar las peticiones de usuarios y clientes que habitualmente requieren pequeños cambios en la prestación del servicio.
- **Gestión de problemas:** es el responsable de analizar y ofrecer soluciones a aquellos incidentes que por su frecuencia o impacto degradan la calidad del servicio.
- **Gestión de acceso a los servicios TI:** es el responsable de garantizar que sólo las personas con los permisos adecuados pueda acceder a la información de carácter restringido.

1.3.2. Gestión de problemas

Según el glosario de términos (APM Group Ltd, 2011) una falla es “la pérdida de la capacidad para operar de acuerdo con las especificaciones, o para entregar los resultados requeridos”. A menudo, una falla produce un incidente. Plantea además que un incidente es “una interrupción no planificada de un servicio de TI o la reducción en la calidad de un servicio de TI. La falla de un elemento de configuración que no ha afectado aún el servicio es también un incidente”. En otro ámbito expone que la Gestión de incidencias o incidencias se define como “el proceso responsable de la gestión del ciclo de vida de todos los incidentes”. La Gestión de incidencias asegura que se restablezca la operación normal del servicio lo antes posible y se minimice el impacto al negocio.

En la Gestión de incidencias, si el incidente es repetitivo o por su impacto afecta la estabilidad del servicio se convierte en un problema y por tanto pasa a la Gestión de problemas, donde se investigan las causas del problema hasta conocer su origen. El conocimiento de las causas del problema posibilita generar una solución temporal y el problema sufre un cambio, una mutación, pasa de ser un problema a un error conocido.

Como se muestra en la Ilustración 4, la Gestión de problemas inicia con la entrada de un incidente, el cual sirve de base para generar el problema. Acto seguido se determinan las causas para luego buscar la solución en la base de soluciones o en la base de datos de errores conocidos. La búsqueda de solución se detalla en la Ilustración 5. Al concluir la búsqueda si se ha encontrado la solución se le asigna al problema, si no se remite el problema al proveedor y se espera que este proponga una solución. Una vez asignada la solución al problema se envía la respuesta al cliente y se espera por la aprobación o el rechazo de la solución enviada. Si es aceptada la solución, se agrega el problema, las causas y su solución a la base de errores conocidos contribuyendo al proceso de Gestión de conocimiento. De lo contrario, si la solución no resuelve el problema, el usuario la rechaza reiniciando el proceso desde la actividad “Buscar solución”.

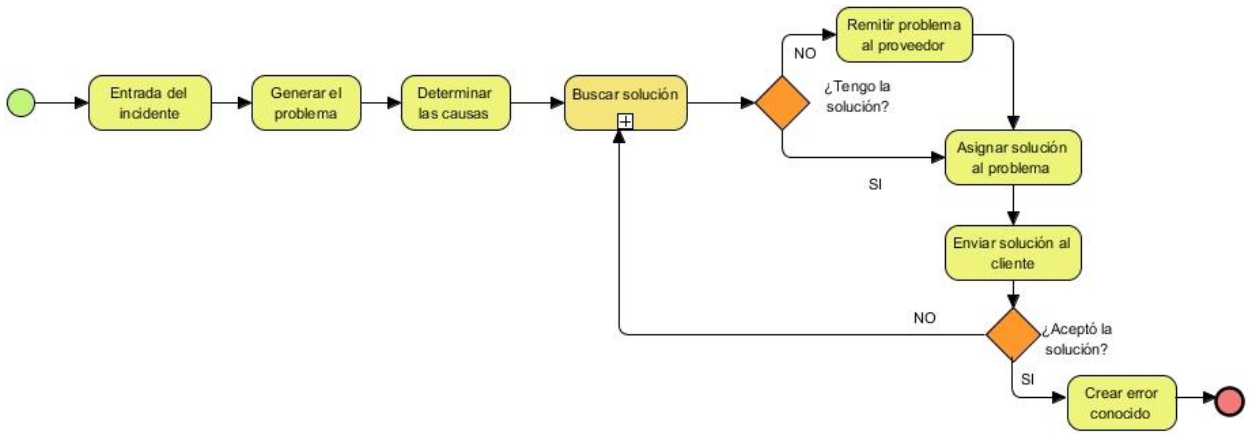


Ilustración 4: Proceso de Gestión de problemas.

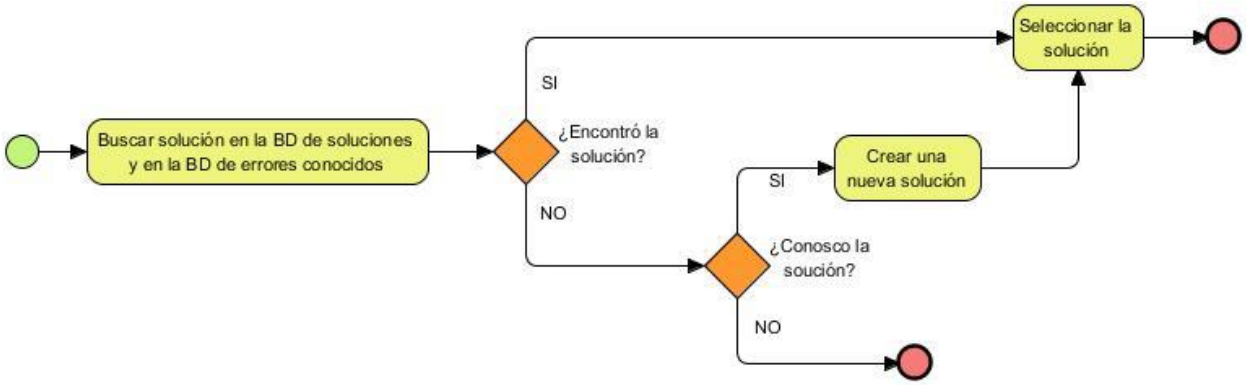


Ilustración 5 Búsqueda de la solución.

Como se evidencia, la Gestión de incidencias y la Gestión de problemas están estrechamente relacionadas. La relación consiste en la derivación de los problemas a partir de las incidencias. La Gestión de problemas no está enfocada a todos los incidentes que ocurren en la organización. La Gestión de incidencias no debe confundirse con la Gestión de problemas, pues a diferencia de esta última, no identifica y analiza las causas subyacentes a un determinado incidente sino exclusivamente restaura el servicio.

1.3.3. Modelo conceptual

El término “modelo”, en el lenguaje común, es extremadamente variable. Como muestra representativa de la variedad de acepciones, a continuación se exponen las definiciones propuestas por el Webster’s New Encyclopedic Dictionary[7]:

1. Copia exacta pero a escala de algo.

[7] El nombre de Webster’s New Encyclopedic Dictionary se refiere a cualquiera de la línea de diccionarios desarrollados por Noah Webster en el siglo XIX los cuales continúan siendo actualizados.

2. Patrón o figura de algo pendiente de hacerse.
3. Descripción o analogía usada para ayudar a visualizar algo que no puede ser directamente observado.
4. Sistema de asunciones, datos e interfaces usados para describir matemáticamente objetos o situaciones.
5. Proyección teórica de un sistema posible o imaginario.

Los modelos que favorecen la comprensión de la necesidad del usuario y los requisitos del software se denominan modelos conceptuales, y son usados durante la actividad de análisis del proceso de requisitos (LSIS, 2004). A continuación se muestra el modelo conceptual del proceso de Gestión de problemas en el Centro de Soporte. En la Ilustración 6 se exponen los principales conceptos así como sus relaciones. Se evidencia la relación existente entre el incidente que inicia el proceso y el problema que se genera. Así como la relación de la tripleta problema, solución y causa que conforman el error conocido.

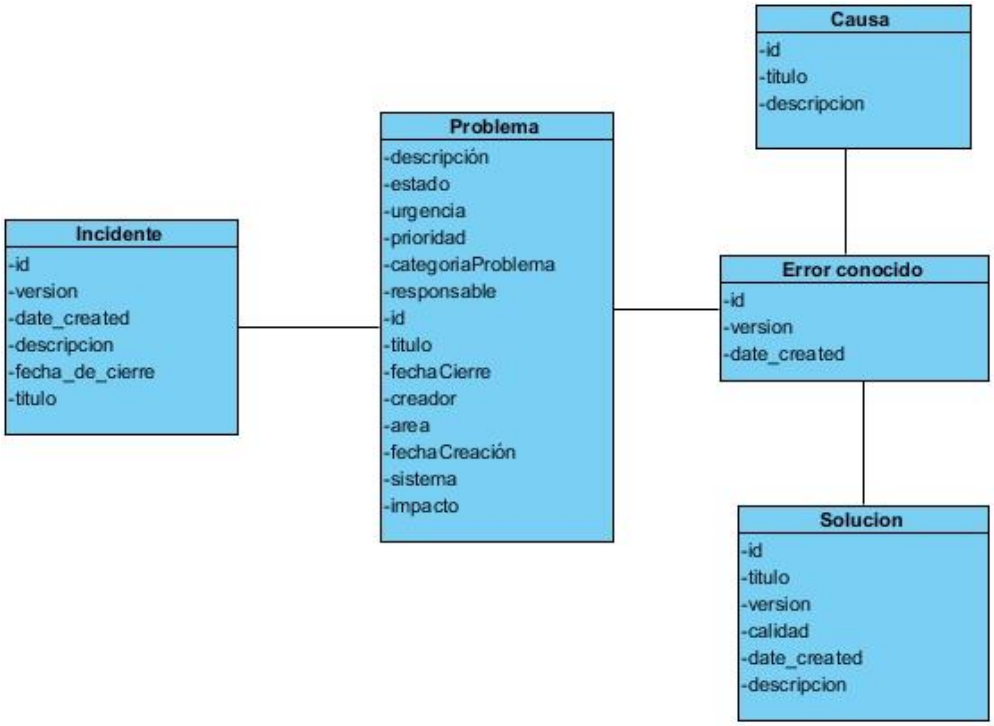


Ilustración 6: Modelo conceptual

1.4. Análisis de soluciones existentes

Los Sistemas para la Gestión de problemas e Incidencias constituyen un conjunto de componentes relacionados entre sí, que deben su existencia a las necesidades de servicios informáticos y al factor humano que exige cada vez más calidad en su gestión. Entre las

tecnologías más populares asociadas a este fenómeno, se encuentran los Service Desk, Help Desk[8] y Call Center[9]. Para realizar la búsqueda de una propuesta de solución que realice las Gestión de problema se tienen tres principios fundamentales. Primero: debe ser un Service Desk o Help Desk de soporte. Segundo: debe implementar la Gestión de problemas basándose en ITIL. Siguiendo la política de la UCI de potenciar el uso de software libre se define el tercero y último criterio: debe ser de código abierto (UCI, 2014).

1.4.1. Ámbito internacional

En el ámbito internacional la búsqueda arrojó como resultado el OneOrZero TMS basado en un sistema de tickets vía web. Pertenece al grupo de tecnologías conocidas como Help Desk que proporcionan distintas características para el manejo de tareas de negocio. Es robusto, rápido, altamente personalizable y compatible con cualquier plataforma virtual. Se rige por la Licencia Pública General (GPL[10]) de código abierto. Los desarrolladores de este sistema ofertan servicios técnicos y personalizados a los clientes basados en su experiencia en el trabajo en equipo. A pesar de ser totalmente libre, las últimas versiones y mejoras del producto son reservadas para la asociación de miembros de desarrollo de este Service Desk. Para hacerse miembro se necesita abonar cierta cantidad de dinero que es usado para costear el desarrollo de OneOrZero y en dependencia de esta cantidad son los privilegios otorgados (Losada Leon, y otros, 2009).

1.4.2. Ámbito nacional

En Cuba, la Gestión de servicios no se ha materializado a gran escala en todas las empresas e instituciones, pero se tienen referencias de entidades como ETECSA y DESOFT que utilizan este servicio tan importante y enriquecedor de conocimientos (Sánchez Ortiz, y otros, 2012). NovaDesk 2.0 es un Service Desk realizado en la UCI con el fin de brindar soporte a los usuarios del Sistema Operativo Nova. Está basado sobre las características del OneOrZero TMS del cual hereda la licencia GPL. Dentro de sus características fundamentales se encuentran (Sánchez Ortiz, y otros, 2012):

- Multi-entidades de gestión y de usuarios.
- Sistema de autenticación de múltiples servidores (local, LDAP, AD).

[8] Es un conjunto de recursos tecnológicos y humanos, para prestar servicios con la posibilidad de gestionar y solucionar todas las posibles incidencias de manera integral.
[9] Es un área donde agentes, asesores, supervisores o ejecutivos, especialmente entrenados, realizan llamadas y/o reciben llamadas desde o hacia: clientes, socios comerciales y compañías asociadas.
[10] Del inglés: *General Public License.*

- Gestión multilingüe (22 idiomas disponibles).
- Permisos y perfiles de sistema.
- Módulo de búsqueda compleja.
- Sistema de exportación en pdf, csv y slk.
- Sistema de notificaciones de eventos (material fungible, la expiración de los contratos y licencias).
- Sistema de verificación de actualizaciones.
- Visualización de estadísticas por medio de informes a modo global, por técnico o empresa, hardware, ubicación o el tipo. Por usuario, categoría y por prioridad.

A pesar de implementar la Gestión de servicios basándose en ITIL está enfocado fundamentalmente a la prestación de servicios a los usuarios de un sistema operativo.

1.4.3. Análisis crítico

El Centro de Soporte UCI necesita un nuevo Service Desk para sustituir la herramienta propietaria que utiliza. Uno de sus especialistas propuso como parte de su tesis doctoral el desarrollo de la mencionada aplicación en el marco de trabajo Grails 2.1.1. Propone además utilizar el estándar ITIL para realizar la Gestión de servicios. Como parte de la investigación del aspirante a doctor se han implementado los módulos de Gestión de incidencias, Gestión de cambio, Gestión de acceso a los servicios de TI, Gestión de mapeo entre otros. Para continuar l desarrollo de manera uniforme y evitar problemas de compatibilidad es necesario que el nuevo módulo de Gestión de problemas continúe con la utilización de Grails.

Se necesita que si existe alguna solución, sea de código abierto permitiéndole a sus especialistas modificar su estructura acorde a su sistema interno de trabajo. Debe estar basada en ITIL. Además se necesita que sea modular, de modo que se pueda separar la función de Gestión de problemas como un proceso independiente. Es de suma importancia que esté implementada en Grails 2.1.1 ya que existen otros módulos implementados en este marco de trabajo. Por último el sistema debe estar disponible en idioma español. A continuación se analizan las soluciones encontradas a partir de los criterios establecidos y se verifica que cumplan con estos parámetros.

Tabla 1: Evaluación de las soluciones encontradas

Nombre	Modular	ITIL	Idioma	Grails 2.1.1
OneOrZero TMS	No	Si	Español	No
Nova Desk 2.0	No	Si	Español	no

Estas propuestas están implementadas en diversas versiones de lenguajes de programación, y se publican pocas especificaciones técnicas de su implementación. Teniendo en cuenta que aun cuando utilizan ITIL en sus procesos y manejan modernas técnicas y tecnologías, ninguna de las propuestas antes planteadas se ajusta a las necesidades del Centro de Soporte UCI. De este modo se evidencia la necesidad de desarrollar un módulo de Gestión de problemas para el Centro de Soporte UCI.

1.5. Herramientas y Tecnologías

Seguidamente se realiza un análisis de la metodología de desarrollo de software y las herramientas a utilizar.

1.5.1. Metodología para el desarrollo

Según lo expuesto por (Abrahamsson, y otros, 2002) desarrollar un buen software depende de un elevado número de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo, en un determinado proyecto es trascendental para el éxito del producto. El desarrollo de software no es una tarea fácil, prueba de ello es que existen numerosas propuestas metodológicas tanto ágiles como tradicionales que inciden en distintas dimensiones del proceso de desarrollo.

Existen dos clasificaciones de metodologías que llevan a los equipos de desarrollo a obtener una mejor documentación de todo el proceso o a sacrificar esta por un proceso más ágil, llamadas Metodologías Ágiles y Metodologías Pesadas. Dada la necesidad de llevar a cabo el desarrollo de la propuesta de solución en un corto periodo de tiempo, manteniendo la colaboración constante con el cliente y la necesidad de garantizar la flexibilidad necesaria frente a la variación de los requisitos, se hace necesario optar por un enfoque ágil de desarrollo de software en lugar de un enfoque pesado.

Según lo planteado por (Figueroa, y otros, 2009) las principales líneas de las metodologías ágiles son:

- Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.

Según lo planteado por (Beck, 1999), (Jeffries, y otros, 2001) y (Abrahamsson, y otros, 2002) entre las principales metodologías ágiles se encuentran Programación Extrema (XP[11]), Iconix, SCRUM, Cristal Methods, Proceso Unificado Ágil (AUP [12]) y otras. Estas metodologías sobreponen la capacidad de respuesta a un cambio sobre el seguimiento estricto de un plan, dado el interés que supone para muchos clientes esta elasticidad.

Grails es un entorno para desarrollo de aplicaciones web sobre la plataforma Java nacida en un contexto muy particular: el de las metodologías de desarrollo de software ágiles. La idea detrás de estos procesos es que los requisitos de una aplicación no siempre pueden definirse completamente antes de comenzar la implementación, y es necesario un ciclo basado en interacciones cortas y una comunicación muy fluida con el cliente. De esta manera, metodologías como eXtreme Programming o Scrum establecen una forma de trabajo en la que la comunicación fluida garantiza que si se produce un cambio total o parcial en los requisitos el equipo de desarrollo será capaz de adaptar la aplicación en un tiempo razonable (Brito Calahorro, 2009).

eXtreme Programming

"Más que una metodología, XP se considera una disciplina, la cual está sostenida por valores y principios propios de las metodologías agiles. Existen 4 valores que sirven como pilares en el desarrollo de estas metodologías" (Echeverry Tobón, y otros, 2007).

- **La comunicación:** en XP la relación con el cliente es tan estrecha, que es considerado parte del equipo de desarrollo.
- **La simplicidad:** no utiliza recursos para la realización de actividades complejas, solo se desarrolla lo que el cliente demanda, de la forma más sencilla.

[11] eXtreme Programming
[12] Del inglés: *Agile Unified Process*

- **La retroalimentación:** está presente desde el principio del proyecto, ayuda a encaminarlo y darle forma.
- **El coraje:** el equipo de desarrollo debe estar preparado para enfrentarse a los continuos cambios que se presentarán en el transcurso de la actividad.

Además de estos valores XP centra su desarrollo en 12 prácticas específicas que la caracterizan. Estas son (Echeverry Tobón, y otros, 2007):

- El juego de la planificación.
- Entregas pequeñas.
- Utilización de metáforas del sistema.
- Diseño simple.
- Pruebas.
- Refactorización.
- Programación por parejas.
- Propiedad colectiva del código.
- Integración continúa del código.
- 40 horas por semana.
- Cliente in-situ.
- Estándares de programación.

Scrum

Scrum es un marco de trabajo de procesos que ha sido utilizado para gestionar el desarrollo de productos complejos desde principios de los años 90. Scrum no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas. Scrum hace patente la eficacia relativa de las prácticas de gestión de producto y de desarrollo, de modo que puedan ser mejoradas (Schwaber, 2011). Además es ligero, fácil de entender y extremadamente difícil de llegar a dominar.

Scrum se fundamenta en la teoría de que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Emplea una aproximación iterativa e incremental para optimizar la predictibilidad y controlar el riesgo.

La implementación del control empírico de procesos se basa en tres pilares (Schwaber, 2011):

- **Transparencia:** todo lo que afecta al resultado del trabajo debe ser conocido y visible para todo el mundo.

- **Inspección:** es necesario realizar un seguimiento de la evolución del trabajo para detectar desviaciones con respecto a lo previsto.
- **Adaptación:** en el momento en el que se detecta alguna desviación o se tienen indicios de que esta se puede producir, es necesario actuar en consecuencia para adaptarse a las nuevas circunstancias.

Scrum plantea la necesidad de conocer profundamente cada uno de los procesos que intervendrán en el desarrollo del software. En el caso de la presente investigación este requisito no se cumple, pues el equipo de desarrollo no cuenta con una vasta experiencia en el trabajo con las tecnologías y los procesos asociados a la construcción de la herramienta que se propone. Teniendo en cuenta lo antes planteado se escoge XP como metodología de desarrollo, debido a su capacidad de comunicación, simplicidad y reutilización del código, características que facilitarán llevar a cabo el proceso de implementación de la solución propuesta.

1.5.2. Herramientas, tecnologías y lenguajes

A continuación se describen las principales herramientas, tecnologías y lenguajes, entre otros recursos seleccionados por políticas del Centro de Soporte UCI.

1. Intellij IDEA

Como Entorno de Desarrollo Integrado (IDE[13]) se utiliza IntelliJ IDEA en su versión 12.1.3 el cual brinda generación de código basado en la información sobre las clases. El editor comprende funciones de apoyo integrado en la refactorización, herramientas de análisis estructural y el código de prueba de errores lógicos. Cuenta con una función de pruebas unitarias con JUnit[14], tiene soporte completo para Java y para distintos complementos. (JetBrains s.r.o., 2013).

2. Apache Tomcat

Para la publicación de servicios web es necesaria la utilización de servidores de aplicaciones web, con ese fin se utiliza Apache Tomcat. Este servidor es una implementación de código abierto del software de Java Serlet y Java Server Pages. Es uno de los servidores de aplicaciones web más utilizados con más de un millón de descargas al mes y más del 70% de penetración en el centro de datos empresarial. Apache Tomcat se utiliza para crear desde

[13]Del inglés: *Integrated Development Environment.*
[14]Marco de trabajo que se utiliza para hacer pruebas unitarias de aplicaciones Java.

simples sitios web en un servidor, hasta grandes redes entre empresas (The Apache Software Foundation , 2013).

3. Groovy and Grails

Groovy es un lenguaje alternativo para la Máquina Virtual de Java, alternativo porque es posible el uso de Groovy en la plataforma de Java casi de la misma forma que se usa el propio código Java. El lenguaje Groovy combina correctamente con el lenguaje Java al desarrollar nuevas aplicaciones, de igual manera puede ser utilizado para la ampliación de aplicaciones existentes. Groovy actualmente se encuentra en su versión 1.5.4 y se encuentra disponible para las plataformas Java desde la versión 1.4 hasta la 1.7.

Una característica de Groovy es que su sintaxis es muy similar a la sintaxis de Java lo que posibilita una curva de aprendizaje pequeña. Sin embargo la principal diferencia entre el lenguaje Groovy y Java es que Groovy permite escribir menos código para realizar las tareas (GoPivotal, Inc, 2013).

Grails es un marco de trabajo web para la plataforma Java que se basa en el lenguaje dinámico Groovy. Mediante el uso de Lenguajes de Dominio Específico (DSLs[15]) potentes pero a la vez sencillos, con la versatilidad de Groovy, y un ecosistema de complementos que mejora la productividad en un conjunto cada vez más amplio de escenarios, Grails se ha hecho inmensamente popular, y un motor de cambio en el espacio Java (GoPivotal, Inc, 2013).

Este marco de trabajo permite disminuir el tiempo de desarrollo de un proyecto de software. En comparación con el típico marco de trabajo de Java, con Grails se necesita menos código para obtener el mismo resultado. Menos código significa menos errores y menos líneas de código de mantener (GoPivotal, Inc, 2013).

4. PostgreSQL

PostgreSQL es un manejador y gestor de bases de datos Objeto-Relacional. Comenzó como un proyecto llamado Ingres en la Universidad de California en Berkeley. Ingres fue desarrollado de manera comercial por Relational Technologies/Ingres Corporation.

Entre las características de PostgreSQL se encuentran (The PostgreSQL Global Development Group , 2013):

- Es un gestor de datos Objeto-Relacional.
- Altamente extensible.

[15] Del inglés: *Domain Specific Languages.*

- Soporte comprensivo para el SQL[16].
- Integridad referencial.
- Interfaz de programación de aplicaciones (API[17]) sumamente flexible.
- Lenguajes procedimentales.
- Control de Concurrencia Mediante Versiones Múltiples (MVCC[18]).
- Arquitectura Cliente/Servidor.

PostgreSQL es ampliamente popular e ideal para aplicaciones web. Tiene una sintaxis QL[19] estándar y fácil de entender. Es multiplataforma y fácil de administrar. PostgreSQL presenta como desventaja que para bases de datos relativamente pequeñas la velocidad de respuesta no es muy eficiente en comparación con otras relativamente grandes.

5. GrailsRendering

Atendiendo a lo expuesto por (GoPivotal, Inc, 2013), GrailsRendering es un complemento que presta servicios para generar reportes. Este complemento añade capacidades de renderización adicionales para aplicaciones Grails a través de la biblioteca de dibujado XHTML[20]. Permite generar reportes procesando plantillas y obteniendo documentos en diferentes formatos.

Dentro de los servicios disponibles se encuentran:

- pdfRenderingService: prestación de servicios para procesar plantillas a pdf[21].
- gifRenderingService: prestación de servicios para procesar plantillas a gif[22].
- pngRenderingService: prestación de servicios para procesar plantillas a png[23].
- jpegRenderingService: prestación de servicios para procesar plantillas a jpeg[24].

6. Twitter Bootstrap

Analizando lo expuesto por (Fontela, 2013) y por (Mozilla, 2014) Twitter Bootstrap es una colección de herramientas de software libre para la creación de sitios y aplicaciones web. Contiene plantillas de diseño basadas en HTML[25] y CSS[26] con tipografías, formularios, botones,

[16] Del inglés: Structured Query Language.
[17] Del inglés: Application Programming Interface.
[18] Del inglés: *Multiversion Concurrency Control*
[19] Del inglés: Query Language
[20] Del inglés: *eXtensible HyperText Markup Language.*
[21] Del inglés: *Portable Document Format.*
[22] Del inglés: *Graphics Interchange Format*.
[23] Del inglés: *Portable Network Graphics.*
[24] Estándar de compresión y codificación digital de imágenes.
[25] Del inglés: *HyperText Markup Language.*
[26] Del inglés: *Cascading Style Sheets.*

gráficos, barras de navegación y otros componentes de interfaz, así como extensiones opcionales de JavaScript. Fue desarrollado por Mark Otto y Jacob Thornton miembros de Twitter, como un marco de trabajo para fomentar la consistencia a través de herramientas sencillas. Es de código abierto y compatible con la mayoría de los navegadores web.

7. JQuery

jQuery es una librería de JavaScript rápida y concisa que simplifica el uso del documento HTML, el manejo de eventos, la animación, y las interacciones AJAX[27] para el desarrollo web rápido (The jQuery Foundation, 2014). Además de ser muy fácil de usar, una de las mayores ventajas de jQuery es que maneja una gran cantidad de "estándares" para varios navegadores web. En la actualidad esta librería es ampliamente utilizada en disímiles aplicaciones web. Su reputación ha crecido exponencialmente al igual que la comunidad de desarrolladores que comparten sus experiencias así como plugins que complementan el trabajo con la librería. Esta herramienta será utilizada fundamental para el trabajo en las vistas por su integración con las GSP[28] generadas por el marco de trabajo. Así como para realizar peticiones asincrónicas al servidor utilizando AJAX.

1.6. Conclusiones del capítulo

Durante el desarrollo del capítulo se realizaron una serie de análisis que permitieron identificar que las soluciones existentes para la Gestión de problemas no pueden ser utilizadas debido a los requerimientos y necesidades del Centro de Soporte UCI. La imposibilidad de la aplicación o adaptación de alguna de estas herramientas evidenció la necesidad de la creación de un módulo que permita realizar la Gestión de problemas en el centro.

Las herramientas y tecnologías a utilizar en el desarrollo del módulo se encuentran subordinadas a una investigación más amplia, de la cual el presente trabajo forma parte. El lenguaje de programación a emplear es Groovy y el marco de trabajo a utilizar es Grails. Para el trabajo con las vistas se empleará Twitter Bootstrap. El servidor de aplicaciones a emplear es Apache Tomcat, mientras que se utilizará PostgreSQL como sistema de gestión de bases de datos. La metodología de desarrollo a emplear es XP. El estudio de estas herramientas y tecnologías le permitió al equipo de desarrollo familiarizarse con el marco de trabajo, el lenguaje de programación y la librería JQuery. Además, permitió vincular los conocimientos previos de Twitter Bootstrap y PostgreSQL con el marco de trabajo.

[27] Acrónimo de: *Asynchronous JavaScript And XML*.
[28] *Groovy Server Page*.

Capítulo 2: Análisis y diseño de la solución

2.1. Introducción

En el presente capítulo se detalla la propuesta de solución presentando sus principales características. Se lleva a cabo el proceso de desarrollo especificando cada uno de los procesos que intervienen en las distintas fases de la metodología seleccionada. Se define el cronograma de iteraciones y el plan de entrega para ajustar el tiempo de la implementación.

2.2. Propuesta de solución

Utilizando la información recopilada en el capítulo precedente, se propone el desarrollo de un módulo que implemente la Gestión de problemas como parte del proceso de Gestión de servicios del Centro de Soporte UCI. La solución informática debe ser capaz de centralizar los problemas, las soluciones y los errores conocidos en una base de datos que brinde un servicio integrado de soporte. Debe además permitirle a los especialistas y técnicos del centro valorar un problema generado y asignarle un impacto y una categoría. El sistema debe permitir asignarle a cada problema un conjunto de soluciones y de igual modo registrar un conjunto de causas.

La autenticación de los usuarios en el sistema debe realizarse utilizando las credenciales asignadas por los servicios telemáticos de la UCI. Asimismo debe brindar un medio para la administración de los usuarios locales. Adicionalmente, la solución debe contar con un sistema de notificaciones basado en las responsabilidades asignadas según el rol del usuario y atendiendo a su intervención en el ciclo de vida del problema. Incluyendo además un sistema de alertas por correo. Este sistema de notificaciones facilitará la comunicación automática del proceso interno de trabajo. Cada problema tiene asociado el usuario que generó el reporte, el cual será notificado cuando el problema pase al estado de solucionado.

2.3. Fase de exploración

“En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema”. (Padanés, et al., 2007)

2.3.1. Metáfora del sistema

Según conceptualiza (Padanés, y otros, 2007) “en XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. La arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema.”

La metáfora definida para el Sistema de Gestión de problemas es: *“Este sistema funciona como un taller de reparaciones, donde llegan equipos descompuestos y son reparados”.* En este caso los *equipos* se refieren a los distintos servicios que brinda el Centro de Soporte, la palabra *descompuestos* a los distintos incidentes de funcionamiento que puedan tener y por último las *reparaciones* a las soluciones que serán dadas por el sistema corrigiendo de esta forma el defecto reportado por los clientes.

2.3.2. Historias de usuario

Las Historias de Usuario (HU) son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer. El tratamiento de las historias de usuario es muy dinámico y flexible. En cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en pocas semanas. Por el elevado número de HU (44) el resto no se incluyen en el presente documento, pueden ser consultadas en el expediente de proyecto *Entregables* con el nombre **Historias de usuario GESERV.doc.**

Tabla 2: Historia de Usuario HU14

Historia de Usuario	
Número: HU14	**Nombre Historia de Usuario:** Asignar una categoría como subcategoría de otra categoría.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Neybis Lago	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 0.4

Riesgo en Desarrollo: Medio	**Puntos Reales:** 0.4
Descripción: La presente historia de usuario tiene como objetivo permitir asignar una categoría como subcategoría de otra categoría.	
Observaciones: Deben existir dos categorías, la primera sin asignar y la segunda asignada a otra categoría o siendo una raíz de categorías.	
Prototipo de interfaz: 	

2.4. Requisitos del sistema

De la palabra “requisito” existen numerosas definiciones, a continuación se presenta la que aparece en el glosario de la IEEE[29]:

1. Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
2. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
3. Una representación documentada de una condición o capacidad como en los puntos anteriores (1) ó (2).

Según plantea (Pérez Huebe, 2005) existen dos tipos de requerimientos los funcionales y los no funcionales a continuación se especifican cada uno de ellos para la solución propuesta.

2.4.1. Requisitos funcionales del software

Los requisitos funcionales “definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas, especifican los servicios que debe proporcionar la aplicación (por ejemplo: la aplicación debe calcular el valor del portafolio de inversión del usuario)” atendiendo a lo expuesto por (Sommerville, 2005). En la Tabla 3 se muestran los requisitos funcionales del sistema.

[29] Del inglés: *Institute of Electrical and Electronics Engineers.*

Tabla 3: Requisitos funcionales

No.	Requisito	No.	Requisito
RF1	Autenticar usuarios.	RF30	Generar reportes en formato pdf.
RF2	Insertar un nuevo usuario en el sistema.	RF31	Generar los informes de soluciones por categorías.
RF3	Eliminar un usuario del sistema.	RF32	Determinar soluciones.
RF4	Modificar un usuario existente en el sistema.	RF33	Visualizar la cantidad de soluciones por usuario.
RF5	Listar los usuarios del sistema.	RF34	Aprobar las soluciones a los problemas transcurrido el tiempo establecido.
RF6	Asignar roles a los usuarios.	RF35	Crear un nuevo problema.
RF7	Visualizar trazas.	RF36	Modificar un problema existente.
RF8	Crear grupos de usuarios.	RF37	Eliminar un problema existente.
RF9	Eliminar grupos de usuarios.	RF38	Listar los problemas existentes.
RF10	Asignar un usuario a un grupo.	RF39	Crear los estados necesarios para calificar a cada problema.
RF11	Eliminar a un usuario de un grupo.	RF40	Crear los niveles de prioridad para calificar a cada problema.
RF12	Listar los grupos de usuarios.	RF41	Crear los niveles de urgencia para calificar a cada problema.
RF13	Listar los usuarios de un grupo.	RF42	Crear los niveles de impacto para calificar a cada problema.
RF14	Crear una categoría.	RF43	Crear una nueva área.
RF15	Modificar una categoría existente.	RF44	Modificar un área existente.
RF16	Eliminar una categoría existente.	RF45	Eliminar un área existente.
RF17	Listar las categorías existentes.	RF46	Listar las áreas.
RF18	Asignar una categoría como subcategoría de otra categoría.	RF47	Asignar a un problema un área de ocurrencia.
RF19	Eliminar una subcategoría de una categoría.	RF48	Adjuntar archivo(s) a cada problema.
RF20	Listar las subcategorías de una categoría.	RF49	Realizar notificaciones por correo.
RF21	Crear una nueva solución.	RF50	Insertar imágenes en la descripción de cada problema.
RF22	Modificar una solución existente.	RF51	Permitir escalamiento entre los problemas.
RF23	Eliminar una solución.	RF52	Generar informes por categorías.

RF24	Listar las soluciones existentes.	RF53	Generar informes por estado.
RF25	Asignar una solución a varios problemas.	RF54	Generar informes por prioridad.
RF26	Aprobar acciones para una solución.	RF55	Generar informes por impacto.
RF27	Remitir solución para aprobación.	RF56	Generar informes por usuario.
RF28	Aprobar solución.	RF57	Generar informes por urgencia.
RF29	Rechazar solución.	RF58	Listar los errores conocidos.

2.4.2. Métricas para la validación de requisitos

Estabilidad de los requisitos: El objetivo de esta métrica es medir la estabilidad de los requisitos y asegurar su adecuación antes de pasar al próximo flujo de trabajo. Se considera que los requisitos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación. (Pressman, 2007)

La estabilidad de los requisitos se calcula como: ETR = [(RT – RM) / RT] * 100.

ETR: valor de la estabilidad de los requisitos.

RT: total de requisitos definidos.

RM: número de requisitos modificados, que se obtienen como la sumatoria de los requisitos insertados, modificados y eliminados.

Esta métrica ofrece valores entre 0 y 100. El mejor valor de ETR es el más cercano a 100 porque mostrará que no se están realizando cambios sobre los requisitos, son estables y, por tanto, es confiable trabajar el análisis y diseño sobre ellos (Pressman, 2007)

Primera iteración: ERT= [(58-12)/58]*100 = 79

Segundo iteración: ERT= [(58-4)/58]*100 = 93

Luego de realizada la primera iteración se determinó que los requisitos tenían un 79% de estabilidad y por tanto no contaban con la estabilidad necesaria. Esta situación cambió para la segunda y última iteración obteniendo un 93% de estabilidad en los requisitos funcionales del sistema.

2.4.3. Requisitos no funcionales del software

“Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los

requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema.” (Sommerville, 2005)

1. Usabilidad

RNU 1 El usuario administrador del módulo debe tener al menos conocimientos básicos de informática.

RNU 2 El módulo debe presentar un acceso fácil y rápido, para facilitar su uso por usuarios con pocos conocimientos en el campo de la informática.

2. Diseño e implementación

RNDI 1 Se utilizará PostgreSQL en su versión 9.1 o superior como Sistema Gestor de Bases de Datos.

RNDI 2 Se utilizará Groovy como lenguaje de programación y Grails como marco de trabajo.

RNDI 3 Se utilizará como arquitectura del sistema Modelo-Vista-Controlador.

RNDI 4 Se utilizará Intellij IDEA como IDE para el desarrollo del sistema.

RNDI 5 Se utilizará Visual Paradigm como herramienta de modelado.

RNDI 6 Para el diseño de las páginas se utilizarán las siguientes tecnologías: CSS, XHTML, JavaScript y JQuery.

3. Funcionamiento

Software:

RNFO 1 Para el funcionamiento en la PC cliente será necesario Firefox 28 o superior.

Hardware:

RNFO 2 El servidor para aplicaciones web deberá tener las siguientes características mínimas:

- Procesador Intel Pentium Dual Core a 2.0 Ghz, equivalente o superior.
- Tarjeta de red o capacidad de conectividad.
- 2 GB de memoria RAM.
- Capacidad de 100 GB de disco duro.

RNFO 3 El servidor para la base de datos deberá tener las siguientes características mínimas:

- Procesador Intel Pentium Dual Core a 2.0 Ghz, equivalente o superior.
- Tarjeta de red o capacidad de conectividad.

- 1 GB de memoria RAM.
- Capacidad de 250 GB de disco duro.

4. Seguridad

Confidencialidad:

RNS 1 El acceso al módulo así como la información se encontrarán protegidos contra accesos no autorizados utilizando mecanismos de autenticación propios del sistema.

RNS 2 La autenticación del módulo será la primera acción del usuario, el cual deberá proporcionar un usuario único y una contraseña, los cuales serán de uso exclusivo del usuario.

RNS 3 Las diferentes áreas del módulo se encontrarán protegidas contra acceso no autorizado utilizando roles.

Integridad:

RNS 4 La información podrá ser modificada solo por el personal autorizado.

RNS 5 Se harán validaciones de la información en el servidor contra ataques de inyección HTML y SQL.

5. Fiabilidad

RNF 1 Ante una inserción de datos, vista de detalles, modificación o eliminación el módulo debe responder en no más de 3 segundos.

RNF 2 El módulo debe responder en un promedio de 6 segundos la petición de un reporte.

RNF 3 El módulo deberá soportar transacciones de cerca de 300 usuarios conectados simultáneamente.

6. Interfaz de usuario

RNIU 1 Las interfaces del módulo contendrán los datos de forma estructurada, permitiendo la correcta interpretación de la información.

RNIU 2 La entrada incorrecta de datos será mostrada al usuario claramente, detallando los campos donde se encuentra el error y mostrando como título el detalle del error.

RNIU 3 El diseño de la interfaz del sistema responderá a la ejecución de acciones de forma rápida, minimizando los pasos a dar en cada proceso.

7. Interconexión

RNI 1 La PC cliente se comunicará mediante el Protocolo de transferencia de hipertexto (HTTP[30]) y el Protocolo de transferencia de hipertexto seguro (HTTPS[31]) con el servidor de aplicaciones, este se comunicará mediante el Protocolo de control de transmisión/Protocolo de Internet (TCP\IP) con el servidor de bases de datos.

RNI 2 Los datos de usuarios y demás datos necesarios para el módulo podrán ser extraídos de los servicios web brindados por el directorio de servicios web de la UCI.

2.5. Fase de planificación

"En esta fase el cliente establece la prioridad de cada historia de usuario, y teniendo en cuenta esto los programadores realizan una estimación de esfuerzo necesario para cada una de ellas. Se realizan acuerdos sobre el material a entregar en la primera iteración y en correspondencia se genera un cronograma junto al cliente." (Padanés, y otros, 2007)

2.5.1. Estimación de esfuerzo

La medida utilizada para la estimación del esfuerzo asociado a la implementación es el punto. Un punto equivale a una semana ideal de programación. Esta medida generalmente toma valores de 1 a 3.

Tabla 4: Plan de esfuerzo por HU

No.	Historia de usuario	Punto de estimación
1.	Autenticar a los usuarios utilizando el directorio LDAP de la Universidad.	1
2.	Autenticar a los usuarios utilizando el directorio local.	1
3.	Gestionar usuarios del sistema.	1
4.	Asignar roles a los usuarios.	1
5.	Crear un sistema de trazas.	2
6.	Crear grupos de usuarios.	1
7.	Eliminar grupos de usuarios.	1
8.	Asignar un usuario a un grupo.	1
9.	Eliminar a un usuario de un grupo.	1
10.	Listar los grupos de usuarios.	1
11.	Listar los usuarios de un grupo.	1

[30] Del inglés: *Hypertext Transfer Protocol.*
[31] Del inglés: *Hypertext Transfer Protocol Secure*.

12.	Realizar búsquedas por parámetros de los usuarios.	2
13.	Gestionar categoría.	2
14.	Asignar una categoría como subcategoría de otra categoría.	1
15.	Eliminar una subcategoría de una categoría.	1
16.	Listar las subcategorías de una categoría.	1
17.	Gestionar solución.	2
18.	Asignar una solución a varios problemas.	1
19.	Permitir asociar imágenes a las soluciones.	1
20.	Aprobar acciones para una solución.	2
21.	Remitir solución para aprobación.	1
22.	Aprobar solución.	1
23.	Rechazar solución.	1
24.	Permitir generar reportes en formato pdf.	2
25.	Generar los informes de soluciones por categorías.	1
26.	Determinar soluciones por el usuario que las creó.	1
27.	Determinar cantidad de soluciones por usuario.	1
28.	Gestionar problemas.	1
29.	Crear los estados necesarios para calificar a cada problema.	1
30.	Crear los niveles de prioridad para calificar a cada problema.	1
31.	Crear los niveles de urgencia para calificar a cada problema.	1
32.	Crear los niveles de impacto para calificar a cada problema.	1
33.	Gestionar área.	1
34.	Asignar a un problema un área de ocurrencia.	1
35.	Adjuntar archivo(s) a cada problema.	1
36.	Realizar notificaciones por correo.	1
37.	Insertar imágenes en la descripción de cada problema.	2
38.	Permitir escalamiento entre los problemas.	1
39.	Generar informes por categorías.	1
40.	Generar informes por estado.	1
41.	Generar informes por prioridad.	1
42.	Generar informes por impacto.	1
43.	Generar informes por usuario.	1

44.	Generar informes por urgencia.	1
45.	Listar los errores conocidos.	1

2.5.2. Plan de iteraciones

Para cada entrega fueron escogidas algunas HU, teniendo en cuenta el orden definido. Este plan define las historias de usuario que deben ser implementadas en cada iteración y las fechas de liberación. A continuación se muestran 3 iteraciones y el número de historias por cada una.

Tabla 5: Plan de iteraciones

Iteración	Historias de usuario	Duración total (semanas)
1	Autenticar a los usuarios utilizando el directorio LDAP de la Universidad	8.5
	Autenticar a los usuarios utilizando el directorio local.	
	Gestionar usuarios del sistema.	
	Gestionar problemas.	
	Asignar roles a los usuarios.	
	Crear un sistema de trazas.	
	Crear grupos de usuarios.	
	Eliminar grupos de usuarios.	
	Asignar un usuario a un grupo.	
	Eliminar a un usuario de un grupo.	
	Listar los grupos de usuarios.	
	Listar los usuarios de un grupo.	
	Gestionar categoría.	
	Asignar una categoría como subcategoría de otra categoría.	
	Eliminar una subcategoría de una categoría.	
	Listar las subcategorías de una categoría.	
	Gestionar solución.	
2	Asignar una solución a varios problemas.	1.8
	Permitir asociar imágenes a las soluciones.	
	Aprobar acciones para una solución.	
	Remitir solución para aprobación.	

	Aprobar solución.	
	Rechazar solución.	
	Gestionar área.	
	Asignar a un problema un área de ocurrencia.	
	Adjuntar archivo(s) a cada problema.	
	Realizar notificaciones por correo.	
	Crear los estados necesarios para calificar a cada problema.	
	Crear los niveles de prioridad para calificar a cada problema.	
	Crear los niveles de urgencia para calificar a cada problema.	
	Crear los niveles de impacto para calificar a cada problema.	
	Listar los errores conocidos existentes.	
3	Permitir generar reportes en formato pdf.	4.3
	Generar los informes de soluciones por categorías.	
	Determinar soluciones por el usuario que las creó.	
	Determinar cantidad de soluciones por usuario.	
	Insertar imágenes en la descripción de cada problema.	
	Permitir escalamiento entre los problemas.	
	Generar informes por categorías.	
	Generar informes por estado.	
	Generar informes por prioridad.	
	Generar informes por urgencia.	
	Generar informes por impacto.	
	Generar informes por usuario.	
	Aprobar las soluciones a los problemas transcurrido el tiempo establecido.	

2.5.3. Plan de entrega

En el momento en que culmina la elaboración de las HU, se inicia el proceso de creación de un plan de entrega. El cual tiene como objetivo fundamental la obtención por parte de los programadores de una estimación detallada del período de tiempo que deben tener en cuenta para la implementación.

Tabla 6: Plan de entrega de las iteraciones

Artefacto		Iteración	Entrega
HU1	HU13	1	13 de marzo
HU2	HU14		
HU3	HU15		
HU4	HU16		
HU5	HU17		
HU6	HU12		
HU7	HU29		
HU8	HU30		
HU9	HU31		
HU10	HU32		
HU11	HU33		
HU18	HU23	2	26 de marzo
HU19	HU34		
HU20	HU35		
HU21	HU36		
HU22			
HU24	HU40	3	25 de abril
HU25	HU41		
HU26	HU42		
HU27	HU43		
HU28	HU44		
HU37	HU45		
HU38	HU39		

2.6. Fase de Diseño

“La arquitectura del software de un programa o sistema de cómputo es la estructura o las estructuras del sistema, que incluyen los componentes del software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos” (Pressman, 2007).

2.6.1. Tarjetas CRC

Las tarjetas Clase-Responsabilidad-Colaboración (CRC) permiten ver las clases no como un depósito de datos, sino que permiten conocer el comportamiento de cada una en un alto nivel. La metodología XP estipula su uso como un artefacto obligatorio durante el desarrollo de un proyecto, debido a los beneficios que aportan a los desarrolladores. A continuación se muestran las tarjetas de una de las clases que responden a funcionalidades de alta prioridad en el sistema. El resto de las tarjetas pueden visualizarse en el documento **"Tarjetas CRC GESERV.doc"** adjunto en el expediente de proyecto *Entregables*.

Tabla 7: Tarjeta CRC de la clase CategoriaController

Nombre de la clase: CategoriaProblemaController	
Responsabilidades	**Clases Relacionadas**
Esta clase es la encargada de asignar una categoría como subcategoría de otra categoría: • asignar() • CategoriaProblemaInstance() **Además de las funcionalidades que genera el marco de trabajo de forma automática:** • index() • list(Integer max) • create() • save() • show(Long id) • edit(Long id) • update(Long id, Long version) • delete(Long id)	ProblemaController SoluciónController ErrorController

2.6.2. Patrones de diseño

En la implementación del sistema se utilizaron varios patrones de diseño, a continuación se mencionan y se explica el modo en que fueron utilizados.

Patrones generales de software para asignación de responsabilidades (GRASP)[32]

Experto: Dentro de un diseño que utilice muchas clases es imprescindible la correcta asignación de responsabilidades. Estas responsabilidades deben ser dadas al experto en información, es decir a la clase que cuenta con la información necesaria para cumplir el trabajo. Si las asignaciones se hacen en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar (Larman, 2007). Un ejemplo de este patrón se encuentra en la clase "Problema" que es la encargada de determinar la cantidad de problemas por categorías, por estado, por prioridad y por impacto. Esta clase es la única que cuenta con toda la información para realizar las búsquedas y determinar los porcientos que representan dentro del total de problemas.

Creador: La creación de objetos es una actividad muy frecuente en los sistemas orientados a objetos, por lo tanto es conveniente asignar esta responsabilidad de manera que potencie un bajo acoplamiento, una mayor claridad y una alta reutilización. La nueva instancia debe ser creada por la clase que tenga la información mínima necesaria para hacerlo. Este patrón se ve evidenciado en la clase "LugarController", esta clase tiene la responsabilidad de crear instancias de la clase "Lugar". El controlador está relacionado con la clase del dominio, recibiendo de la vista los datos necesarios para crear la instancia del objeto. "LugarController" es el creador de objetos de "Lugar" como se evidencia en la Ilustración 7.

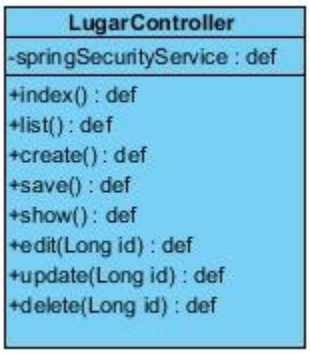


Ilustración 7: LugarController

Bajo acoplamiento: el principio de funcionamiento de Grails se manifiesta a través de la implementación de un método de trabajo particular que establece que a cada clase del dominio que necesite vistas se le genera un controlador. El controlador maneja una variable nombrada *params* creada específicamente para almacenar valores. Esta variable es procesada por las

[32]Del inglés: *General Responsibility Assignment Software Patterns.*

clases del dominio garantizando que se puedan modificar arbitrariamente los controladores o las clases del dominio mientras que *params* permanecerá intacto.

Controlador	Clase del dominio

`def create() {`
`  [lugarInstance: new Lugar(params)]`
`}`
	`Lugar(Map params) {`
	`  nombre=params.nombre`
	`  descripcion=params.descripcion`
	`  direccion=params.direccion`
	`  contacto=params.contacto`
	`  pais=params.pais`
	`}`

Alta cohesión: Grails permite asignar responsabilidades con una alta cohesión ya que los controladores definen las acciones para las plantillas y colaboran con otras clases para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades. Las clases controladoras contienen diferentes funcionalidades que se encuentran estrechamente relacionadas posibilitando que el software sea flexible frente a grandes cambios.

Controlador: Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Un evento del sistema es un evento de alto nivel generado por un actor externo; un evento de entrada externa (Larman, 2007). El controlador es el encargado de recibir el evento, interpretarlo y ejecutar una determinada operación de respuesta. La utilización del marco de trabajo Grails permite reconocer fácilmente este patrón ya que todos los controladores terminan con la palabra Controller. En la Ilustración 8 se refleja el uso de este patrón.

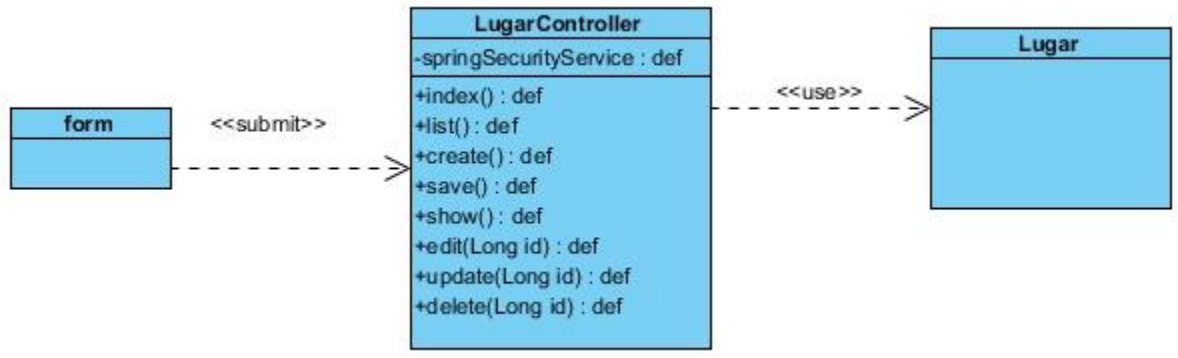


Ilustración 8: Patrón GRASP Controlador

Patrones GoF[33]

Singleton: Este patrón garantiza que solamente se cree una instancia de la clase y provee un punto de acceso global a él. Todos los objetos que utilizan una instancia de esa clase usan la misma instancia. La utilización de servicios dentro del marco de trabajo define la creación de un objeto "Service" que será utilizado por todas las clases. Este es el caso del servicio de seguridad springSecurityService, el cual es utilizado por todos los controladores para manejar el acceso a los distintos métodos. La Ilustración 9 muestra la definición del servicio como una llamada a la instancia ya creada.

```
class ProblemaController {
    def springSecurityService

    static allowedMethods = [save: "POST", update: "POST", delete: "POST", guardarArchivoAdjunto: "POST"]

    @Secured(['ROLE_TECNICO', 'ROLE_ESPECIALISTA', 'ROLE_ADMIN'])
    def index() {redirect(action: "list", params: params)}

    @Secured(['ROLE_TECNICO', 'ROLE_ESPECIALISTA', 'ROLE_ADMIN'])
    def list(Integer max) {...}
```

Ilustración 9: ProblemaController

2.6.3. Estilo arquitectónico

Grails sigue un estilo generalmente empleado en el desarrollo de aplicaciones web, denominado Modelo-Vista-Controlador. Este estilo establece que los componentes de un sistema de software deben organizarse en tres capas distintas según su misión (Brito Calahorro, 2009):

- **Modelo:** contiene los componentes que representan y gestionan los datos manejados por la aplicación. En el caso más típico, los objetos encargados de leer y escribir en la base de datos.
- **Vista:** los componentes de esta capa son responsables de mostrarle al usuario el estado actual del modelo de datos, y presentarle las distintas acciones disponibles.
- **Controlador:** contendrá los componentes que reciben las órdenes del usuario, gestionan la aplicación de la lógica del negocio sobre el modelo de datos y determina que debe mostrarse en la vista.

El marco de trabajo Grails propone un modo particular definiendo como base del desarrollo las clases del dominio o modelo. A partir de esta clase se crean los controladores y estos a su vez

[33] Gang of Four

a las vistas. Como se puede apreciar en la Ilustración 10 los controladores todos cuentan con un conjunto de vistas y utilizan a las clases del dominio para el manejo de los datos.

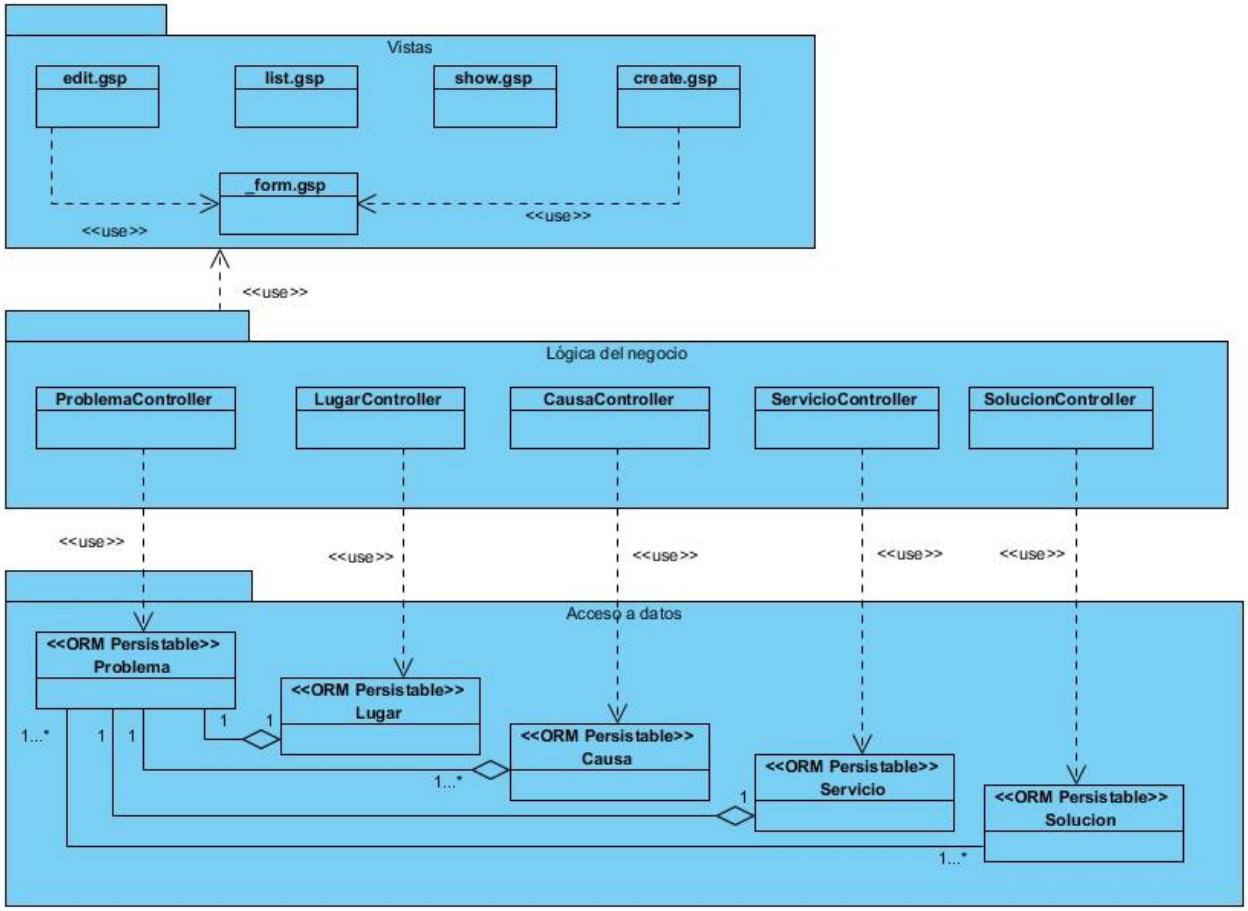


Ilustración 10: Estilo Modelo Vista Controlador

2.6.4. Validación del diseño

"El diseño de software se encuentra en el núcleo técnico de la ingeniería y se aplica de manera independiente al modelo de software que se utilice. Una vez que se analizan y se especifican los requisitos, el diseño de software es la última acción de ingeniería correspondiente dentro de la actividad del modelado, la cual establece una plataforma para la construcción (generación de código y pruebas)" (Pressman, 2007). Para validar el diseño realizado se tendrán en cuenta distintos parámetros como son la responsabilidad de las clases, la reutilización y la complejidad de implementación y mantenimiento. Las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC) permiten evaluar estos parámetros y adicionalmente el bajo acoplamiento y la cantidad de pruebas unitarias que deben realizarse. Para evaluar las métricas son necesarios los valores de los umbrales para los parámetros de calidad. Para consultar los umbrales utilizados ver **Anexo 1**.

TOC: Esta métrica se basa en la cantidad de operaciones que tiene encapsulada una clase y evalúa los siguientes atributos de calidad:

- Responsabilidad: un aumento del TOC implica un incremento de la responsabilidad asignada a la clase.
- Complejidad de implementación: un aumento del TOC implica un incremento de la complejidad de implementación de la clase.
- Reutilización: un aumento del TOC implica una disminución del grado de reutilización de la clase.

Los resultados obtenidos luego de aplicar la métrica se muestran en la Ilustración 11.

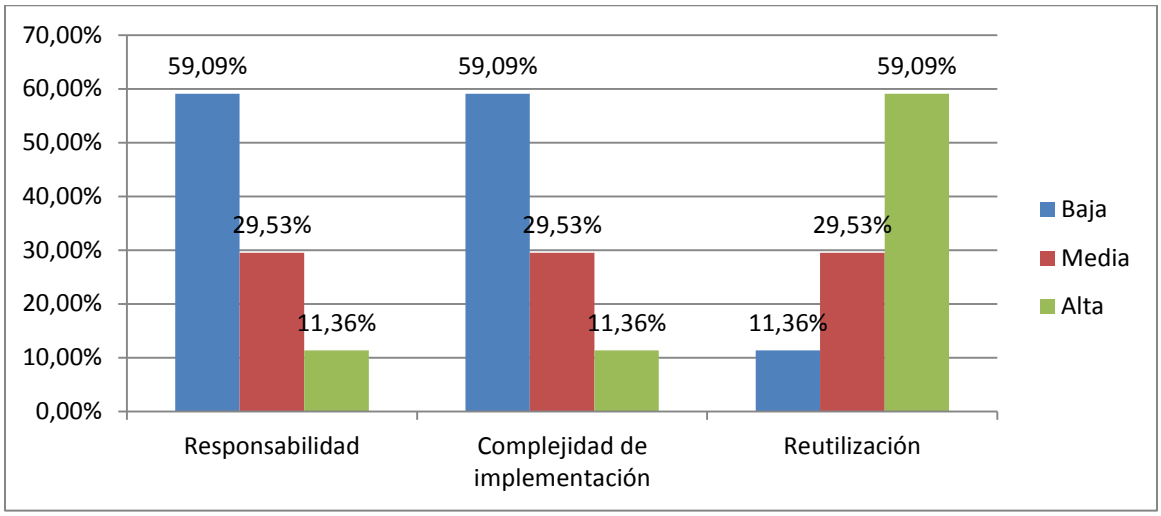


Ilustración 11: Representación de la métrica TOC

RC: Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

- 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.
- Reutilización: un aumento del RC implica una disminución en el grado de reutilización de la clase.
- Cantidad de pruebas: un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias.

La Ilustración 12 muestra los resultados de la aplicación de la métrica RC para los parámetros complejidad de mantenimiento y cantidad de pruebas.

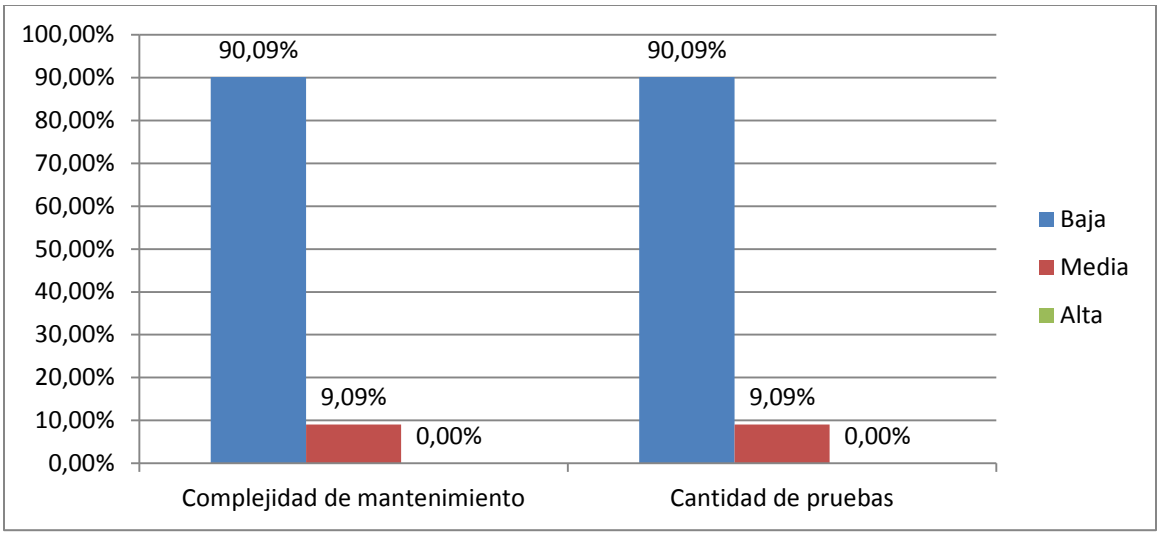


Ilustración 12: Representación de la métrica RC

Por otro lado la Ilustración 13 muestra el resultado de la aplicación de esta métrica para el parámetro acoplamiento.

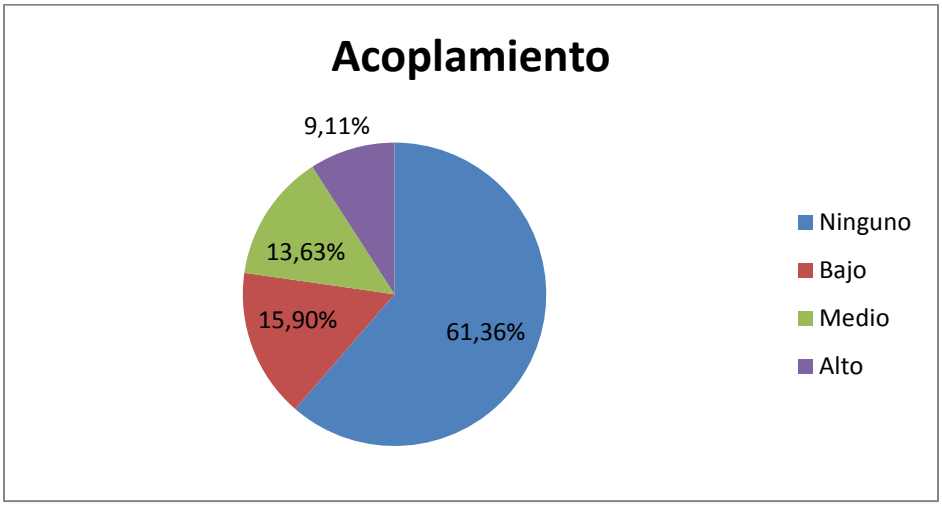


Ilustración 13: Representación del acoplamiento en la métrica RC

Luego de aplicadas las métricas se constató que el diseño propuesto provee un nivel bajo de responsabilidad. Esto acarrea además un porciento de complejidad de implementación pequeño lo cual se traduce en menos esfuerzo. El parámetro de complejidad de mantenimiento viene ligado fundamentalmente a las relaciones que existen entre las clases y a la dificultad de realizar modificaciones, se consideran buenos los resultados atendiendo a que el 90% de las mismas presentan un nivel bajo. Tomando en cuenta que más del 90% de las clases tienen baja necesidad de realizar pruebas unitarias el diseño propuesto reducirá el esfuerzo empleado. En el caso del parámetro de la reutilización se obtuvieron los resultados deseados,

el equipo de desarrollo persigue como objetivo que el código pueda ser reutilizado en otros módulos del Sistema de Gestión de servicios. Por último, se tiene en cuenta el acoplamiento, que atendiendo a los resultados obtenidos el 77.26% de las clases tienen bajo o ningún acoplamiento, por lo que se reafirma que la aplicación del patrón Bajo Acoplamiento es correcto. De esta manera queda validado el diseño de la propuesta de solución atendiendo a los parámetros definidos.

2.7. Conclusiones del capítulo

En este capítulo se abordaron los aspectos fundamentales del análisis y diseño de la propuesta de solución. La descripción de la metáfora del sistema permitió crear una expresión en lenguaje natural que exprese en principio básico el funcionamiento global que se desea para la propuesta de solución, esto se traduce en una mejor comprensión para las personas ajenas al proceso. La realización de las HU permitió determinar cuáles eran las funcionalidades básicas que deseaban los especialistas y técnicos del Centro de Soporte UCI. Estas funcionalidades fueron traducidas a requisitos funcionales y en el proceso se obtuvieron además algunas especificaciones que fueron convertidas en requisitos no funcionales. La estimación del esfuerzo y el plan de iteraciones permitieron crear un cronograma que facilita la implementación teniendo en cuenta tanto los compromisos con el cliente, como la capacidad interna para ajustar las técnicas de Gestión de servicios a la propuesta de solución.

La creación de un plan de entrega permitió ajustar la realización del sistema a menos de 4 meses. La creación de las tarjetas CRC permitió la separación de las funcionalidades implementadas por el marco de trabajo y las que se necesitan implementar en el desarrollo. Por último, la validación del diseño realizado a través de métricas permitió la proyección futura de resultados, basándose en la premisa de que un buen diseño es la base para generar una buena implementación.

Capítulo 3: Implementación y pruebas de la solución

3.1. Introducción

En este capítulo se abordarán los aspectos fundamentales del proceso de desarrollo y pruebas. Se expondrán los detalles de la ejecución de la fase de desarrollo propuesta por la metodología. En el transcurso de todo este proceso se definirán los estándares de codificación, la seguridad del sistema y el modelo de datos. Además se expondrán las condiciones mínimas para el funcionamiento del módulo. En la fase de pruebas se ejecutarán pruebas unitarias, de sistema y de aceptación como forma de validación funcional de la propuesta de solución.

3.2. Fase de desarrollo

En esta fase de la metodología se codifican las especificaciones recogidas por las HU de manera tal que se cumplan los requisitos de todo el sistema. XP plantea que el mejor artefacto que se puede generar es el código, su buena calidad influirá en gran medida en el producto final. La calidad del código es influenciada por los estándares de codificación los cuales serán tenidos en cuenta en esta fase. Se expondrá el método de manejo de la seguridad del sistema para establecer una forma de proteger la información y los distintos módulos de la aplicación.

3.2.1. Estándares de codificación

Los estándares de codificación garantizan la comunicación fluida y directa entre los programadores, permitiendo el aumento de la reutilización y el mantenimiento de los sistemas. El equipo de desarrollo definió antes de iniciar la implementación que utilizará el estándar CamelCase[34]. En el caso de la definición de variables se decide utilizar el lowerCamelCase. Como se utiliza el marco de trabajo Grails es necesario utilizar el estándar predefinido. Este estándar se describe a continuación:

Vistas: todas las vistas son definidas utilizado lowerCamelCase, excepto el caso de las plantillas a las cuales se les añadirá _ antes del nombre por ejemplo _miPlantilla.

Controladores: todos los controladores son definidos utilizando UpperCamelCase y terminan con la palabra Controller por ejemplo *ProblemaSolucionadoController*.

[34] CamelCase es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en CamelCase se asemejan a las jorobas de un camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula. Si comienza con mayúscula, se denomina UpperCamelCase y, si no, lowerCamelCase

Servicios: para definir los servicios se utiliza UpperCamelCase y se termina con la palabra Service por ejemplo *UsuarioService*.

Clases de dominio: la definición de las clases del dominio la realiza el marco de trabajo de forma automática utilizando UpperCamelCase.

Trabajos (Jobs): para definir los trabajos o tareas programadas se utiliza UpperCamelCase y los nombres termina en la palabra Job por ejemplo *SolucionesVencidasJob.*

3.2.2. Seguridad del sistema

La seguridad del sistema es garantizada utilizando el marco de trabajo Spring Security, para su uso en aplicaciones Grails es necesario la instalación de un complemento del mismo nombre. Cuando dentro de una organización, los roles se crean para diferentes funciones de trabajo y los permisos para realizar ciertas operaciones se asignan a roles específicos, es necesario establecer la seguridad basándose en estos roles. Atendiendo a los distintos roles que pueden tener los usuarios del sistema: *técnico, especialista, cliente, calidad, proveedor y administrador*, y que a esos roles se les ha asignado distintos permisos se utilizará el enfoque Control de Acceso Basado en Roles (RBAC[35]). Este enfoque permite centralizar el trabajo en cada uno de los roles y controlar el flujo de actividades a través de sus responsabilidades. La utilización de Spring Security puede variarse atendiendo a necesidades de los desarrolladores, en el caso de la propuesta de solución se utilizará haciendo uso del enfoque RBAC.

3.2.3. Diagrama de despliegue

El diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. El primer nodo representado en la Ilustración 14 es la PC Cliente en la cual los usuarios para acceder a la propuesta de solución necesitan tener instalado Mozilla Firefox en su versión 28. El segundo nodo es el Servidor Web, el cual cuenta con un componente que es la máquina virtual de Java en su versión 1.7 actualización 50 la cual servirá de base para la ejecución del componente denominado Apache Tomcat. El Apache Tomcat 2.1.1 manejará las peticiones que se realicen a la propuesta de solución. Igualmente se refleja el PostgreSQL en su versión 9.1-1.1 como servidor de base de datos en el tercer nodo. El cuarto y último nodo representa el servidor de directorio activo de la UCI el cual será utilizado para la autenticación al sistema con las credenciales de los usuarios.

[35] Del inglés: *Role-Based Access Control.*

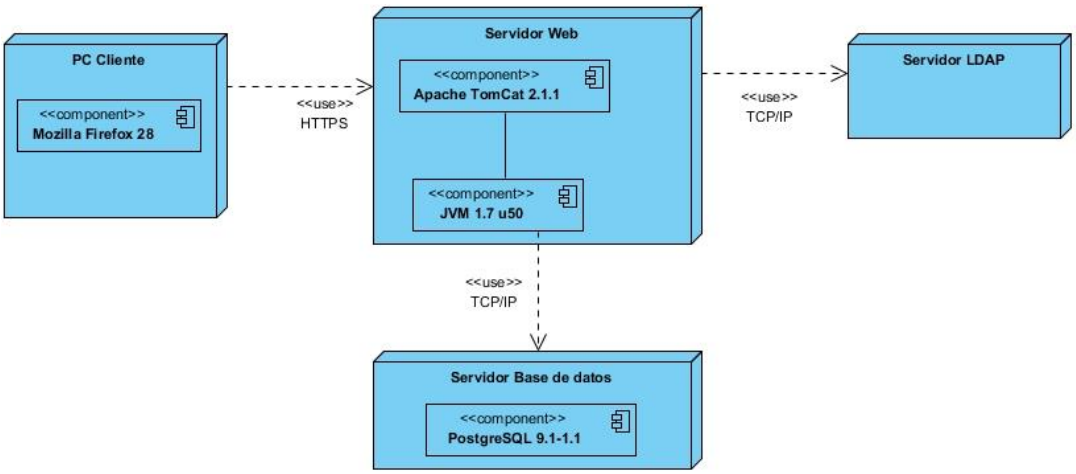


Ilustración 14: Diagrama de despliegue

3.2.4. Modelo de Datos

Se obtuvo el Modelo de datos de la propuesta de solución luego de aplicar la técnica de modelado Entidad-Relación-Atributo (ERA), el diagrama generado puede ser consultado en el expediente de proyecto *Entregables* con el nombre **MERA GESERV.jpg**. La Ilustración 15 representa en su parte superior las tablas *public.problema, public.causa* y *public.solución,* estas serán las encargadas de almacenar los datos referentes a los problemas que arriban al sistema, el conjunto de soluciones existentes en el sistema y las causas asignadas. Por último se representa la asignación de una solución a un problema creando un error conocido representado por la tabla public.conocimiento.

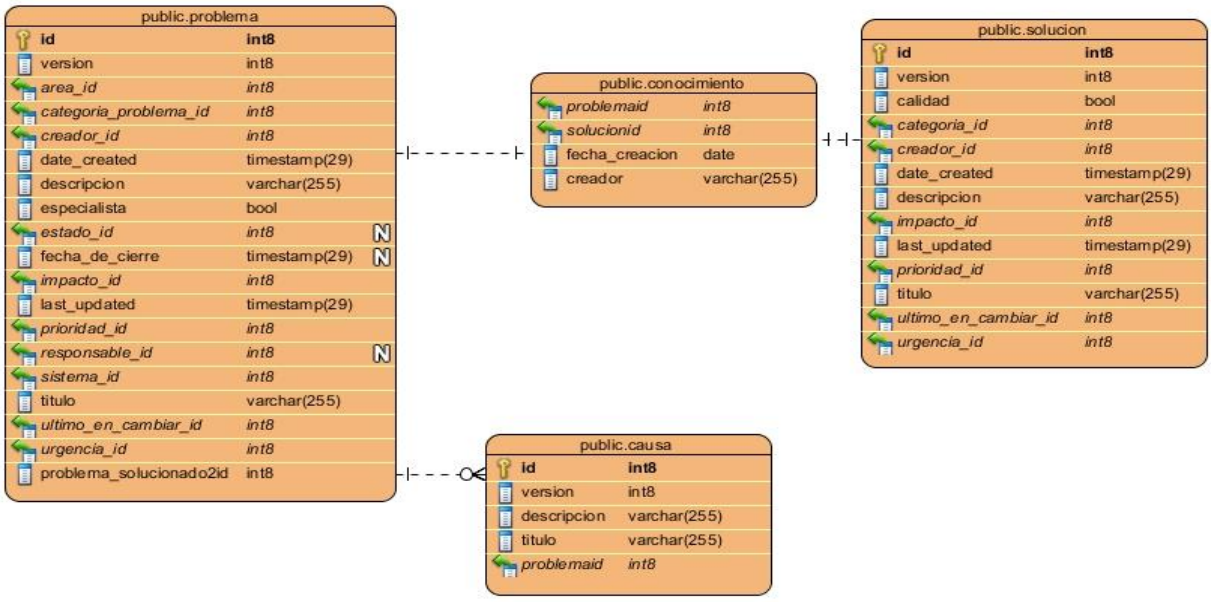


Ilustración 15: Fragmento del modelo de datos

3.3. Fase de pruebas

Se escogen algunas de las pruebas planteadas por (Pressman, 2007) para aplicarlas al módulo implementado. Este autor propone como pauta a seguir que el proceso debe realizarse desde la parte más pequeña del software hasta la más grande, formando una espiral entre los niveles de prueba. El objetivo fundamental de las pruebas es determinar si el código generado funciona correctamente y realiza las operaciones deseadas. Además permiten comprobar que el producto cumple con los requisitos acordados con el cliente. Por último, verifican que el sistema sea capaz de funcionar normalmente en un ambiente parecido o en el ambiente de despliegue.

Para cumplir con lo antes mencionado se escaló desde las pruebas unitarias (encargadas de probar el código), pruebas de sistema (dirigidas a probar la ingeniería del sistema) hasta las pruebas de aceptación (encargadas de probar los requisitos).

3.3.1. Pruebas unitarias

Las pruebas unitarias se centran en un esfuerzo de verificación de la unidad más pequeña del diseño de software: el componente y el módulo del software. “Las pruebas de unidad se centran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente.” (Pressman, 2007). Al examen del detalle procedimental, dígase de las estructuras lógicas, la colaboración de los componentes; que busca probar que se ejecuten todas las condiciones y la integridad de los bucles se le denomina *prueba de caja blanca*.

Prueba de caja blanca

“La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, se pueden obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada funcionalidad.” (Pressman, 2007). Dentro de las técnicas de prueba de caja blanca se encuentra la *prueba de la ruta básica*. Esta técnica le permite a los desarrolladores de casos de pruebas obtener una medida de la complejidad lógica de un diseño procedimental y provee una medida para definir un conjunto básico de rutas de ejecución.

Para realizar esta técnica se debe construir una estructura que represente en notación simple el flujo del control, a esta representación se le llama gráfica de flujo. Una de las dificultades fundamentales de esta técnica es la determinación del número de rutas, para esto se utiliza la métrica de software complejidad ciclomática. Esta métrica consiste en determinar una medida cualitativa de la complejidad lógica del programa. “Cuando se emplea en el contexto del

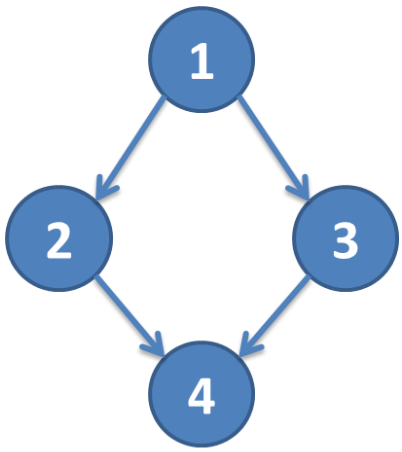
método de prueba de la ruta básica, el valor calculado mediante la complejidad ciclomática define el número de rutas independientes en el conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan aplicado por lo menos unas vez.” (Pressman, 2007).

```
def edit(Long id) {
    def paisInstance = Pais.get(id)
    if (!paisInstance) {
        flash.message = message(code: 'default.not.found.message', args: [message(code: 'pais.label', default: 'Pais'), id])
        redirect(action: "list")
        return
    }

    [paisInstance: paisInstance]
}
```

Ilustración 16: Código del método *edit* del controlador *PaisController*

Para el cálculo de la complejidad ciclomática se tiene en cuenta que esta corresponde al número de regiones. Estas regiones son calculadas utilizando la fórmula V(G) = (E-N)+2, donde E representa el número de aristas y N el número de nodos. Para la ejecución de las pruebas se utilizó la herramienta JUnit[36] embebida con el marco de trabajo. Esta herramienta permite declarar los métodos *populateInvalidParams* y *populateValidParams* los cuales proveen valores a todas las acciones y comprueba el resultado mediante la respuesta obtenida.

Prueba estructural de Caja Blanca	**Probador:** Yorgüy Antonio Batista Desdín
El número de regiones corresponde a la complejidad ciclomática. E=4 N=4 V (G) = (E – N) + 2=2 Rutas linealmente independientes: 1) 1-2-4 2) 1-3-4	Representación de la gráfica de flujo. 

[36] JUnit es un marco simple para escribir pruebas repetibles. Es un ejemplo de la arquitectura xUnit para marcos de pruebas unitarias.

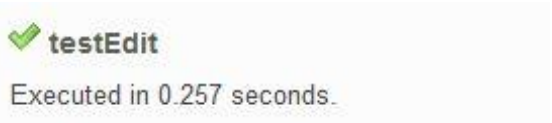
Caso de prueba para el camino básico 1	
Descripción: El dato de entrada es el id del país que se desea modificar	
Condición de ejecución: El usuario debe poseer el rol de administrador.	
Procesamiento de la prueba	
Dato de entrada:	El id es incorrecto o nulo
Tipo de dato esperado:	Mensaje de error

Método automático	def populateInvalidParams(params){
	params["id"]=null
	params["id"]=-10
	}

Resultado de la prueba:  	
Evaluación del caso de prueba:	Satisfactoria

Caso de prueba para el camino básico 2	
Descripción: El dato de entrada es el id del país que se desea modificar	
Condición de ejecución: El usuario debe poseer el rol de administrador.	
Procesamiento de la prueba	
Dato de entrada:	El id es correcto.
Tipo de dato esperado:	Mensaje de operación satisfactoria

Método automático	def populateValidParams(params){
	params["id"]=10
	}

Resultado de la prueba:  	
Evaluación del caso de prueba:	Satisfactoria

En el expediente de proyecto *Entregables* en el documento **Diseño de casos de pruebas de Caja Blanca.docx** se muestran el resto de los casos de pruebas de caja blanca aplicados a la solución. Teniendo en cuenta que el enfoque establecido para la realización de las pruebas es

incremental es decir de lo más pequeño a lo más general se consideró oportuno realizar las pruebas unitarias a cada clase del dominio o controlador generado, de esta manera se garantiza la calidad de estas partes. En la Tabla 8 se puede observar el número de iteraciones y las no conformidades encontradas.

Tabla 8: Iteraciones de pruebas

Número de Iteraciones	Número de no conformidades	Asociadas a
1ra	30	Errores de interfaz, de validación y ortografía.
2da	10	Errores de interfaz, de validación y ortografía.
3ra	0	

3.3.2. Pruebas de sistema

Las pruebas del sistema ejercitan profundamente el sistema con el objetivo de comprobar que se hayan integrado adecuadamente todos los elementos del software y del hardware, y que se realizan las funciones adecuadas. Las pruebas ya aplicadas se centran en el código, de ahí surge la necesidad de comprobar el rendimiento y la respuesta del sistema ante peticiones simultáneas de distintos usuarios en varios escenarios.

Pruebas de resistencia (Stress)

Las pruebas de resistencia están diseñadas para enfrentar a los programas a situaciones anormales de peticiones. Con este tipo de pruebas al sistema, se comprueba la respuesta ante demandas de recursos en cantidad, frecuencia o volúmenes anormales buscando la falla del programa.

Pruebas de rendimiento (Carga)

Esta prueba está diseñada para probar el rendimiento del software en tiempo de ejecución. Es muy común que se realicen las pruebas de rendimiento junto a las pruebas de resistencia y, frecuentemente, requieren instrumentación tanto de software como de hardware.

Resultados de las pruebas de rendimiento y resistencia

Para llevar a cabo las pruebas de carga y stress se utilizó la herramienta JMeter 2.11. Esta herramienta es desarrollada en la plataforma Java dentro del proyecto Jakarta. JMeter permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web. También permite la ejecución de pruebas entre distintos ordenadores para realizar pruebas de

rendimiento. A continuación se muestran en la Tabla 9 los resultados obtenidos con la herramienta que fueron extraídos de las imágenes relacionadas en el expediente de proyecto *Entregables* en el documento **Resultados pruebas JMeter.docx**.

Tabla 9: Resultados de las pruebas de carga y stress

Aplicado a	Usuarios (Muestras)	Tiempo de ejecución (ms)				Rendimiento		
		Mín.	Max.	Media	Mediana	% Error	Pet/seg	Kb/seg
Crear Problema	10	299	1288	740	661	0.0%	4.5/sec	60
	150	1162	6406	8165	7608	0.0%	4.2/sec	56.1
	300	388	38285	21776	23164	0.0%	3.4/sec	45.7
Crear Categoría	10	1702	3968	2563	2173	0.0%	1.8/sec	23.7
	150	6431	35806	31580	33837	0.0%	1.8/sec	23.7
	300	564	42949	22425	23303	0.0%	4.0/seg	52.6
Crear Solución	10	1829	2728	2300	2305	0.0%	1.4/sec	18.8
	150	1768	39344	28616	26515	0.0%	1.0/sec	13.3
	300	347	45996	17539	18008	0.0%	2.6/sec	33.2
Gestionar Servicio	10	305	539	435	440	0.0%	2.8/sec	37.4
	150	262	19885	8005	7809	0.0%	2.1/sec	28.1
	300	614	36653	15774	16095	0.0%	2.1/sec	28.0
Asignar Solución a un Problema	10	275	596	423	410	0.0%	2.6/sec	34.7
	150	2456	32758	28529	25919	0.0%	1.6/sec	20.9
	300	996	35789	21341	22410	0.0%	2.2/sec	20.8

Indicadores

- **Columna Mínimo (Mín.):** tiempo mínimo de ejecución invertido para una petición con *n* usuarios realizando peticiones de manera simultánea.
- **Columna Máximo (Máx.):** tiempo máximo de ejecución invertido para una petición con *n* usuarios realizando peticiones de manera simultánea.
- **Media:** tiempo de ejecución promedio de una petición con *n* usuarios.

- **Mediana:** el 50% de las peticiones realizadas por *n* usuarios tardaron menos del valor reflejado.
- **Error:** relación entre el total de peticiones y el total de peticiones que originaron errores.
- **Rendimiento (Pet./seg):** número de peticiones que el servidor puede procesar en un segundo.
- **Rendimiento (Kb./seg):** cantidad de datos que el servidor puede procesar en un segundo.

Análisis de los resultados de las pruebas al sistema

Realizando un análisis de los resultados de la Tabla 9 arrojados al realizar las pruebas de rendimiento y resistencia, se puede constatar que frente al caso crítico de concurrencia de 300 usuarios el sistema presenta tiempos máximos de respuesta que rondan entre 30 y 40 segundos, y tiempos mínimos menores de 1 segundo. Para todas las muestras la ocurrencia de errores se mantuvo en 0, lo que significa que se ejecutaron satisfactoriamente todas las peticiones. Además el promedio de procesamiento de peticiones del servidor por segundo es de 4 con un volumen de procesamiento entre 13Kb y 60Kb de datos, lo que representa una adecuada capacidad de procesamiento. Si se tiene en cuenta que en la actualidad el Centro de Soporte UCI cuenta con 20 trabajadores y 15 servicios, la posibilidad de que se conecte al sistema 1 usuario por cada servicio serían 35 usuarios conectados simultáneamente. El sistema es capaz de soportar 8 veces la carga a la que puede enfrentarse en el peor de los casos $\left(\frac{300}{35} = 8.6\right)$. Como es permisible pensar con el aumento de prestaciones de servicios aumentan la cantidad de posibles usuarios conectados al sistema.

3.3.3. Pruebas de aceptación

“La validación del software se logra mediante una serie de pruebas que demuestren que se cumple con los requisitos… Al construir software personalizado para un cliente se aplica una serie de pruebas de aceptación que permiten al cliente validar todos los requisitos.” (Pressman, 2007). Teniendo en cuenta que la solución propuesta será utilizada por un número creciente de usuarios con distintos roles dentro del negocio (técnicos, especialistas, proveedores y clientes del centro de soporte) no es práctico realizar pruebas de aceptación formales para cada uno. “La mayoría de los constructores de productos de software emplean procesos llamados prueba alfa y prueba beta para descubrir errores que sólo el usuario final podría detectar” (Pressman, 2007). Las pruebas alfa se realizan en el lugar de trabajo del desarrollador, de esta manera se

crea un ambiente controlado donde los usuario finales pueden trabajar con el sistema y registrar los errores y problemas.

Para la aplicación de estas pruebas se generó el documento **Caso de prueba de aceptacion.doc** que se encuentra en el expediente de proyecto *Entregables*, el objetivo fundamental de estas pruebas es validar si las HU fueron implementadas según las especificaciones del Centro de Soporte.

Tabla 10: Iteraciones de los casos de prueba basados en HU

Iteración	No conformidades
1ra	16
2da	4
3ra	0

3.4. Aprobación de la solución

Luego de la culminación de las pruebas unitarias y de sistema se procedió a realizar las pruebas de aceptación; estas últimas bajo un ambiente controlado son capaces de determinar si el software realiza lo que desea el cliente. Terminadas estas pruebas y corregidas las no conformidades el cliente avaló la solución con la entrega del Certificado de Aceptación del Producto por parte de Luis J. Guzmán Hernández director del Centro de Soporte UCI. Este documento puede ser consultado en el **Anexo 3**.

3.5. Validación de la investigación

A consideración de los autores el rendimiento laboral es la cantidad de problemas procesados en una jornada laboral. Se estableció una comparación entre el sistema ManageEngine ServiceDesk y con el módulo desarrollado, realizando el proceso de Gestión de problemas. El tiempo de solución de un problema es el tiempo que se demoran los técnicos y especialistas en llevar a cabo las tareas de la Gestión de problemas dígase: generar problema, determinar causas, buscar solución, asignar una solución al problema y enviar la solución al cliente. Se tuvo en cuenta las actividades del proceso y se obtuvieron los tiempos promedios de cada actividad. La disminución del tiempo de solución de los problemas favorecerá el rendimiento laboral.

Tabla 11: Tabla comparativa antes y después

Actividades para gestionar un problema			
No.	**Actividades**	**ManageEngine ServiceDesck (minutos)**	**Módulo (minutos)**
1	Generar problema	2	1.5
2	Determinar causas	---	2
3	Buscar solución	10	7
4	Asignar una solución al problema	1	0.45
5	Enviar solución al cliente	3	---
	TOTAL	16	10.95

El desarrollo del módulo Gestión de problemas posibilita un aumento en el rendimiento laboral de los técnicos y especialistas a partir de la disminución del tiempo empleado para realizar las actividades que conforman la Gestión de problemas. Atendiendo a los resultados de la Tabla 11 se puede afirmar que hay una disminución de 5 minutos en el procesamiento de los problemas. Esta disminución favoreció al rendimiento laboral como se muestra en la Tabla 12.

Tabla 12: Rendimiento laboral

	ManageEngine ServiceDesck (problemas resueltos/jornada laboral)	GePro (problemas resueltos/jornada laboral)
Rendimiento laboral	14	21

Se evidencia el aumento del rendimiento en un 33.33%, lo que favorece el mayor aprovechamiento de la jornada laboral. Este aumento incrementa la cantidad de solicitudes atendidas mejorando la calidad del servicio de soporte y de igual manera tributando a una mejor atención al cliente. Atendiendo a lo antes planteado se demuestra que el desarrollo del módulo de Gestión de problemas contribuye a este proceso, lo que disminuye el tiempo de solución de problemas y aumenta el rendimiento de los trabajadores.

3.6. Conclusiones del capítulo

Este capítulo abordó las fases finales del proceso de desarrollo establecido por la metodología. En la fase de desarrollo se estableció el estándar de codificación a utilizar lo que permitió proyectar la calidad del mejor artefacto que genera la metodología, el código. Se definió

además el manejo de la seguridad del sistema utilizando RBAC lo cual permitió controlar a que parte del software puede acceder cada usuario. En la fase de pruebas se examinó el código desde su parte más pequeña a través de las pruebas unitarias. Se comprobó la respuesta de la propuesta ante la concurrencia de 300 usuarios realizando operaciones. Por último se estableció un ambiente controlado dónde se comprobó que la implementación realizada se ajustaba a lo especificado en las HU. De esta manera se concluyó el proceso de desarrollo y se determinó que el producto generado cumple con todas las exigencias del cliente. Se realizó un análisis de las variables del problema y luego de efectuar la comparación de los resultados del sistema implantado y la propuesta de solución se determinó que el módulo reduce el tiempo de solución de los problemas y aumenta la productividad de los técnicos y especialistas.

Conclusiones generales

- El análisis de los sistemas que implementan la Gestión de problemas utilizando ITIL arrojó que no era factible utilizar ninguna de estas soluciones debido a que no se ajustan a las características requeridas en el Centro de Soporte UCI, y por lo tanto confirmó la necesidad de desarrollar la nueva propuesta.
- El estudio de las herramientas, tecnologías y metodologías permitió determinar cuáles se ajustaban a las características del marco de trabajo Grails, equipo de desarrollo y producto a realizar. Utilizando Intellij IDEA, PostgreSQL y Apache Tomcat para el desarrollo del producto de esta investigación.
- El diseño permitió obtener una vista de la estructura estática del sistema teniendo en cuenta distintos patrones de diseño como el bajo acoplamiento, experto, creador y la alta cohesión. También se contribuyó a aumentar la escalabilidad del producto mediante el uso de estilos arquitectónicos como el MVC, lo cual propició la alta reutilización de las clases diseñadas.
- El proceso de implementación permitió desarrollar el producto a partir del diseño elaborado, cumpliendo con todos los requisitos identificados para la Gestión de problemas según ITIL 2011. El uso del estándar de codificación empleado contribuyó a la legibilidad del código y la comunicación fluida entre los programadores, lo cual permitirá proveer una guía para futuros mantenimientos del producto.
- El proceso de pruebas permitió garantizar un mínimo de calidad aceptable del producto desarrollado, lo cual se tomará como validación del cumplimiento de los requisitos en el módulo implementado. La aplicación de las pruebas de sistema permitió comprobar el comportamiento del módulo en un ambiente de concurrencia de hasta 300 usuarios. Se comprobó que cumplía con todas las especificaciones del cliente y se obtuvo el certificado de aceptación.

Recomendaciones

Realizar un sistema de análisis de la información que gestione de forma automática las propuestas de soluciones basándose en un sistema de casos. La implementación de un sistema que utilice Inteligencia Artificial para proveer conocimiento contribuiría con la Gestión de conocimiento en el Centro de Soporte UCI.

Bibliografía

Linux New Media Spain SL. 2014. Linux Megazine. [En línea] 2014. [Citado el: 15 de Febrero de 2014.] http://www.linux-magazine.es/issue/50/008-009_Inseguridades50.pdf.

Abrahamsson, P., y otros. 2002. *Agile software development methods Review and analysis.* s.l. : VTT Publications, 2002.

Andino, Arias y Giovanny, Franklin. 2008. Diseño y construcción de un sistema de gestión de incidentes para un service desk fundamentado en Itil. *bibdigital.epn.edu.ec.* [En línea] Julio de 2008. http://bibdigital.epn.edu.ec/handle/15000/707.

APM Group Ltd. 2013. Itil Official Site. [En línea] 2013. [Citado el: 20 de Septiembre de 2013.] http://www.itil-officialsite.com/.

—. 2011. Itil Oficial Site: Glossary of terms and definitions. [En línea] 29 de Julio de 2011. [Citado el: 20 de Noviembre de 2013.] www.itil-officialsite.com.

APM Group. 2013. Pink Elephant The IT Management Experts: Official Site. [En línea] 2013. [Citado el: 3 de Noviembre de 2013.] http://www.pinkelephant.com/PinkVERIFY/.

Bardallo, Josep. 2009. Slideshare: Gsx Problem Management. [En línea] Mayo de 2009. [Citado el: 2 de Noviembre de 2013.] http://www.slideshare.net/Josep_Bardallo/gsx-problem-management.

Beck, K. 1999. *Extreme Programming Explained. Embrace Change.* s.l. : Pearson Education, 1999.

BMC Software, Inc. 2013. BMC Software. [En línea] 2013. [Citado el: 15 de Enero de 2014.] http://www.bmc.com/products/product-listing/service-desk-express-suite.html.

Bologna, J. y Walsh, A. M. 1997. *The Accountant´s Handbook of Information Technology.* s.l. : John Wiley and Sons, 1997.

Bon, J. *Fundamentos de la gestión de servicios de TI basada en ITIL V3.* Berlín : Van Haren Publishing.

Brito Calahorro, Nacho. 2009. *Manual de desarrollo web con Grails.* [Documento] s.l. : ImaginaWorks Software Factory S.L.U, 2009.

Centro de Soporte UCI. 2013. Centro de Soporte. [En línea] 2013. [Citado el: 15 de Noviembre de 2013.] https://soporte.uci.cu/.

Chappell, D y Jewell, T. 2001. *Java Web Service.* California : O'Reilly Media, 2001.

Classifying ITIL processes; a taxonomy under tool support aspects. **Brenner y Michael. 2006.** s.l. : IEEE, 2006.

Consuegra Rabi, Elide y Gonzalez Quesada, Armin Alexander. 2010. Estrategia de implementacion del Modelo Cobit en la Facultad 15 de la Universidad de las Ciencias Informaticas. *Repositorio Institucional.* [En línea] 2010. http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_03354_10.

Cordoví García, Carlos Heriberto y Hernández Rizo, Claudia. 2013. *Consumo de datos enlazados mediante búsqueda textual y facetada.* [Documento] Habana : Universidad de las Ciencias Informáticas, 2013.

Cruz Rojas, Orlando y Puig Gonzalez, Claudio. 2009. Sistema basado en casos para la asistencia al proceso de soporte y ayuda en la gestion de incidencias tecnologicas en la Universidad de Ciencias Informaticas. *Repositorio Institucional.* [En línea] 2009. http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_2487_09.

de Armas Roche, Eduardo. 2010. Procedimiento para la gestion de incidencias en los proyectos de software de gestion de la UCI. *Repositorio Institucional .* [En línea] 2010. http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_03241_10.

EasyVista. 2013. EasyVista. *Sitio web oficial de EasyVista.* [En línea] 2013. [Citado el: 7 de Noviembre de 2013.] http://www.easyvista.com/es.

Echeverry Tobón, Luis Manuel y Delgado Carmona, Luz Elena. 2007. *Caso práctico de la metodología ágil XP al desarrollo de software.* Pereira : Universidad Tecnológica de Pereira, 2007.

Elaboración del plan de Gestión de Problemas y Cambios para la Empresa Eléctrica Quito basado en ITIL. **Palma Tigasi, Jimmy Javier y Pullupaxi Ataballo, Mayra Alejandra. 2013.** Quito : QUITO/EPN/2013, 2013.

Embracing Change with Extreme Programming. **Beck, Kent. 1999.** 1999, Computer (revista de la IEEE Computer Society), Vol. 32, No. 10, págs. 70-77.

ETECSA. 2014. Empresa de Telecomunicaciones de Cuba S.A. [En línea] ETECSA, 2014. http://www.etecsa.cu.

Feo Gamio, Lic Gema y García Pérez, Dra Ana María. 2013. MODELO TÉCNICO DE SOPORTE A PROYECTOS INFORMÁTICOS CON TECNOLOGÍAS LIBRES. *Repositorio Institucional.* [En línea] 29 de Septiembre de 2013. http://repositorio_institucional.uci.cu//jspui/handle/ident/7986.

Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. 2009. *Metodologías Tradicionales Vs. Metodologías Ágiles.* 2009.

Fontela, Alvaro. 2013. OpenWebCMS. [En línea] 2013. [Citado el: 1 de Febrero de 2014.] http://openwebcms.es/2013/que-es-bootstrap/.

Gestión del conocimiento. **Pérez, Augusto, y otros. 2005.** Buenos Aires : Grupo Norma Editorial, 2005.

GoPivotal, Inc. 2013. Grails: Oficial Site. [En línea] 2013. [Citado el: 10 de Septiembre de 2013.] http://grails.org/.

GoPivotal, Inc. 2013. Spring: Home Site. [En línea] 2013. [Citado el: 16 de Octubre de 2013.] http://projects.spring.io/spring-security/.

Gravitar. 2013. Gravitar: información sin límite. [En línea] 2013. [Citado el: 30 de Octubre de 2013.] http://www.gravitar.biz.

Implementing an ITIL-based IT service management measurement system. **Lahtela, Antti, Jantti, Marko y Kaukola, Jukka. 2010.** s.l. : IEEE, 2010. 5432788.

Ingeniería Dric S.A. de C.V. 2013. Ingeniería Dric soluciones integrales TI. [En línea] 2013. [Citado el: 11 de Octubre de 2013.] http://manageengine.mx/service-desk/index.html.

IT Governance Institute. 2008. *Alineando COBIT 4.1, ITIL V3 e ISO/IEC 27002 en beneficio de la empresa.* s.l. : IT GOVERNANCE INSTITUTE, 2008.

IT Process Maps GbR. 2011. Introducción a ITIL® Versión 3 y al Mapa de Procesos ITIL® V3. [En línea] Marzo de 2011. http://albinogoncalves.files.wordpress.com/2011/03/introduccion-mapa-de-procesos-itil-v3.pdf.

ITIL Implementation: The Role of ITIL Software and Project Quality. **Roar Eikebrokk, Tom y Iden, Jon. 2012.** Los Alamitos, CA, USA : IEEE Computer Society, 2012.

itSMF Ltd. 2007. *An introductory overview of ITIL v3.* s.l. : The UK Chapter of the itSMF, 2007. 0-9551245-8-1.

Jeffries, R., Anderson, A. y Hendrickson, C. 2001. *Extreme Programming Installed.* s.l. : Addison-Wesley, 2001.

JetBrains s.r.o. 2013. JetBrains: Oficial Site. [En línea] 2013. [Citado el: 2013 de Septiembre de 14.] http://www.jetbrains.com/idea/.

Larman, Craig. 2007. *UML y Patrones.* s.l. : Printice Hall, 2007.

—. *UML y Patrones.* s.l. : Printice Hall.

Losada Leon, Efrain y Cuba Garcia, Katiuska. 2009. Sistema de Control y Gestion de las Interrupciones y Servicios de las Tecnologias de la Informacion y las Comunicaciones. *Repositorio Institucional.* [En línea] Mayo de 2009. http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_2403_09.

LSIS. 2004. Unidad Docente de Ingeniería del Software (UDIS) . [En línea] 8 de Junio de 2004. http://is.ls.fi.upm.es/.

Machuca, Sergio, y otros. 2007. Modelo de grafos para el estudio de la disponibilidad y la gestión de los niveles de servicio en servicios de IT. *Google Scholar.* [En línea] 2007. [Citado el: 10 de Marzo de 2014.] http://sedici.unlp.edu.ar/handle/10915/23368.

ManageEngine ServiceDesk Plus. 2013. Manage Engine. [En línea] Zoho Corporation Pvt. Ltd, 2013. [Citado el: 10 de Diciembre de 2013.] http://www.manageengine.com/products/service-desk/.

Medina Somoza, Olivia Angelica, Rosell Quesada, Pedro Alain y Felipe Alvarez, Yarelis. 2007. Sistema de Gestion de Informacion para la Casa de Autoria DVD de la UCI Basado en ITIL. *Repositorio Institucional.* [En línea] 2007. [Citado el: 6 de Febrero de 2014.] http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_0609_07.

Molina Conde, Kamelia y Guirado Más, Yorlen. 2012. *Módulo de gestión de la información asociada a la Aduana en la Ventanilla Única.* [Documento] Habana : Universidad de las Ciencias Informáticas, 2012.

Molina, Marlon. 2011. Slideshare. [En línea] 2011. [Citado el: 22 de Noviembre de 2013.] http://www.slideshare.net/tecnofor/novedades-itil-edicin-2011-v3.

Mozilla. 2014. Mozilla Hispano. [En línea] 2014. [Citado el: 2 de Febrero de 2014.] http://www.mozilla-hispano.org/desarrollando-con-twitter-bootstrap-parte-i/.

OGC ITIL. **The ITIL® Lifecycle Publication Suite. 2011.** 2011. ISBN:9780113310500.

Ortas, A. 2001. *Aproximación a la Ingeniería de Requerimientos.* Uruguay : Universidad ORT Uruguay, 2001.

OSIATIS S.A. 2011. ITIL-Gestión de Servicios TI. [En línea] 2011. [Citado el: 15 de Octubre de 2013.] http://itil.osiatis.es/Curso_ITIL/.

Otto, Mark y Thornton, Jacob. 2013. LibrosWeb. [En línea] 2013. [Citado el: 4 de Noviembre de 2013.] http://librosweb.es/bootstrap_3/.

Oval Riveron, Yaima. 2007. Servicio de Soporte Tecnico utilizando la tecnologia Service-Desk. *Repositorio Institucional.* [En línea] 2007. [Citado el: 2 de Marzo de 2014.] http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_0868_07.

Padanés, María Carmen y Letelier, Patricio. 2007. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : Laboratorio de Sistemas de Información. Departamento de Sistemas Informáticos y Computación., 2007.

Pérez Huebe, Ma. de Lourdes. 2005. *Ingeniería de Requerimientos.* Pachuca : Universidad Autónoma del Estado de Hidalgo, 2005.

Planeación y Gestión de Proyectos Informáticos Parte 2: Estimación del Esfuerzo. **B, Tabares y Silvia, Dr.Marta. 2011.** Medellín : Universidad de Medellín, 2011.

Pressman, Roger. 2007. *Ingeniería del Software, Un enfoque práctico.* s.l. : Mc Graw Hill, 2007.

roactivanet. [En línea]
http://www.proactivanet.com/UserFiles/File/Noticias/Gesti%C3%B3n%20del%20Conocimiento_Inteli%281%29.pdf.

Rodríguez Corbea, Maite y Ordoñez Pérez, Meylin. 2007. *La metodología XP aplicable al desarrollo del software educativo en Cuba.* [Documento] Habana : Universidad de las Ciencias Informáticas, 2007.

Ruiz Alvarez, Lisset. 2010. Propuesta del proceso prestacion de servicio para proyectos de servicio de la Universidad de Ciencias Informaticas. *Repositorio Institucional.* [En línea] 2010. http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_03400_10.

Sánchez Ortiz, Susana, Pérez Benitez, Alfredo y Oval Riverón, Yaima. 2012. NOVADESK: SISTEMA PARA LA GESTIÓN DE SOPORTE. *Respositorio Institucional.* [En línea] Marzo de 2012. http://repositorio_institucional.uci.cu//jspui/handle/ident/3831.

Sánchez, Fernando. 2014. CMMI vs. ITIL: dos enfoques complementarios . *Computer World.* [En línea] IDG Communications S.A.U., 2014. http://www.computerworld.es/archive/cmmi-vs-itil-dos-enfoques-complementarios.

Schwaber, Ken. 2011. Scrum Guide. *Scrum Official Site.* [En línea] 2011. [Citado el: 15 de Febrero de 2014.] https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf.

SICELCA IT Systems. 2013. Consultoría en Gestión de Tecnología de Información. [En línea] 2013. http://www.sicelca.com/.

Softel. 2013. Softel Sitio Oficial. [En línea] Softel, 2013. http://www.softel.cu.

Sommerville, Ian. 2005. *Ingeniería del software.* Madrid : Pearson Educación S.A., 2005. 84-7829-074-5.

Sun microsystems. 2013. IT Infrastructure Library: Las Mejores Prácticas para la Gestión de Servicios de TI en su. [En línea] 2013. sun.es/services/itil.

Sun Microsystems. 2013. IT Infrastructure Library: Las Mejores Prácticas para la Gestión de Servicios de TI en su. [En línea] 2013. sun.es/services/itil.

Systematic Approach to Successful Implementation of ITIL. **Ahmad, Norita y Shamsudin, Zulkifli M. 2013.** s.l. : Procedia Computer Science, 2013, Vol. 17. 1877-0509.

Tecnofor. 2013. Tecnofor: Escuela especializada en enseñar gestión y tecnología. [En línea] 2013. [Citado el: 10 de Noviembre de 2013.] http://www.tecnofor.es.

The Apache Software Foundation . 2013. Apache Tomcat. [En línea] 2013. [Citado el: 25 de Septiembre de 2013.] http://tomcat.apache.org/.

The jQuery Foundation. 2014. The jQuery Foundation. [En línea] 2014. https://jquery.org/.

The PostgreSQL Global Development Group . 2013. PostgreSQL: Official Site. [En línea] 2013. [Citado el: 13 de Septiembre de 2013.]

Tjassing, Ruby. 2008. *Gestion de Servicios TI basado en ITIL: Guia De Bolsillo Spanish Version.* s.l. : Van Haren Publishing, 2008.

Transforming IT service management-the ITIL impact. **Cater-Steel, Aileen, Toleman, Mark y Tan, Wui-Gee. 2006.** s.l. : Australasian Association for Information Systems, 2006.

UCI. 2014. Resolución 179/14 Resolución rectoral de migración a software libre. [En línea] 22 de Febrero de 2014. http://mediaserver.uci.cu/periodico/Resoluci%C3%B3n-Migraci%C3%B3n-SWL-22_04_2014.pdf.

—. 2014. Universidad de las Ciencias Informáticas. [En línea] 2014. http://www.uci.cu/?q=entorno-productivo.

Universidad de las Ciencias Informáticas. 2014. Resolución 179/14 Resolución rectoral de migración a software libre. [En línea] 22 de Febrero de 2014. http://mediaserver.uci.cu/periodico/Resoluci%C3%B3n-Migraci%C3%B3n-SWL-22_04_2014.pdf.

Valdes Lopez, Yuniel. 2010. Estrategia para el diseño de niveles de servicios utilizando la metodologia ITIL en un entorno de producción de software en la Universidad de las Ciencias Informáticas. *Repositorio Institucional.* [En línea] 2010. http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_03357_10.

von Solms, Basie. 2005. Information Security governance: COBIT or ISO 17799 or both? *Computers & Security.* 2005, Vol. 24, 2.

Zamora Dominguez, Eylen y Amelo Caballero, Ivan Doniek. 2008. Análisis y Diseño de un Sistema de Gestión para el Soporte a Servicios de Tecnologías de la Información. *Repositorio Institucional.* [En línea] 2008. [Citado el: 25 de Marzo de 2014.] http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_1652_08.

Zapata Retrepo, Elena. 2010. *Diplomatura Articulación de Tecnologías de Información y Comunicación TIC para el desarrollo de competencias en las áreas de lengua castellana.* Bogotá : Universidad pontífica Bolivariana, 2010.

Glosario de términos

Incidente: Es una interrupción no planificada de un servicio de TI o la reducción en la calidad de un servicio de TI. La falla de un elemento de configuración que no ha afectado aún el servicio es también un incidente - por ejemplo, la falla en un disco de un conjunto de discos espejos.

Problema: Es una causa de uno o más incidentes. Usualmente no se conoce la causa al momento de crear el registro del problema, y el proceso de Gestión de problemas es responsable de la investigación posterior.

Servicio: Es un medio de entregar valor a los clientes, al facilitar los resultados que los clientes quieren lograr sin apropiarse de los costos y riesgos específicos. A veces se utiliza el término 'Servicio' como sinónimo de servicio base, servicio de TI o paquete de servicios.

Base de conocimiento: Es una base de datos lógica que contiene datos e información utilizados por el sistema de gestión del conocimiento del servicio.

Anexos

Anexo 1: Umbrales de la métrica TOC y RC.

TOC		
Atributos de calidad	**Categoría**	**Umbrales**
Responsabilidad	Baja	x ≤ promedio
	Media	promedio ≤ x ≤ 2 * promedio
	Alta	x > promedio
Complejidad de implementación	Baja	x ≤2* promedio
	Media	promedio ≤ x ≤ 2 * promedio
	Alta	x > 2*promedio
Reutilización	Baja	x > promedio
	Media	promedio ≤ x ≤ 2 * promedio
	Alta	x ≤ promedio

RC		
Atributos de calidad	**Categoría**	**Umbrales**
Complejidad de mantenimiento	Baja	x ≤ promedio
	Media	promedio ≤ x ≤ 2 * promedio
	Alta	x > 2*promedio
Reutilización	Baja	x >2* promedio
	Media	promedio ≤ x ≤ 2 * promedio
	Alta	x ≤ promedio
Cantidad de pruebas	Baja	x ≤ promedio
	Media	promedio ≤ x ≤ 2 * promedio
	Alta	x >2* promedio

Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	x>2

Anexo 2: Entrevista realizada a los especialistas y técnicos del Centro de Soporte UCI.

Entrevista

(Realizada al jefe de departamento, especialistas, técnicos)

Objetivo: comprobar cómo se desarrolla el proceso de solución de problemas en el Centro de Soporte.

Compañero o compañera.

Se necesita su valiosa colaboración para la realización de esta investigación, de sus respuestas dependen los resultados de la misma. **MUCHAS GRACIAS**.

Datos generales.

- ✓ Nombre y Apellidos_____
- ✓ Años de experiencia_____
- ✓Rol que desempeña __ Especialista

  __ Técnico

Aspecto a encuestar

1. ¿Cuáles son los pasos que se llevan a cabo para dar solución a un problema? ¿Qué tiempo demora en llevarlos a cabo?

   a. ¿Le gustaría que se realizara de otra manera el proceso de gestión de problemas en aras de agilizar el proceso? Argumente en caso de ser afirmativa su respuesta.

2. ¿Qué tiempo usted demora como promedio en determinar la causa de un problema? ¿Por qué?

3. ¿Qué tiempo usted demora en encontrar la solución a un problema? ¿Por qué?

4. La respuesta a un mayor número de problemas aumenta la productividad del centro. ¿A cuántos problemas usted le da solución en un día?

5. ¿Qué beneficios encuentra usted en la realización de un sistema capaz de almacenar la descripción de los problemas, las causas y sus soluciones?

Anexo 3: Certificado de Aceptación del Producto.

CENTRO DE SOPORTE

UCI
Universidad de las Ciencias Informáticas

Por este medio avalo que la tesis de grado "Módulo de Gestión de Problemas para el Centro de Soporte de la Universidad de las Ciencias Informáticas" de los tesistas Andy Daniel Meriño Coronado y Yorgüy Antonio Batista Desdin es de gran importancia para el Centro de Soporte y la aplicación generada formará parte de la Plataforma de Gestión de Servicio. El sistema cuenta con los requerimientos especificados por el clientes, así como un buen diseño. Además cumple con lo establecido por la metodología de gestión de servicio.

Sobre el Trabajo de Diploma realizado considero que favorece la calidad de la gestión de servicio, sirviendo de soporte y ayuda para los desarrolladores del proyecto productivo que dará continuación al trabajo presentado, sentando las bases para la futura construcción de la plataforma concebida.

Los diplomantes han puesto en evidencia los conocimientos adquiridos siendo capaces de ponerlos en practica para solucionar la problemática planteada. Han mantenido una aptitud muy responsable y comprometida con la investigación, logrando realizar un excelente trabajo individual y en equipo.

Y para que así conste se da por aceptada el día 28 del mes 05 del año 2014 por:

UCI Universidad de las Ciencias Informáticas Centro de Soporte

Director de Centro de Soporte Luis J. [illegible]mán Hernández