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

**Título:** Sistema Informático para la Evaluación de la seguridad del Control de Acceso de Sistemas de Información.

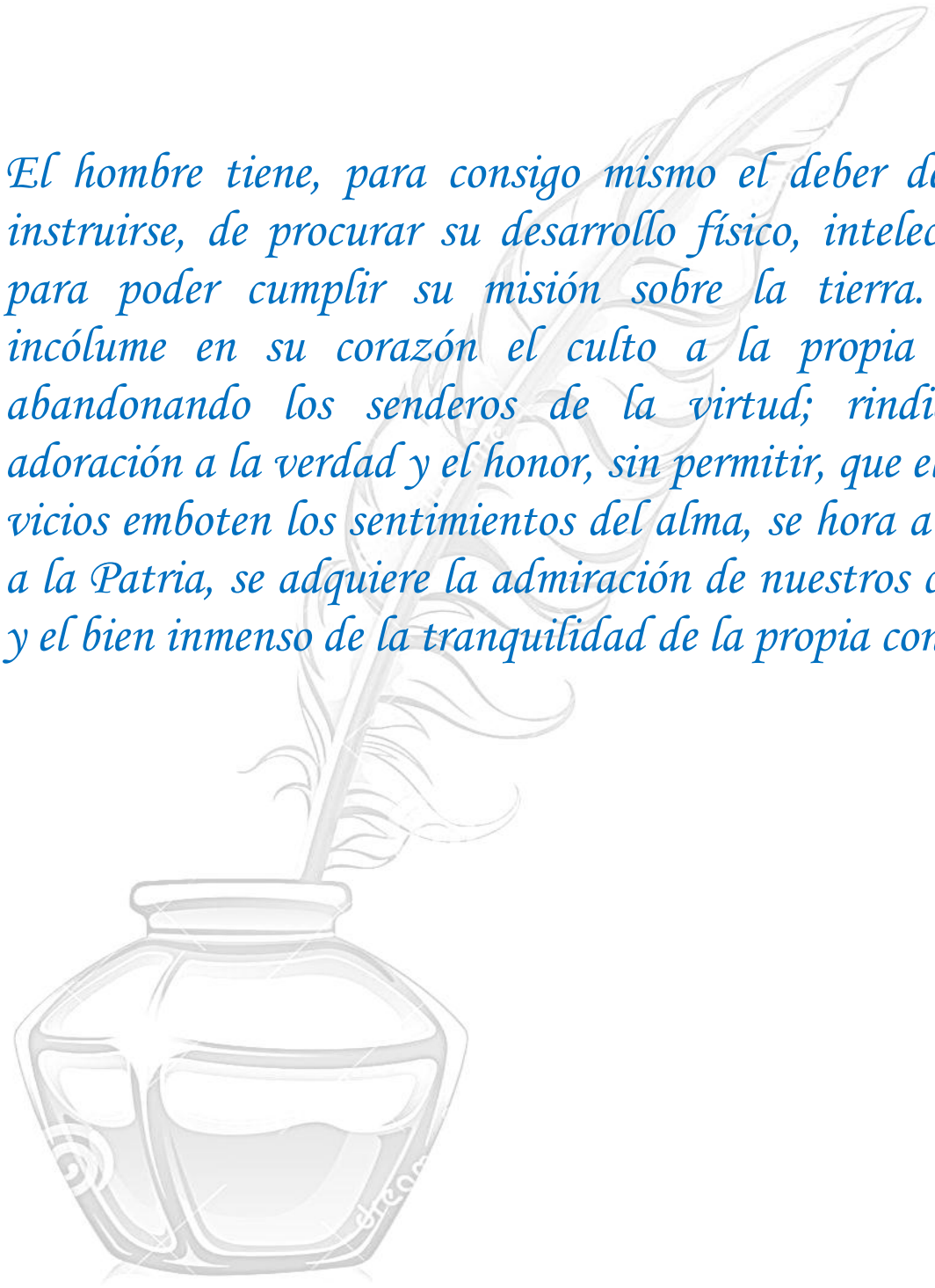

**Autor:**
Wilfredo Palma Pérez

**Tutores:**
Ing. Maribel Silva Muñoz
Ing. Juniel Tamayo Hernández

**Universidad de las Ciencias Informáticas**
Ciudad de la Habana, 2014

*El hombre tiene, para consigo mismo el deber de estudiar, de instruirse, de procurar su desarrollo físico, intelectual y moral, para poder cumplir su misión sobre la tierra. Conservando incólume en su corazón el culto a la propia dignidad; no abandonando los senderos de la virtud; rindiendo perenne adoración a la verdad y el honor, sin permitir, que el egoísmo y los vicios emboten los sentimientos del alma, se hora a Dios, se sirve a la Patria, se adquiere la admiración de nuestros conciudadanos y el bien inmenso de la tranquilidad de la propia conciencia.*

**A:. M:.**



# DECLARACIÓN DE AUTORÍA

Yo, Wilfredo Palma Pérez declaro ser autor del presente Trabajo de Diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma del Autor

Wilfredo Palma Pérez

_____

Firma del Tutor

Ing. Maribel Silva Muñoz

_____

Firma del Tutor

Ing. Juniel Tamayo Hernández

# DATOS DE CONTACTO

**Autor:** Wilfredo Palma Pérez

**Correo Electrónico:** wpalma@estudiantes.uci.cu

**Tutor:** Ing. Maribel Silva Muñoz.

**Correo Electrónico:** msmunoz@uci.cu

**Tutor:** Ing. Juniel Tamayo Hernández

**Correo Electrónico:** jthernandez@uci.cu

# DEDICATORIA

Dedico completamente el resultado, esfuerzo y lauros alcanzados durante todos mis años como estudiante y especialmente los que he dedicado en el noble empeño de realizar esta hermosa carrera universitaria y esta Tesis, a toda mi familia, destacando principalmente:

A mi madre María por quererme, cuidarme y creer en mí.
A mi padre Wilfredo (Ito) por quererme y estar siempre cuando se necesita.
A mi abuela Marisol por quererme y ayudarme siempre.
A mi abuelo Isidro por incitarme siempre al estudio y a la superación.
A mi abuelo Orlando por hacerme reír siempre con sus locuras.
A mi abuelo Carlos por tenerme siempre presente.
Por último y especialmente a mi hermano Wilenis por dejarme ser siempre en la vida su ejemplo a seguir.

# AGRADECIMIENTOS

A mi familia, por apoyarme en todo momento, especialmente en los más difíciles, por confiar en mí desde el momento en que dije “Ya no voy a ser Médico, voy a ser Ing. Informático”, aunque reconozco que llevo un médico dentro.

A mis tutores Maribel, siempre atenta y amable ante los problemas sin perder nunca su sonrisa en los momentos de más trabajo y dificultad, y Juniel(JT) que más que mi profesor de Programación y tutor, es mi amigo.

A mis amigos de toda la vida, los que estudiamos y pasamos juntos casi todo el tiempo, aunque a pesar de que las dificultades separen los caminos que nos trazamos siempre encontramos un rato para compartirlo entre nosotros, especialmente a Marlon y Jorge.

A mis amigos de la UCI, los que estuvieron en mi grupo y los que no, los que me acompañaron en los momentos de alegría sin perder nunca su brillo propio y me ayudaron en los momentos de dificultad brillando aún más, especialmente a Yorgüy, Andy, Mailyn, Abel, Mario, Gaby, Zoemi, Zoíma, Isis y Lisandra.

A mis profesores de la carrera, los que mostraron su lado humano, gracias por confiar en mí y ayudarme en mis estudios, especialmente a Didier, Vilma, Odette, Yasser y Katia.

A los profesores que aunque nunca me dieron clases me brindaron siempre sus conocimientos para reforzar los míos sin importar el día ni la hora, principalmente a Carlos Luis, Yunior, Anie, Pedro y Sergio Luis.

A todos los locos del Taller, con los que pase mucho tiempo y que me manifestaron muchas veces su ayuda y apoyo, especialmente a Jorge (Tío), Tomás (Tommy), Alberto, Pedrito, Yenia, Eduardo, Alain, Amedh, Akell, Yordan, Miriam y Rigo.

A mis amigos de la guagua por hacer que los viajes sean más agradables, con los que me reí mucho, especialmente a Massiel, Nurisel, Cristina y Lázaro.

A todos los que de una forma u otra han contribuido con mi paso por la UCI y a los que han puesto su granito de arena en los demás aspectos de mi vida.

A todas estas personas ¡Muchas Gracias!

# RESUMEN

Las Tecnologías de la Información y las Comunicaciones aplicadas a los procesos empresariales constituyen el eslabón fundamental de las organizaciones para lograr mayor eficiencia en la gestión interna. Los Sistemas de Información (SI) gestionan los procesos empresariales y utilizan una serie de elementos esenciales para garantizar la seguridad de la información y el control de acceso. Evaluar la fiabilidad de los procesos de Identificación y Autenticación, Autorización y Auditoría contribuye a la seguridad de los SI. En la actualidad la evaluación de la seguridad del control de acceso se realiza según regulaciones, normas y estándares. Sin embargo este proceso no se encuentra informatizado de manera que permita la persistencia de la información y la generación de reportes. El objetivo del trabajo es desarrollar un sistema informático que integre indicadores para determinar la fiabilidad de la evaluación de la seguridad del control de acceso de los procesos de Identificación y Autenticación, Autorización y Auditoría de los SI. El sistema informático propuesto contribuye a la evaluación de riesgos de seguridad y a la toma de decisiones en las organizaciones.

# ÍNDICE

Introducción ....1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA ....8
1.1- Normas o estándares, recomendaciones, resoluciones, controles, decretos y métricas. 8
- ISO/IEC 27001 ....8
- ISO/IEC 27002 ....8
- ISO/IEC 27004 ....8
- ISO/IEC 15504 ....8
- ISO/IEC 15408 Criterio Común (CC) ....9
- Objetivos de Control para la Información y la Tecnología relacionada ....9
- MAGERIT ....9
- Biblioteca de Infraestructura de TI ....9
- Publicación Especial 800-53 del NIST ....9
- Proyecto de los 10 Mejores Elementos ....9
- Las guías de OWASP ....9
- Norma de Buenas Prácticas para la Seguridad de la Información ....10
- Resolución 127/2007 ....10
- Decreto Ley 281 ....10
- Métricas de Seguridad del Centro de Seguridad de Internet: ....10

1.2- Soluciones existentes para la evaluación de la seguridad del control de acceso ....10
- Listas de Control de Acceso ....10
- Control de Acceso Discrecional ....11
- Control de Acceso Obligatorio ....11
- Control de Acceso Basado en Roles ....11
- Sistema de Puntuación de Vulnerabilidades Comunes ....11
- Técnica Ajustada de Puntuación de Riesgos de Negocio ....11
- OSSTM ....12
- Puntuación Lógica de Preferencias ....12

1.3- Sistemas líderes en el mundo para la evaluación de la seguridad del control de acceso ....12
- Gestor de Entidades Tivoli de IBM ....12

- Suite de Gestión de Entidades y Acceso de Oracle ....12

1.4- Metodologías de desarrollo....13
1.4.1- Proceso Unificado de Rational ....13
1.4.2- SCRUM ....13
1.4.3- Programación Extrema....14
1.5- Lenguajes de modelado....14
1.5.1- Notación de Modelado de Procesos de Negocio ....15
1.5.2- Lenguaje Unificado de Modelado ....15
1.6- Herramientas de Ingeniería de Software Asistida por Computadora ....16
1.6.1- Rational Rose....16
1.6.2- Visual Paradigm ....17
1.7- Lenguajes de programación ....18
1.7.1- C++ ....18
1.7.2- C Sharp ....18
1.7.3- Java....18
1.8- Marcos de trabajo para Java ....20
1.8.1- Swing ....20
1.8.2- API de Persistencia de Java....20
1.9- Entornos de Desarrollo Integrado ....20
1.9.1- Visual Studio ....21
1.9.2- NetBeans....21
1.10- Sistemas de Gestión de Bases de Datos....22
1.10.1- MySQL ....22
1.10.2- PostgreSQL 9.1....23
1.11- Métrica de Desarrollo de Software....23
1.11.1- Tamaño Operacional de Clase....24
1.12- Pruebas ....25
1.12.1- Pruebas Unitarias....25
1.12.2- Pruebas de Aceptación ....26
CAPÍTULO 2: DESARROLLO DE LA SOLUCIÓN INFORMÁTICA....28
2.1- Breve introducción de SIECA ....28
2.2- Indicadores y Criterios utilizados en el desarrollo....28
2.2.1- Indicadores....28

2.2.1- Criterios ..... 29
2.3- Algoritmo Lógica de Puntuación de Preferencias (LSP) adaptado ..... 30
2.4- Requisitos ..... 31
2.4.1- Requisitos Funcionales ..... 32
2.4.2- Requisitos No Funcionales ..... 33
2.5- Historias de Usuario ..... 34
2.6- Tarjetas Clase-Responsabilidad-Colaboración ..... 37
2.7- Arquitectura del sistema ..... 38
2.7.1- Arquitectura en tres capas ..... 38
2.8- Patrones ..... 39
2.8.1- Patrones Generales de Software para Asignación de Responsabilidades ..... 40
2.8.2- Patrones del Grupo de Cuatro ..... 42
2.9- Modelo de Datos ..... 43
Conclusiones del Capítulo ..... 44
CAPÍTULO 3: ANÁLISIS DE LOS RESULTADOS ..... 45
3.1- Métricas de Desarrollo ..... 45
3.1.3- Tamaño Operacional de Clase ..... 45
3.2- Pruebas unitarias ..... 48
3.2.1- Prueba de Complejidad Ciclomática ..... 48
3.2.2- Prueba del Camino Básico ..... 49
3.3- Resultados de las Pruebas de Aceptación ..... 50
Conclusiones del Capítulo ..... 51
CONCLUSIONES ..... 52
REFERENCIAS BIBLIOGRÁFICAS ..... 53
ANEXOS ..... 59
GLOSARIO DE TÉRMINOS ..... 67

# Introducción

*"La creciente globalización, el proceso de internacionalización de las empresas, el incremento de la competencia en los mercados de bienes y servicios, la rapidez en el desarrollo de las tecnologías de información, el aumento de la incertidumbre en el entorno y la reducción de los ciclos de vida de los productos originan que la información se convierta en un elemento clave para la gestión, así como para la supervivencia y crecimiento de la organización empresarial*" (Baryolo, 2012).

Las Tecnologías de la Información y las Comunicaciones (TIC) aplicada a los procesos empresariales se ha convertido en el eslabón fundamental de las organizaciones, y son utilizadas para lograr mayor eficiencia en su gestión interna. Áreas como la contabilidad, administración del personal, planificación comercial, gestión del inventario y control de la producción pueden ser mejoradas en cuanto a valor y racionalización del trabajo. El uso de estas Tecnologías de la Información (TI) propicia un adelanto significativo en la administración de inventarios, contabilidad y planilla de salarios. Además, las empresas pueden desarrollar vínculos colaborativos con otras empresas y clientes para incrementar las ventas y la distribución, o para organizar la gestión de la cadena de suministro.

Todo esto es posible debido al trabajo en grupo de los empleados con los Sistemas de Información (SI) que según (Oficial, 2011) se definen como *"un conjunto organizado de personas, procesos y recursos, incluyendo la información y sus tecnologías asociadas, que interactúan de forma dinámica, para satisfacer las necesidades informativas que posibilitan alcanzar los objetivos de una o varias organizaciones".*

Los SI incluyen la información de una compañía, el material y los recursos de software que permiten a una corporación almacenar y hacer circular estos datos. Por consiguiente, dichos sistemas son capaces de generar y/o difundir información proveniente de los procesos internos de las empresas, incrementar la asesoría a los clientes porque están diseñados para suplir sus necesidades, apoyar el sistema de producción físico y proporcionar información acerca de las operaciones de producción. Debido a esto son capaces de recopilar así como almacenar información relacionada con los recursos humanos para transformarla para

posteriormente distribuirla a los usuarios de la propia organización, además proporcionan a los directivos información sobre el desempeño global de la empresa, mezclar varias líneas de producción o subsistemas, además de servir de ayuda para la toma de decisiones.

*"Todas las empresas, independientemente de su tamaño, organización y volumen de negocio, son conscientes de la importancia de tener implantadas una serie de políticas de seguridad tendentes a garantizar la continuidad de su negocio en el caso de que se produzcan incidencias, fallos, actuaciones malintencionadas por parte de terceros, pérdidas accidentales o desastres que afecten a los datos e informaciones que son almacenados y tratados"* (Trasobares).

Si bien es cierto que a nivel mundial existe una tendencia a aumentar la seguridad de los sistemas y a crear aplicaciones seguras, según un estudio del Centro Nacional de Inteligencia también es cierto *"que el ritmo de crecimiento de las mismas que afectan a los SI ha sido prácticamente exponencial"* (Centro Nacional de Inteligencia - Centro Nacional Criptológico, 2009). Según medios de prensa internacionales algunos ejemplos de estos ataques recientes son:

- Enero *"el departamento de Energía de Estados Unidos anunció que en enero fue atacado por piratas informáticos que le robaron datos sobre sus funcionarios y subcontratistas"* (Infobae, 2013).
- *"6 de febrero se supo que la Reserva Federal de los EEUU fue atacada por piratas que obtuvieron información de la entidad"* (Infobae, 2013).
- Abril: *"A través de Twitter, MTGox reconoce que está luchando contra un ataque de denegación de servicios (DDoS), que consiste en un bombardeo de datos sobre la página, que no puede asumir y cae"* (El País, 2013).
- En el mes de mayo *"las páginas web del Gobierno holandés, englobadas bajo la dirección* Rijksoverheid.nl *(Gobiernonacional.nl) han sufrido un ciberataque que ha entorpecido el acceso a miles de personas"* (El País, 2013)."
- *"11 Abril: Se produce un ataque masivo de fuerza bruta para acceder a sitios basados en WordPress, con intención de insertar código malicioso"* (Wordpress, 2013).
- Abril: *"Un ataque informático dejó sin servicio la página web del Congreso de los Diputados"* (El País, 2013).

- *"Un pirata informático intervino las cuentas de Twitter del presidente en funciones de Venezuela, Nicolás Maduro, y del oficialista Partido Socialista Unido de Venezuela"* (El País, 2013).
- Octubre: *"El sitio de Yahoo! en Japón ha sido atacado por* hackers *que han conseguido robar un archivo con 22 millones de nombres y datos sobre las cuentas personales"* (El País, 2013).

Dado que los SI la mayor parte del tiempo están instalados o son utilizados en un entorno de elevado flujo de información, difícil y vulnerable, pudiendo difundir además la información interna de las propias organizaciones; cobra importancia la seguridad que el propio sistema brinda a la empresa y que es sustentada por políticas y procesos, procedimientos de software y hasta el propio hardware.

Los SI deben garantizar la seguridad en los programas y bases de datos, orientados a mantener las tres cualidades propias de la información: disponibilidad, integridad y confidencialidad para que estos se usen únicamente para los propósitos que fueron creados dentro del marco previsto.

De manera general la seguridad del control de acceso de la información de los SI contribuye a la confidencialidad de modo que la información sólo sea accesible por las personas autorizadas y con los permisos adecuados, la integridad que contribuye a mantener la exactitud de la información almacenada y generada, la disponibilidad para garantizar que los clientes o los usuarios sólo puedan acceder a los recursos autorizados en el momento adecuado.

Los SI utilizan una serie de elementos esenciales para garantizar la seguridad de la gestión y el control de los usuarios y clientes, *"el acceso lógico al software de la aplicación y la información se debiera limitar a los usuarios autorizados"* (ISO, 2005), fundamental esto para *"controlar el acceso del usuario a la información y las funciones del sistema de aplicación, en concordancia con una política de control de acceso definida"* (ISO, 2005).

Según Microsoft al decir del control de acceso menciona que la ventaja fundamental es que este *"se concentra en garantizar el derecho a acceder a datos y recursos del sistema configurando los mecanismos de autentificación y control que aseguran que los usuarios de estos recursos sólo posean los derechos que se les han otorgado por lo que debe estudiarse*

*de modo que no evite que los usuarios desarrollen usos necesarios y así puedan utilizar los sistemas de información en forma segura"* (Microsoft, 2011).

Del control de acceso el Ministerio de Informática y las Comunicaciones de la República de Cuba define que *"es el proceso de restringir y auditar el acceso de los usuarios (persona, rol, sistema, entre otros) a los recursos (información, sistema, objetos, ficheros, entre otros) gestionados por SI, a través de la concepción integrada de los procesos de identificación, autenticación, autorización y auditoría"* (Resolución No. 127 /2007, 2007).

El control de acceso posee tres procesos fundamentales: Identificación y Autenticación, Autorización, Auditoría, según (Baryolo, 2012) estos se definen como se muestra:

- ***"Identificación:*** *es la acción por parte de un usuario de presentar su identidad a un sistema, generalmente se usa un identificador de usuario. Establece que el usuario es responsable de las acciones que lleve a cabo en el sistema".*
- ***"Autenticación:*** *es la verificación de que el usuario que intenta identificarse es válido, usualmente se implementa con una contraseña en el momento de iniciar una sesión".*
- ***"Autorización:*** *es un proceso para determinar si un sujeto identificado y autenticado tiene acceso al recurso solicitado. Da la posibilidad de ejecutar operaciones específicas, dependiendo de sus derechos de acceso pre-configurados. La política de autorización debe ser gestionada por el administrador o agente de seguridad responsable de apoyar y llevar a cabo la política de seguridad en la organización".*
- ***"Auditoría:*** *es el proceso de registro y análisis de todas las acciones ejecutadas por los sujetos sobre los recursos, a través de un SI. La auditoría es un aspecto crítico para identificar violaciones, debilidades, amenazas y predecir comportamientos y oportunidades de mejoras que apoyen la toma de decisiones en las organizaciones".*

*"La necesidad de control en el acceso a los sistemas es de vital importancia pues la concentración de muchas funciones antes dispersas se vuelve particularmente crítica si la información ingresada y procesada por un sistema de información por falta de efectivos procedimientos de control es intencional o inadvertidamente destruida o distorsionada"* (Sánchez, 2009).

Dado el hecho de que la seguridad de la información en las organizaciones es fundamental para la gestión de sus procesos de negocio, el control de acceso es la primera acción utilizada en aras de proteger la información de posibles modificaciones no autorizadas y garantizar la protección de los recursos del sistema.

El proceso de evaluación de la seguridad del control de acceso posee un conjunto de carencias que no satisfacen la obtención de resultados reales y fiables. El Centro Nacional de Calidad Software (CALISOFT) certifica productos de software desarrollados en la Universidad de las Ciencias Informáticas y en el resto del país. Encuestas realizadas a los especialistas de CALISOFT, arrojaron las siguientes conclusiones:

- El proceso de evaluación de la seguridad se basa en los requisitos no funcionales de seguridad de la entidad a la que pertenece el sistema a evaluar. Este proceso no considera los controles de seguridad de aplicación internacional reflejados en normas y estándares.
- Insuficientes controles para la evaluación de la seguridad del control de acceso lo cual provoca que no sean considerados la mayor cantidad de controles de seguridad de normas y estándares
- Los tiempos previstos para el proceso de evaluación aumentan en la medida en que el proceso se realiza manualmente. Además se puede incurrir en errores humanos durante los cálculos asociados a la evaluación
- No existe una estandarización con relación al algoritmo o conjunto de pasos a seguir durante el proceso de evaluación
- No están definidas el conjunto de métricas a utilizar.

Los escenarios anteriores traen consigo inconsistencias en el proceso de evaluación, reportes a los directivos poco confiables, inseguridad de la evaluación.

Evaluar la seguridad de un sistema contribuye a evitar fraudes efectuados, fortalecer las políticas, los objetivos, metodología, estándares, asignación de las tareas y adecuada administración de los recursos humanos y empresariales, disminuir el incumplimiento de los plazos acordados y la mala calidad de los resultados esperados.

Un método para evaluar la fiabilidad de la evaluación de la seguridad del control de acceso de los SI protege la información que estos gestionan y permite obtener una evaluación de los procesos para determinar cuan seguros están los procedimientos internos que dependen de la seguridad de su mecanismo de control de acceso.

Un adecuado mecanismo de control de acceso posibilita que las organizaciones sean capaces de administrar y mejorar los riesgos operacionales en los procesos de negocios, el acceso a los SI y recursos informáticos, así como optimizar los controles para mitigar los riesgos operacionales, aumenta la confiabilidad de la información financiera u organizacional que se deriva de la gestión de la propia entidad.

Considerando la problemática planteada se asume como **problema a resolver:** ¿Cómo evaluar los procesos de identificación y autenticación, autorización y auditoría de los Sistemas de Información para determinar la fiabilidad de la evaluación del control de acceso? Luego se obtiene como **objeto de estudio:** Proceso de desarrollo de software de gestión con el **objetivo:** Desarrollar un sistema informático que integre indicadores para determinar la fiabilidad de la evaluación de la seguridad del control de acceso de las evaluaciones de los procesos de Identificación y Autenticación, Autorización y Auditoría de los Sistemas de Información.

El **campo de acción** de la investigación se centra en la fiabilidad de la evaluación de la seguridad del control de acceso de las evaluaciones de los procesos de identificación y autenticación, autorización y auditoría de los Sistemas de Información**.**

Se sostiene la siguiente **idea a defender:** la realización del diseño e implementación de un sistema informático que permita evaluar los procesos de identificación y autenticación, autorización y auditoría de los Sistemas de Información permitirá determinar la fiabilidad de la evaluación de la seguridad del control de acceso.

Para dar solución al problema planteado se trazaron las siguientes **tareas de investigación:**

- Análisis de los procesos de Identificación y Autenticación, Autorización y Auditoría de los Sistemas de Información.
- Definición de los indicadores para la evaluación de los procesos de Identificación y Autenticación, Autorización y Auditoría de los Sistemas de Información.

- Diseño de las historias de usuario y las tarjetas clase-responsabilidad-colaboración y las historias de usuario de los procesos de Identificación y Autenticación, Autorización y Auditoría de los Sistemas de Información.
- Validación del diseño del sistema mediante la aplicación de métricas de software.
- Implementación de las funcionalidades del sistema para la evaluación de los procesos de Identificación y Autenticación, Autorización y Auditoría de los Sistemas de Información.
- Validación de los resultados obtenidos a través de pruebas al sistema, a fin de comprobar su correcto funcionamiento.

Es necesario que las organizaciones sean capaces, basándose en una serie de criterios, de evaluar, diagnosticar y calcular cuan seguros son los SI que existen en las empresas, propiciando que:

- Las organizaciones hacer un análisis basado en las normas y estándares de los sistemas.
- Posibilitando una mejora de los sistemas que la componen.
- Se reduzca o inclusive elimine la incidencia de amenazas digitales y los riesgos empresariales y estructurales que podrían existir.
- Se posibilite con toda la información obtenida redefinir o refinar sus procesos internos, conllevando esto a la mejora continua de la organización.

El *"objetivo de las organizaciones es disminuir los riesgos sin necesidad de realizar fuertes inversiones en software y sin contar con una gran estructura de personal. Para ello es necesario conocer y afrontar los riesgos a los que se somete la información, contemplar procedimientos adecuados y planificar e implantar controles de seguridad. Es decir, a partir de la aplicación de un sistema de gestión de la información en una organización, se comienza a trabajar de manera más sistemática y controlada sobre lo que sucede en los sistemas de información y sobre la propia información que se maneja"* (Seguridad de la Información, 2012). De esta manera serán capaces las organizaciones de contribuir en gran medida a un mayor incremento en la fidelidad de la toma de decisiones por parte de los directivos.

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se especifican los conceptos fundamentales relacionados con el dominio del problema planteado. Se realiza fundamentalmente un análisis detallado y crítico de las metodologías de desarrollo, lenguajes de modelado, herramientas de modelado, marcos de trabajo, entornos de desarrollo, sistemas para la gestión de las bases de datos. Con el objetivo de identificar las ventajas y desventajas, fortalezas y debilidades de las principales propuestas existentes, para seleccionar de ahí las que cumplen con las necesidades de la investigación.

## 1.1- Normas o estándares, recomendaciones, resoluciones, controles, decretos y métricas.

- **ISO/IEC 27001**: esta norma asegura que la gestión de la seguridad de la información debe realizarse mediante un proceso sistemático, documentado y conocido por toda la organización. *"Este proceso es el que constituye un Sistema de Gestión de la Seguridad de la Información (SGSI), que podría considerarse, por analogía con una norma tan conocida como la ISO 9001, como el sistema de calidad para la seguridad de la información"* (Alvarez Zurita, y otros, 2007).
- **ISO/IEC 27002**: "*establece los lineamientos y principios generales para iniciar, implementar, mantener y mejorar la gestión de la seguridad de la información en una organización"* (Standard and techniques, 2005).
- **ISO/IEC 27004**: "*La norma internacional ISO/IEC 27004 establece la necesidad de un programa de medidas o métricas. Para una valoración de proceso de auditoría, la evaluación de la calidad de auditoría es sistematizada en un Programa de Medida de la Calidad de la Auditoría"* (Popa, 2011).
- **ISO/IEC 15504**: también conocido como Software Process Improvement Capability Determination, abreviado SPICE, en español, «Determinación de la Capacidad de Mejora del Proceso de Software» es un modelo para la mejora y evaluación de los procesos de desarrollo y mantenimiento de sistemas de información y productos de software.

- **ISO/IEC 15408 Criterio Común (CC)** "*posibilita las evaluaciones de productos y sistemas de seguridad en un esfuerzo para definir una evaluación de la seguridad de TI. Constituye una guía útil para la gestión de productos y sistemas de TI con funciones de seguridad"* (Room, 2001).
- **Objetivos de Control para la Información y la Tecnología relacionada** (COBIT por sus siglas en inglés) "*constituye un conjunto de mejores prácticas que proporciona un marco de trabajo de dominios y procesos. Proporciona una estructura manejable y lógica"* (Criteria, 2012). *"Esta Norma tiene como misión: Investigar, desarrollar, publicar y promover un conjunto internacional y actualizado de objetivos de control para tecnología de información que sea de uso cotidiano para gerentes y auditores"* (Domínguez, 2004).
- **MAGERIT**: "*es la Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información de las Administraciones Públicas promovida por el Consejo Superior de Informática. MAGERIT define los procedimientos para guiar a la Administración paso a paso en el establecimiento de la protección necesaria y como respuesta a su dependencia creciente respecto de las técnicas electrónicas, informáticas y telemáticas"* (Esteban, 2004).
- **Biblioteca de Infraestructura de TI** (ITIL por sus siglas en inglés) *"permite cumplir con los atributos de disponibilidad, confidencialidad, integridad, autenticidad y no repudio de la información"* (Garzaro, 2007).
- **Publicación Especial 800-53 del NIST** "*propone un conjunto de controles de seguridad con el fin de que sean implementados en una organización para brindar protección a la información y a los SI de tipo federal"* (Federal Information Processing Standards Publication 200 (FIPS-200). Computer Security Division, 2006).
- **Proyecto de los 10 Mejores Elementos** *"integra parte del Proyecto de Seguridad para Aplicaciones Web de Código Abierto"* (OWASP por sus siglas en inglés), *"contribuye a lograr la seguridad de aplicaciones y gestionar el riesgo que las aplicaciones de software crean en su organización"* (OWASP, 2010) .
- **Las guías de OWASP** permiten verificar la seguridad que el sistema ejecuta. *"Son guías disponibles que mantienen el enfoque de un problema general, sin proporcionar*

*suficiente información para encontrar, diagnosticar y resolver los problemas de seguridad"* (Foundation, 2008).

- **Norma de Buenas Prácticas para la Seguridad de la Información** *"contiene una amplia gama de características que cubre todo el espectro de los arreglos que deben hacerse para mantener los riesgos de negocio asociados con los SI"* (ISO/IEC, 2007).
- **Resolución 127/2007** *"proporciona vigencia al "Reglamento de Seguridad para las Tecnologías de la Información". Constituye un Reglamento de seguridad para las TI que regula el sistema para la Seguridad y Protección de la Información Oficial"* (Comunicaciones, 2007).
- **Decreto Ley 281** *"constituye un conjunto de disposiciones legales que establecen los principios para la integración del Sistema de Información del Gobierno determinado por su organización y funcionamiento"* (Gaceta Oficial de la República de Cuba, 2011).
- **Métricas de Seguridad del Centro de Seguridad de Internet:** "*Las métricas de seguridad que define el Centro de Seguridad de Internet (CIS por sus siglas en inglés) están centradas en el esfuerzo involucrado en los procesos de seguridad, tales como el esfuerzo por corregir una vulnerabilidad que pueden ser utilizados para mejorar la eficiencia, métricas de todo el impacto y los beneficios de la organización"* (Security, 2009).

### 1.2- Soluciones existentes para la evaluación de la seguridad del control de acceso

*"El proceso de autorización se nutre de los datos que arroja como salida el proceso de identificación y autenticación para discernir los privilegios de cada usuario. Tiene la responsabilidad de establecer las políticas o privilegios de acceso de los usuarios sobre los recursos en los diferentes dominios"* (Baryolo, 2012).

- **Listas de Control de Acceso**

  Una Lista de Control de Acceso (ACL, por sus siglas en inglés) *"puede ser considerada como una estructura de datos jerárquica que puede contener múltiples ACL, cada una de ellas define un conjunto de permisos asociados a un determinado recurso. Los —sujetos, objetos y permisos‖ son los conceptos fundamentales que se utilizan para*

*definir las reglas de acceso aplicadas directamente a los usuarios o a los grupos donde pertenecen"* (Pinagapani, 2009).

- **Control de Acceso Discrecional**

  Control de Acceso Discrecional (DAC, por sus siglas en inglés) *"es una forma de autorización basada en los sujetos y grupos a los que pertenece un objeto. Se dice que es discrecional en el sentido de que un sujeto puede transmitir sus permisos a otro sujeto sin la aprobación de un administrador general"* (Downs).

- **Control de Acceso Obligatorio (MAC, siglas en inglés)**

  En el Control de Acceso Obligatorio (MAC, por sus siglas en inglés) *"todos los sujetos y objetos son clasificados basándose en niveles predefinidos de seguridad. Para establecer las políticas por niveles, los sujetos y objetos son marcados con etiquetas de seguridad, que siguen el modelo de clasificación de la información militar (desde desclasificado, hasta alto secreto)"* (Robles, 2008).

- **Control de Acceso Basado en Roles (MAC, siglas en inglés)**

  La definición básica de Control de Acceso Basado en Roles (MAC, por sus siglas en inglés) establece que *"los usuarios son asignados a roles, los permisos son asociados a roles y los usuarios adquieren permisos siendo miembros de roles. Las asignaciones usuario-rol y permiso-rol pueden ser muchos-a-muchos, por lo que un usuario puede pertenecer a muchos roles y un rol puede poseer muchos usuarios. De manera similar un permiso puede ser asociado a muchos roles y un rol puede tener asociado muchos permisos"* (Tao, 2010).

- **Sistema de Puntuación de Vulnerabilidades Comunes**

  *"El Sistema de Puntuación de Vulnerabilidades Comunes (CVSS por sus siglas en inglés) es un estándar abierto que se aplica universalmente a cualquier vulnerabilidad informática. CVSS es adoptado por organizaciones como Cisco, US National Institute of Standards and Technology (NIST), Qualys y Oracle*" (NIST, 2007).

- **Técnica Ajustada de Puntuación de Riesgos de Negocio**

  *"La Técnica Ajustada de Puntuación de Riesgos de Negocio Ajustada (BAR por sus siglas en inglés) clasifica defectos de seguridad de acuerdo a su tipo de vulnerabilidad, el grado de riesgo y posible impacto. BAR presenta un método de estimación rápido y*

*ligero que cuantifica el riesgo en términos de tiempo o dinero"* (Jaquith, 2002; Rostyslav Barabanov, 2011).

- **OSSTM**

  *"La Metodología de Prueba de Seguridad de Código Abierto (OSSTM por sus siglas en inglés) es un proyecto que está libre de la influencia comercial y política. Permite realizar un seguimiento de los objetivos y parte de los objetivos de prueba, la forma en que se prueban y los tipos de controles empleados"* (OSSTMM, 2010).

- **Puntuación Lógica de Preferencias**

  *"La Puntuación Lógica de Preferencias (LSP por sus siglas en inglés) es un método que permite evaluar y seleccionar sistemas complejos de hardware y software. La evaluación se realiza mediante operadores de Lógica Continua que propone la agregación de operadores lógicos Disyunción, Conjunción, Generalización (GCD por sus siglas en inglés) para construir un modelo de evaluación"* (Bayucan, 1997; Dujmovic, 1996).

### 1.3- Sistemas líderes en el mundo para la evaluación de la seguridad del control de acceso

- **Gestor de Entidades Tivoli de IBM**

  *"Este sistema proporciona una gestión de identidades basada en políticas en entornos heredados y de e-business. Incorpora un motor de flujo de trabajo y utiliza los datos de identidad para distintas actividades, como auditorías y generación de informes. Para gestionar el proceso de identificación y autenticación implementa estándares y protocolos como SAML, LDAP y OpenID. A través de ellos provee los mecanismos necesarios para su aplicación en entornos multidominios"* (Solis, 2008).

- **Suite de Gestión de Entidades y Acceso de Oracle**

  *"Oracle Identity and Access Management Suite es considerada por los especialistas en la temática, la solución más completa de control de acceso en la actualidad. Permite gestionar todo el ciclo de vida de las identidades de los usuarios en entornos de uno o varios dominios"* (Oracle, 2010).

### 1.4- Metodologías de desarrollo

Las metodologías de desarrollo de software son *"un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información"* (CMS, 2008), es decir, son las encargadas de definir quién hace qué, cuándo y cómo lo tiene que hacer.

Estas metodologías son utilizadas según las necesidades del proceso de desarrollo de software. Actualmente existen dos importantes tendencias, las pesadas que son utilizadas para desarrollar proyectos de gran envergadura y que además exigen una gran cantidad de tiempo, planificación y personal para el desarrollo y las ágiles que son utilizadas para proyectos menos formales, donde existen pocos desarrolladores y el tiempo es realmente una limitante.

#### 1.4.1- Proceso Unificado de Rational

La metodología de desarrollo Proceso Unificado de Rational (RUP) cuya versión más utilizada se creó en 1988 se ubica dentro de las de tipo pesado, son llamadas así debido a la gran cantidad de tiempo de desarrollo y al gran número de especialistas que se necesitan para crear el proyecto de software. Es utilizada fundamentalmente *"en el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto"* (Santiago Zaragoza, 2014).

El ciclo de vida de RUP se caracteriza por:

- Dirigido por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

#### 1.4.2- SCRUM

Fue desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Esta metodología define un marco para la gestión de los proyectos que ha sido utilizado con éxito durante los últimos años. *"Scrum está especialmente diseñada para admitir cualquier tipo de cambios en los requisitos, especialmente en un mercado de alta competitividad, aparte de priorizar el trabajo en dependencia del valor que tenga para el negocio. Los requisitos y las prioridades se*

*revisas y ajustan en intervalos de tiempos muy cortos y regulares, por lo permite adaptar el software a las necesidades del cliente en tiempo real. Otra característica fundamental es que el equipo de desarrollo, se dirige hacia un único objetivo: construir un producto de calidad, por lo que la gestión de proyecto es el encargado de definir las características del producto, así como de eliminar cualquier obstáculo que impida el cumplimiento del objetivo final del equipo"* (Figueroa, y otros, 2010).

### 1.4.3- Programación Extrema

La Programación Extrema (XP) es una metodología ágil de desarrollo de software ideada principalmente por Kent Beck en 1999. *"Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito de un proyecto. Su prioridad es que el sistema funcione correctamente y no la creación exhaustiva de documentos, los cuales deben ser cortos y estar centrados en la información esencial. El equipo de desarrollo sigue estrictamente el orden de prioridad de las tareas definidas por el cliente"* (Letelier, y otros, 2008). *"Se centra en un proceso de cuatro fases principales: Planificación, Diseño, Desarrollo y Prueba"* (Acosta, 2010). Sus principales características son:

- Diseñada principalmente para entornos dinámicos.
- Pensada para equipos pequeños (pocos programadores)
- Orientada hacia la codificación.
- Hace énfasis en la comunicación informal, verbal con el cliente.
- Utiliza poco tiempo de desarrollo.

Según las características de ambas metodologías, se elige XP por ser la que responde a las necesidades principales de tiempo, entorno, cantidad desarrolladores, y además de esto, incluye al cliente como parte fundamental del equipo de desarrollo.

## 1.5- Lenguajes de modelado

Los lenguajes de modelado son un conjunto estandarizado de símbolos y reglas, utilizados en la actualidad como una técnica para tratar con sistemas complejos o para simplificar el análisis del diseño de aplicaciones informáticas. La utilización de estos modelos permite a los ingenieros visualizar y representar el sistema que se va a construir de una manera sencilla y

fácilmente comprensible, por lo que pueden utilizarse como medio de comunicación con el cliente.

### 1.5.1- Notación de Modelado de Procesos de Negocio

La Notación de Modelado de Procesos de Negocio (BPMN) fue publicado en el año 2004 en su versión 1.0 por su desarrollador Business Process Management Initiative (BPMI), actualmente es mantenida por el OMG (Object Management Group).

El principal objetivo de este estándar es brindar *"la capacidad de entender sus procedimientos internos de negocio en una notación gráfica y dará a las organizaciones la capacidad de comunicar estos procedimientos de una manera estándar. Además, la notación gráfica facilita el entendimiento de la utilidad de las transacciones de colaboración y negocios entre las organizaciones"* (Object Management Group, 2014).

La finalidad esencial de la síntesis BPMN es servir como lenguaje unificador para cerrar la brecha existente entre el diseño de todos los procesos de negocios que realizan las empresas y su posterior implementación por parte de los desarrolladores.

BPMN fue desarrollada para dar soporte únicamente a los procesos de negocio empresariales, lo que significa que cualquier otro tipo de modelado que se pretende realizar con fines distintos a estos por ejemplo los modelos de datos no podrán llevarse a cabo.

### 1.5.2- Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML) es uno de los más conocidos y usados en la actualidad. Creado por Ivar Jacobson, James Rumbaugh y Grady Booch. Comienza a establecerse como un estándar en el año 1994 pero no es hasta 1997 que el OMG (Object Management Group) lo acepta. Posteriormente en el año 2001 se actualiza a su versión 2.0, la cual integra una gran cantidad de métodos propios de la Programación Orientado a Objetos (POO) con el objetivo de asegurar una amplia cobertura para el dominio de los sistemas.

*"Este lenguaje soporta un conjunto rico en elementos de notaciones gráficas. Describe la notación para clases, componentes, nodos, actividades, flujos de trabajo, casos de uso, objetos, estados y como modelar la relación entre todos estos elementos. UML soporta extensiones personalizadas a través de elementos estereotipados. "Con este lenguaje los ingenieros de software y las organizaciones pueden construir modelos rigurosos, trazables y*

*mantenibles, que soporten el ciclo de vida de desarrollo del software completo*" (Sparks, 2006).

Debido a la necesidad concreta que se tiene sobre los lenguajes de modelado se pretende utilizar para esta investigación el Lenguaje de Modelado Unificado (UML) ya que es útil en una variedad de problemas de ingeniería, permite el modelado funciones del sistema, se integra correctamente con la Programación Orientada a Objetos y representa eficientemente las bases de datos que se necesitan, además que posee componentes reutilizables, por lo que puede describir métodos o procesos y permite detallar profundamente los artefactos del sistema y documentar y construir sus principales componentes, por lo que es capaz de cumplir con las necesidades de modelado existentes para la creación de la solución informática.

### 1.6- Herramientas de Ingeniería de Software Asistida por Computadora

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE por sus siglas en inglés) son aplicaciones cuyo destino fundamental es aumentar la productividad en el desarrollo de software reduciendo la cantidad de tiempo y dinero utilizadas en su creación.

*"El objetivo primordial de las herramientas de esta categoría consiste en representar objetos de datos de negocios, sus relaciones, y ayuda a comprender mejor la forma en que fluyen estos objetos de datos entre distintas zonas de negocio en el seno de la compañía"* (Asensio, 2014).

Estas herramientas se pueden utilizar en cualquier aspecto del ciclo de vida del desarrollo de un software y es útil para realizar el cálculo de costos, generar código a partir de diagramas UML, realizar un análisis y diseño del proyecto, detección de errores, integración completa con herramientas de desarrollo y lenguajes de programación etc.

#### 1.6.1- Rational Rose

*"La herramienta CASE Rational Rose 7.0 está diseñada para la modelación visual mediante UML para el análisis y diseño de sistemas Orientados a Objetos. A través de ella se puede generar código para Java, C++, Ada, Visual Basic, CORBA y Oracle. Se utiliza para modelar los sistemas antes de producirlos y abarca todo el ciclo de vida del mismo"* (Mestras Pavón, 2007).

*"Es una solución de desarrollo controlado por modelo (MDD) para desarrollar sistemas complejos. Admite, de acuerdo con el lenguaje UML (Unified Modeling Language), construcciones basadas en modelo para la automatización del desarrollo, incluida la ejecución de modelos y la generación de código totalmente ejecutable"* (IBM, 2013).

### 1.6.2- Visual Paradigm

*"Visual Paradigm for UML 8.0 (VP) es una herramienta que soporta el ciclo de vida completo en el desarrollo de software: análisis y desarrollos orientados a objetos, construcción, prueba y despliegue. Permite diseñar diagrama de clases, código inverso, generación de código a partir de diagramas y generar documentación"* (Visual Paradigm For UML, 2012). Entre su principales ventajas se tiene: *"Multiplataforma: Soportada en plataforma Java para Sistemas Operativos Windows, Linux, Mac OS"*; *"Ingeniería de Código: Permite la generación de código e ingeniería inversa para los lenguajes: Java, C, C++, PHP, XML, Python, C#, VB .Net, Flash, ActionScript, Delphi y Perl"*; *"Integración con Entornos de Desarrollo: Apoyo al ciclo de vida completo de desarrollo de software en IDE como: Eclipse, Microsoft Visual Studio, NetBeans, Sun ONE, Oracle JDeveloper, Jbuilder y otros"*; *"Modelado de Bases de Datos: Generación de bases de datos y conversiones de diagramas entidad-relación a tablas de bases de datos, además de mapeos de objetos y relaciones"* (Dirección de Información, 2012).

De acuerdo al análisis antes realizado y a las características previamente mencionadas se escoge para el modelado de la aplicación informática el uso de la herramienta CASE Visual Paradigm 8.0 ya que posee varias ventajas que se ajustan a las necesidades existentes para la creación de la aplicación informática que se propone, entre estas se destacan: es posible utilizarla en múltiples plataformas, brinda soporte para lenguajes de programación Java, permite el modelado de bases de datos y su posterior exportación a código SQL, es capaz de generar toda la documentación necesaria. Además se puede emplear en cualquier fase del desarrollo del software, por lo que contribuirá una mayor flexibilidad y agilidad para la adaptación al cambio.

### 1.7- Lenguajes de programación

*"Un lenguaje de programación es un conjunto de reglas que especifican cuales secuencias de símbolos constituyen un programa, y lo que la ejecución del programa describe"* (Ben-Ari, 1996). También el propio autor define: *"un lenguaje de programación es un mecanismo de abstracción. Posibilita al programador especificar un cálculo abstracto, y dejar implementar al programa las especificaciones de forma detallada para la ejecución en una computadora*" (Ben-Ari, 1996).

#### 1.7.1- C++

C++ es un lenguaje de programación multiparadigma diseñado a mediados de los años 80 del siglo pasado por Bjarne Stroustrup con la intención de extender el uso del lenguaje C. *"Soporta varios estilos de programación, por ejemplo: procedural, orientado a objetos; intenta ser tan eficiente y portable como lo es el lenguaje C. Además, permite la sobrecarga de los operadores, la inclusión de directivas del preprocesador y los tipos genéricos, entre otras funcionalidades. Soporta plenamente el polimorfismo y la herencia, destacándose la existencia de la herencia múltiple"* (Pressman, 2002).

#### 1.7.2- C Sharp

C Sharp (C#) es un lenguaje de programación orientado a objetos, creado en 1999 por Microsoft como parte de su plataforma .NET. Su base fundamental son los lenguajes C y C++. Este lenguaje se diseñó para ser robusto y a la vez sencillo, su uso está extendido y sus principales aplicaciones sobresalen en los entornos de escritorio y web. Entre sus principales características y ventajas destacan que es: completamente orientado a objetos lo que facilita el uso con este tipo de paradigma de programación, soporta la encapsulación, herencia y polimorfismo, incluye mecanismos de control de acceso, gestión dinámica de la memoria del sistema para evitar el desbordamiento de la misma, posee normas de sintaxis que evitan la conversión entre tipos de datos que no son compatibles y un alto grado de manejo de excepciones.

#### 1.7.3- Java

Java, desarrollado por Sun Microsystems a principios de los años 90 es un lenguaje de programación orientado a objetos que toma gran parte de su sintaxis de los lenguajes más

populares de esa época C y C++. Sumado a esto se encuentran sus principales características según destaca (Fernández, 2010):

- ***Robusto**: la robustez es la medida de la fiabilidad de un programa y Java contiene varias funciones integradas que la garantizan, por ejemplo es un lenguaje basado en tipos, no tiene punteros, realiza automáticamente la recolección de elementos no utilizados y fomenta el uso de interfaces.*
- ***Seguro**: la seguridad está ligada a la robustez, por ejemplo, al no tener punteros no se corrompe la memoria. Además tiene integradas otras funciones de seguridad como son: el verificador de código de bytes que proporciona el primer nivel de defensas, a continuación el cargador de clases y luego el administrador de seguridad, que refuerza las normas de seguridad para el entorno de ejecución.*
- ***Arquitectura neutral**: los programas de Java se realizan para que se ejecuten en cualquier equipo sin recompilarlos.*
- ***Dinámico**: las bibliotecas de Java se encuentran evolucionando constantemente, sin embargo eso no implica que los programas anteriores dejen de funcionar. Otro elemento que explica el dinamismo es la preferencia de Java de las interfaces sobre las clases.*
- ***Interpretado**: los programas de Java son interpretados. En lugar de ser compilado en ejecutables nativos, el código de Java es traducido en códigos de bytes no asociados a una plataforma, por lo que no son dependientes a ningún entorno de ejecución o sistema operativo.*
- ***Orientado a objetos**: Un buen diseño conlleva a ser reutilizable, extensible y sostenible. Java cuenta con bibliotecas de clases previamente diseñadas que se van enriqueciendo de una a otra versión y constituyen la base sobre la cual el programador trabaja.*

Para el desarrollo de la investigación y la implementación de la aplicación se ha decidido utilizar el lenguaje de programación Java ya que este posee una sintaxis correctamente estructurada, alto grado de robustez. Además de un entorno de ejecución reducido y

altamente optimizado, utilizado en aplicaciones de escritorio, por lo que posee una gran portabilidad.

### 1.8- Marcos de trabajo para Java

#### 1.8.1- Swing

*"Swing es una biblioteca gráfica para Java. Incluye widgets para interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas. Arquitectónicamente es un framework MVC para desarrollar interfaces gráficas para Java con independencia de la plataforma"* (Llorente, 2011).

Completamente integrado con la Máquina Virtual de Java desde su versión 6.0, Swing es capaz de ejecutarse correctamente en cualquier sistema operativo gracias a que utiliza componentes ligeros y reutilizables que no dependen del entorno de desarrollo de los sistemas, sino que utiliza los propios.

Es una biblioteca altamente extensible con una arquitectura particionada, lo que significa que los usuarios pueden crear sus propias implementaciones de los componentes existentes, creando alternativas para diferentes desarrollos. Todo esto contribuye a que el desarrollo de sus componentes sea más activo y permite representar varios estilos apariencia en dependencia del sistema operativo en el que se ejecute.

#### 1.8.2- API de Persistencia de Java

La API de Persistencia de Java (JPA, por sus siglas en inglés) está fundamentalmente desarrollada para la plataforma Java EE, por lo que es utilizada principalmente para manejar información de bases de datos relacionales en aplicaciones de escritorio. Sus principales ventajas son que agilizan el desarrollo orientado a objetos, implementa funciones seguras contra posibles ataques, permite la persistencia de datos de una manera ágil y rápida, es compatible con buenas prácticas de diseño y permite un enfoque orientado a objetos al interactuar con una base de datos, siguiendo un patrón de mapeo de objetos.

### 1.9- Entornos de Desarrollo Integrado

Las herramientas para el desarrollo son programas informáticos integrados por un conjunto de utilidades que posibilitan la creación de aplicaciones informáticas para equipos de cómputo,

servidores, dispositivos móviles, etc. Son capaces de gestionar varios lenguajes de programación. Generalmente están compuestos por un editor del código, un compilador, un depurador y un constructor de interfaz gráfica usuario. Amigables para el usuario, son capaces de funcionar en múltiples plataformas.

### 1.9.1- Visual Studio

Visual Studio 2010 es un Entorno de Desarrollo Integrado (IDE) desarrollado específicamente para el sistema operativo Windows, fue lanzada por primera vez en el año 1998. Aunque actualmente se han desarrollado numerosas extensiones para utilizar un sinnúmero de lenguajes, los más utilizados son Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET. Dentro de sus características se pueden encontrar: licencia propietaria y con soporte exclusivo de Microsoft, posibilita la creación de aplicaciones para estaciones de trabajo, provee un conjunto de herramientas integradas que permiten un flujo de trabajo completo, capaz de crear entornos colaborativos entre los probadores y los desarrolladores, interfaz gráfica altamente personalizable, además reduce significativamente la complejidad de acceso a bases de datos.

### 1.9.2- NetBeans

NetBeans 7.3 es un Entorno de Desarrollo Integrado (del inglés IDE) *"de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores la creación rápida de diferentes tipos de aplicaciones utilizando la plataforma Java"* (Netbeans, 2014). Esta herramienta posee considerable éxito entre los desarrolladores ya que tiene una gran comunidad en constante crecimiento. *"Permite escribir, compilar, hacer un debug, ensamblar y desplegar aplicaciones, y aunque está escrito en Java, brinda soporte para toda clase de lenguajes de programación"* (Netbeans, 2014). Entre sus principales características se encuentran: libre y multiplataforma, capaz de gestionar aplicaciones de escritorio, soporte para sistemas de control de versiones, completa compatibilidad con sistemas Windows y Linux, soporte nativo para la gestión de bases de datos, integración total con herramientas CASE como Visual Paradigm, completamente enfocado para trabajar con herramientas para la creación de interfaces gráficas de usuario como Swing.

Para el desarrollo de la aplicación informática se utilizará el IDE NetBeans 7.3. Se eligió por ser libre, multiplataforma, principalmente creado y con soporte completo y total para el lenguaje de programación Java, altamente integrado con herramientas para la creación de interfaces gráficas de usuario Swing, sencillo y amigable, completa vinculación con herramientas para el control de versiones y modelado UML, además de que se cuenta con experiencia en el uso de la aplicación.

### 1.10- Sistemas de Gestión de Bases de Datos

Los Sistemas de Gestión de Bases de Datos (Database Management System, DBMS en inglés) *"son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan"* (Alvear Rodriguez, y otros, 2005). *"El objetivo primordial de un SGBD es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la bases de datos"* (Leivas, 2009). Además son capaces de abstraer al usuario de detalles como el almacenamiento físico de los datos, permiten la posibilidad de gestionar la seguridad, brindan amplias posibilidades de organización de los datos y disminuyen enormemente el tiempo empleado en el desarrollo de las aplicaciones.

#### 1.10.1- MySQL

MySql 5.6.4 *"opera a través de un intérprete de comandos. Los comandos de MySQL siguen el estándar de SQL (Structured Query Language), que es un lenguaje normalizado para operar con Bases de Datos Relacionales. Es desarrollado sobre la filosofía de software libre. Fue implementado por la empresa MySQL AB, que también le brinda soporte, pero puede utilizarse gratuitamente y su código fuente está disponible para los usuarios"* (Oracle, 2010). De sus características se destacan: se puede obtener bajo licencia GNU si el proyecto es compatible con esta licencia de lo contrario hay que comprarlo, está desarrollado en su mayoría en ANSI C, para agilizar las búsquedas utilizan las tablas organizadas en b-trees, optimizando así dichas búsquedas, ofrece seguridad utilizando para esto un sistema de contraseñas y privilegios seguros, permite el uso de memoria compartida entre clientes y servidores.

### 1.10.2- PostgreSQL 9.1

PostgreSQL 9.1 es un sistema para gestionar bases de datos relacionales, orientado a objetos y publicado bajo licencia BSD, su inicio se remonta al año 1982 en el proyecto "Ingres" en la Universidad de Berkeley liderado por Michael Stonebraker. *"Dada su propia arquitectura libre, posee una comunidad de desarrolladores que se ocupa de su implementación y evolución de manera desinteresada"* (PostgreSQL Core Team, 2013). Entre sus principales características se encuentran: es un sistema objeto-relacional ya que incluye herencia, tipos de datos, funciones. Posibilita la gestión de múltiples procesos sobre las tablas, es decir, un usuario puede estar escribiendo y el otro puede leer sin problemas de bloqueo, es capaz de utilizar tipos de datos creados por los propios usuarios implementa disparadores (Triggers del inglés) que a su vez posibilitan la creación de vistas, herencia de tablas e integridad transaccional, es un sistema multiusuario y multiplataforma, soporta el uso de lenguajes como C, C++, Java, Perl, PHP, Python, Ruby, etc.

Según el estudio anteriormente realizado y las necesidades de la investigación se selecciona a PostgreSQL 9.1 debido a que *"es un sistema de base de datos de código abierto muy potente. Soporta gran parte del estándar SQL, y en algunos aspectos, está diseñado para que sea extensible por los usuarios. Se caracteriza por posibilitar transacciones Atomicidad, Consistencia, Aislamiento y Durabilidad (ACID), claves foráneas, vistas, secuencias, sub-peticiones, disparadores, tipos y funciones definidos por el usuario, reunión externa, control de concurrencia multiversión"* (Internet Engineering Task Force, 2013). Debido a lo anteriormente planteado, PostgreSQL 9.1 posee las soluciones a las necesidades existentes para la gestión de la Base de Datos de la aplicación a desarrollar, es un sistema libre, multiplataforma, además se cuenta con experiencia en su utilización.

## 1.11- Métrica de Desarrollo de Software

*"Mediciones para el software que se pueden aplicar al proceso de desarrollo con el intento de mejorarlo sobre una base continua. Se utilizan para la estimación, el control de la calidad, la evaluación de productividad y el control de proyectos. Ayuda a evaluar la calidad de los resultados de trabajos técnicos y en la toma de decisiones tácticas a medida que el proyecto evoluciona"* (Ortiz, 2007).

### 1.11.1- Tamaño Operacional de Clase

El Tamaño Operacional de Clase (TOC) *"expresa el número de métodos asignados a una clase. El impacto que tiene su aplicación en los atributos de calidad mencionados se evidencia con mayormente en responsabilidad, complejidad y reutilización. La relación está expresada porque el aumento de la cantidad de métodos de la clase es directamente proporcional a la responsabilidad y complejidad de la clase, lo que supone un aumento de los mencionados indicadores. De igual manera, se traduce en una reducción en la reutilización que pueda tener la misma"* (Cid Escalona, y otros, 2009).

A continuación se muestran las clasificaciones de TOC que pueden tener las clases al obtener un valor de umbral determinado. La variable *N* será la cantidad de procedimientos de acuerdo a la clase analizada y la variable *Promedio* será el valor del promedio de la cantidad de procedimientos del total de clases.

Parámetros establecidos para el análisis:

1. **Responsabilidad:** Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.
    - Baja es cuando: N <= Promedio
    - Media es cuando: Promedio > N <= 2 * Promedio
    - Alta es cuando: N > 2 * Promedio
2. **Complejidad de implementación:** Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
    - Baja es cuando: N > 2 * Promedio
    - Media es cuando: Promedio > N <= 2 * Promedio
    - Alta es cuando: N <= Promedio
3. **Reutilización:** Aumento del TOC provoca disminución del grado de reutilización de la clase debido a que la clase posee un alto grado de especialización de sus componentes.
    - Baja es cuando: N <= Promedio
    - Media es cuando: Promedio > N <= 2 * Promedio
    - Alta es cuando: N > 2 * Promedio

### 1.12- Pruebas

La creación de software conlleva a que existan fallos humanos en su implementación, característica esta inherente al propio desarrollo. Estos fallos pueden existir u ocurrir desde el mismo comienzo de la implementación. Por lo que es necesario definir una serie de pruebas que sean capaces de medir, comprobar o evaluar el estado "constructivo" del sistema creado, debido a que esto es un elemento crítico para categorizar su calidad.

*"Uno de los pilares de la Programación Extrema es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones dividiendo dichas pruebas en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final"* (Noble, y otros, 2004).

#### 1.12.1- Pruebas Unitarias

*"Prueba unitaria es la más micro escala de las pruebas; para probar funciones particulares o módulos de código. Típicamente hechas por desarrolladores y no por probadores"* (Duharte, 2008).

*"Las pruebas unitarias constituyen un proceso para validar que una porción del código del sistema funciona apropiadamente de manera aislada. Las pruebas unitarias proveen la capacidad para probar un módulo del software de forma reproducible, eficiente y automatizada. Las pruebas unitarias fueron definidas como el nivel más fino de granularidad pues verifican una simple unidad de funcionalidad y deben probar que cada método de una clase satisface su contrato documentado"* (Duharte, 2008).

##### 1.12.1.1- Prueba del Camino Básico

*"El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del*

*conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa"* (Pressman, 2002).

**1.12.1.2- Complejidad Ciclomática**

*"La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez"* (Pressman, 2002).

La complejidad ciclomática puede calcularse de tres maneras diferentes:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
2. La complejidad ciclomática, V (G), de un grafo de flujo G se define como V (G)=A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
3. La complejidad ciclomática, V (G), de un grafo de flujo G también se define como V (G) P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.

**1.12.2- Pruebas de Aceptación**

*"Las pruebas de aceptación son diseñadas a partir de cada iteración y basadas en las Historias de Usuarios, en ella es el cliente el encargado de comprobar que ha sido correctamente implementada. Estas pruebas son reconocidas por su similitud con las llamadas "Pruebas de caja negra", pues como los clientes son los principales responsables de la inspección, también son los autorizados de diseñar la prioridad o el orden de resolución en caso de que falle alguna funcionalidad"* (Escribano, 2002).

Para la validación de la solución informática propuesta, se realizarán todas las pruebas y métricas anteriormente expuestas, dado que son un mecanismo crítico para la garantía de la calidad del software y figuran una revisión final, de los requisitos, del diseño y de la codificación.

**Conclusiones del capítulo**

- La información es un activo importante sobre en el desarrollo de los procesos empresariales.
- Las organizaciones tienen la necesidad de asegurar su información sensible del acceso no autorizado de terceros utilizando para esto los procesos de Identificación y Autenticación, Autorización y Auditoría.
- La evaluación de la fiabilidad de la evaluación de la seguridad del control de acceso es fundamental en la garantía del correcto funcionamiento organizacional.
- las tecnologías seleccionados para el desarrollo del sistema fueron: XP como metodología de desarrollo, UML para el modelado, VisualParadigm como herramienta CASE, Java como lenguaje de programación, NetBeans para el desarrollo de la aplicación y PostgreSQL para la persistencia de los datos como gestor de bases de datos.

# CAPÍTULO 2: DESARROLLO DE LA SOLUCIÓN INFORMÁTICA

En el presente capítulo se presenta el diseño e implementación del Sistema Informático para la Evaluación de la seguridad del Control de Acceso (SIECA). Muestra las etapas de diseño de la solución, presentando los artefactos generados por la metodología seleccionada. Conjuntamente con los requisitos funcionales y no funcionales presentes en la aplicación, la arquitectura en la que se basa la construcción del sistema y los Patrones Generales de Software para Asignación de Responsabilidades (GRASP por sus siglas en inglés) que se evidencian en su implementación. Presentación y explicación de las Historias de Usuario (HU) y las Tarjetas Clase, Responsabilidad y Colaboración (CRC) de los procesos que intervienen en la ejecución del sistema.

## 2.1- Breve introducción de SIECA

El desarrollo del Sistema Informático para la Evaluación de la seguridad del Control de Acceso (SIECA) permite hacer un análisis de los SI. SIECA contribuye a mejorar la administración de los procesos de la empresa, eliminar la incidencia de amenazas a la seguridad de la información, riesgos empresariales y estructurales. Garantiza la fiabilidad de la evaluación de la seguridad del control de acceso en los SI sobre la base del uso de tecnologías libres, multiplataforma, con alto grado de seguridad y adaptabilidad.

## 2.2- Indicadores y Criterios utilizados en el desarrollo

Para el desarrollo de SIECA fue necesario realizar un estudio de las principales normas, estándares, resoluciones y controles[1] que a nivel mundial dirigen el control de acceso, dicho estudió arrojó una serie de 10 indicadores comunes entre ellos que a su vez fueron derivados en 24 criterios de evaluación, en cuya utilización se centra la aplicación de SIECA

### 2.2.1- Indicadores

- I1. Empleo de estándares para el intercambio de información.
- I2. Empleo de métodos criptográficos y mecanismos seguros de comunicación (envío, recepción y almacenamiento de información sensible).

[1] Véase el epígrafe 1.1- Normas o estándares, recomendaciones, resoluciones, controles, decretos y métricas.

- I3. Empleo de soluciones para la federación de identidades entre dominios.
- I4. Empleo de pruebas desafío-respuesta en los eventos donde se necesite determinar cuando el usuario es una persona o no.
- I5. Empleo de estándares para la representación de la información de autorización.
- I6. Empleo de protocolos para la gestión de autorización de la red.
- I7. Empleo de estándares de control de acceso de la red.
- I8. Empleo de controles internos propuestos por marcos de trabajo de aplicación internacional.
- I9. Empleo de modelos para la gestión de la trazabilidad.
- I10. Empleo de estándares para la gestión de trazas.

### 2.2.1- Criterios

- C1. Implementación del estándar Kerberos.
- C2. Implementación del estándar OpenPGP.
- C3. Implementación del estándar EDIFACT.
- C4. Implementación del estándar AC X.509 (PKI X.509, S/MIME, IPSec, TLS, WAP, SOAP, WSDL, UDDI, WS Security, X.500, LDAP, XACML, XKMS).
- C5. Implementación de algoritmos de cifrado simétrico (DES-TDES, RC (2/4/5/6), AES, IDEA, SEAL, Blowfish, Serpent).
- C6. Implementación de algoritmos de cifrado asimétrico (Diffie-Hellman, RSA, DSA, ElGamal, Rabin).
- C7. Implementación del estándar Passport.
- C8. Implementación del estándar Shibboleth.
- C9. Implementación del estándar Liberty Alliance.
- C10. Implementación del estándar PAPI.
- C11. Implementación de la prueba CAPTCHA.
- C12. Implementación del estándar SPKI/SDSI.
- C13. Implementación del estándar SAML.
- C14. Implementación del estándar TACACS+.

- C15. Implementación del estándar RADIUS.
- C16. Implementación del estándar DIAMETER.
- C17. Implementación del estándar WEP.
- C18. Implementación del estándar WPA.
- C19. Implementación del estándar 802.1X.
- C20. Implementación del estándar PANA.
- C21. Implementación del estándar EAP.
- C22. Implementación de marcos de trabajo COSO.
- C23. Implementación de modelos para la gestión de trazas MDE.
- C24. Implementación de estándares para la gestión de trazas POSIX 1003.1q.

### 2.3- Algoritmo Lógica de Puntuación de Preferencias (LSP) adaptado

La evaluación del sistema SIECA es cuantitativa y está basada en las evaluaciones de los procesos de Identificación y Autenticación, Autorización y Auditoría que agrupan las evaluaciones de los indicadores y estos a su vez las de los criterios de evaluación. A continuación se presenta la adaptación del algoritmo LSP para la evaluación de indicadores y procesos.

Sobre la base del funcionamiento del algoritmo LSP y sus conceptos fundamentales (*Variables de Performance, Preferencias Elementales y Preferencia Global*) se describe la interrelación de los estos elementos en el proceso de evaluación de la seguridad del control de acceso de un SI.

La evaluación de un SI es cuantitativa, se determina a partir de la evaluación de los criterios de evaluación y está basada en los conceptos de *Variables de Performance, Preferencias Elementales y Preferencia Global*. Las *Variables de Performance* son los criterios de evaluación con los que cumple el sistema que se evalúa. Las *Preferencias Elementales* son las evaluaciones de los criterios del indicador y se obtienen mediante funciones que determinan la parte que representa la cantidad de criterios que se cumplen en el sistema que se evalúa de la sumatoria de las cantidades de todos los criterios que deben ser cumplidos por el sistema (24 criterios). La *Preferencia Global* es la evaluación

del proceso y se obtiene mediante la sumatoria de las *Preferencias Elementales* de los criterios que lo forman. En la figura se muestra el algoritmo LSP adaptado para la evaluación de un sistema.

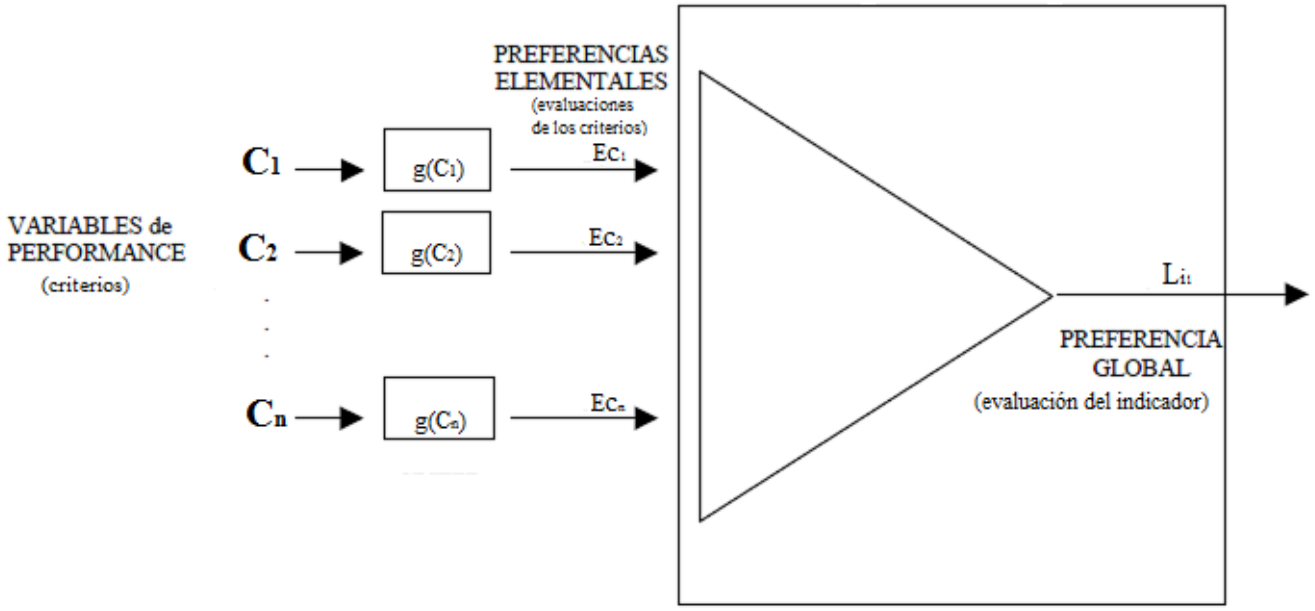



**Figura 1**: Estructura del algoritmo LSP adaptado.

Las funciones $g(C_n)$ determinan la parte que representa la cantidad de criterios que se cumplen en el sistema que se evalúa de la sumatoria de las cantidades de todos los criterios que deben ser cumplidos por el sistema (24 criterios). Las preferencias $Ec_n$ son las evaluaciones de los criterios de evaluación y $L_i$ es la preferencia global o evaluación del sistema.

### 2.4- Requisitos

*"Un requisito puede definirse como un elemento necesario dentro de un sistema, que puede representar una condición o capacidad, para satisfacer un contrato, estándar, especificación u otro documento formal. Esta actividad debe hacerse con gran precisión, por el papel primordial que juega en la producción de software, enfocada en la definición de lo que se desea producir,*

*mediante una descripción más clara del comportamiento del sistema"* (Martínez Díaz, y otros, 2010).

Los requisitos son divididos en dos categorías: Funcionales y No Funcionales, en general, los requisitos funcionales definen lo que el sistema debería de hacer, por otra parte los no funcionales verifican cómo un sistema debería ser.

*"Una buena definición de requisitos no funcionales es tan importante como los requisitos funcionales; estos deben de ser apropiadamente definidos, analizados y trazados"* (Martínez Díaz, y otros, 2010).

**2.4.1- Requisitos Funcionales**

Los requisitos funcionales al decir de Synergix son la "c*aracterística requerida del sistema que expresa una capacidad de acción del mismo – una funcionalidad; generalmente expresada en una declaración en forma verbal"* (Synergix, 2013).

*"Los requisitos funcionales no alteran la funcionalidad del producto, esto quiere decir que los requisitos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen"* (Martínez Díaz, y otros, 2010).

Los requisitos funcionales describen lo que el sistema debe hacer, son todas las condiciones y capacidades que debe cumplir el software, o producto en general, para que las peticiones del cliente queden satisfechas.

Los requisitos funcionales de SIECA son:

- Registrar indicadores.
- Registrar criterios de evaluación.
- Registrar sistema informático a evaluar.
- Mostrar los indicadores para la evaluación de sistemas.
- Listar los criterios para la evaluación asociados a cada indicador.
- Guardar el registro de criterios.
- Guardar el registro de indicadores.
- Guardar el registro de sistemas
- Evaluar un sistema.
- Guardar el resultado de la evaluación del sistema
- Mostrar el reporte de la evaluación del mismo.

### 2.4.2- Requisitos No Funcionales

"C*aracterística requerida del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo, que señala una restricción del mismo"* (Synergix, 2013).
Según las definiciones del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) los requisitos no funcionales se dividen en las siguientes categorías:

- **Requisitos de Software:** debe mencionarse el software del que debe disponer el sistema.

  SIECA necesita para su correcto funcionamiento:

  1. Máquina Virtual de Java versión 7 o posterior para el sistema operativo que utilice.
  2. PostgreSQL 9.1 o posterior.
  3. Un tiempo de entrenamiento de un máximo de 7 días para usuarios del sistema.

- **Requisitos de Hardware:** enunciar los elementos de hardware con los que se disponen.

  SIECA para su ejecución necesita una computadora personal (PC) con como mínimo:

  1. Procesador: 1.5 GHz.
  2. RAM: 512 Mb.
  3. Disco Duro: 40 Gb.

- **Requisitos diseño y la implementación:** especifica o restringe la codificación o construcción de un sistema; son restricciones que han sido ordenadas y deben ser cumplidas estrictamente. El sistema SIECA debe ser capaz de ser ejecutado equipos sin importar el sistema operativo instalado, además con los patrones de diseño más utilizados a nivel mundial.

- **Requisitos de apariencia o interfaz externa:** describe la apariencia del producto. Es importante destacar que no se trata del diseño de la interfaz en detalle sino que especifican cómo se pretende que sea la interfaz externa del producto.

SIECA debe ser de fácil uso con una interfaz amigable para sus usuarios, de modo que no cause ninguna molestia al cliente. Estará organizada de forma tal que no será ninguna dificultad para cualquier tipo de usuario utilizar fácilmente la aplicación.

- **Requisitos de Seguridad:** La seguridad puede ser tratada en tres aspectos diferentes en el sistema SIECA:
  - Confidencialidad: la información manejada por el sistema está protegida de acceso no autorizado y divulgación ya que cuenta con una herramienta de validación de usuarios utilizando nombres de usuarios vinculados a contraseñas.
  - Integridad: la información operada por el sistema será centro de minuciosa protección contra la posible corrupción de los datos.
  - Disponibilidad: los usuarios autorizados se les certificará el acceso a la información y que los mecanismos utilizados para alcanzar la seguridad no ocultarán o retardarán a los usuarios.

### 2.5- Historias de Usuario

"Las Historias de Usuario (HU) *"son utilizadas en la metodología de desarrollo ágil XP para representar una breve descripción del comportamiento del sistema. Emplea terminología del cliente sin lenguaje técnico, se realiza una por cada funcionalidad del sistema. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo"* (Jiménez, 2011).

Como síntesis según las palabras de Pressman *"las historias de usuario describen las necesidades requeridas para el software que se construirá"* (Pressman, 2002).

Las historias de usuario (HU) que se identificaron durante la etapa de análisis de la solución informática son: Gestionar Criterio, Gestionar Proceso, Gestionar Sistema y Evaluar Sistema.

**Tabla 1:** HU Gestionar Criterio.

| HU "Gestionar Criterio" | |
|---|---|
| **Número:** 1 | **Nombre HU**: Gestionar Criterio |
| **Modificación de Historia de Usuario Número:** | |
| **Referencia:** No | |
| **Programador:** Wilfredo Palma Pérez | **Iteración Asignada**: 1 |
| **Prioridad en Negocio:** Alta **(Alta/Media/Baja)** | **Puntos Estimados(días)**: 15 |
| **Riesgo en Desarrollo:** Alto | **Puntos Reales(días)**: 24 |
| **Descripción:** La aplicación permite Registrar (interfaz para adicionar), Modificar (interfaz para modificar) y Eliminar (interfaz para modificar) un indicador utilizando para esto el código del Indicador en cuestión y los parámetros que en dependencia de la de la acción a realizar se muestren en la aplicación. | |
| **Observaciones:** | |

**Tabla 2:** HU Gestionar Proceso.

| HU "Gestionar Proceso" | |
|---|---|
| **Número:** 2 | **Nombre HU**: Gestionar Proceso |
| **Modificación de Historia de Usuario Número:** | |
| **Referencia:** No | |
| **Programador:** Wilfredo Palma Pérez | **Iteración Asignada**: 1 |
| **Prioridad en Negocio:** Alta **(Alta/Media/Baja)** | **Puntos Estimados(días)**: 12 |
| **Riesgo en Desarrollo:** Alta | **Puntos Reales(días)**: 18 |
| **Descripción:** La aplicación permite Registrar (interfaz Adicionar), Modificar (interfaz Modificar) y Eliminar (interfaz Modificar) un proceso utilizando para esto los criterios que en dependencia de la situación que se presente se muestren en la aplicación. | |
| **Observaciones:** | |

**Tabla 3:** HU Gestionar Sistema.

| HU "Gestionar Sistema" | |
|---|---|
| **Número: 3** | **Nombre HU**: Gestionar Sistema |
| **Modificación de Historia de Usuario Número:** | |
| **Referencia: No** | |

| | |
|---|---|
| **Programador: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |
| **Prioridad en Negocio: Alta (Alta/Media/Baja)** | **Puntos Estimados(días)**: 17 |
| **Riesgo en Desarrollo: Alto** | **Puntos Reales(días)**: 26 |
| **Descripción:** La aplicación permite Registrar (interfaz Adicionar), Modificar (interfaz Modificar) y Eliminar (interfaz Modificar) un sistema utilizando para esto los parámetros que en dependencia de la situación que se presente se muestren en la aplicación. | |
| **Observaciones:** | |

**Tabla 4:** HU Evaluar Sistema

| **HU "Evaluar Sistema"** | |
|---|---|
| **Número:** 4 | **Nombre HU**: Evaluar Sistema |
| **Modificación de Historia de Usuario Número:** | |
| **Referencia:** No | |
| **Programador: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |
| **Prioridad en Negocio:** Alta **(Alta/Media/Baja)** | **Puntos Estimados(días)**: 14 |
| **Riesgo en Desarrollo:** Alto | **Puntos Reales(días)**: 23 |
| **Descripción:** La aplicación es capaz de evaluar un sistema especificado y se encarga de calcular el nivel porcentual de seguridad que presenta el sistema evaluado. Inmediatamente se le muestra al usuario esta información, dándole a conocer de esta forma el nivel de seguridad que tiene el Sistema que analizó. | |
| **Observaciones:** | |

Las HU anteriormente presentadas demuestran que poseen una prioridad en el negocio de nivel alto lo que significa que posean funcionalidades esenciales para el desarrollo, las que el cliente decide que son fundamentales en el correcto funcionamiento del sistema. Además poseen un riesgo de desarrollo alto debido a que por la complejidad de la implementación o a lo esencial de su uso puedan surgir errores que lleven a la inoperatividad del sistema.

### 2.6- Tarjetas Clase-Responsabilidad-Colaboración

*"Las CRC son una de las principales piezas de diseño que propone la metodología XP. Permite al programador centrarse y apreciar el desarrollo orientado a objetos a través de la representación de las clases"* (Jiménez, 2011).

*"Estas tarjetas constan de tres secciones donde se recogen el nombre de la clase, las funcionalidades o responsabilidades y los colaboradores con otras clases. Una clase es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades o funciones de una clase son las características y las acciones que realizan, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades"* (Jiménez, 2011).

*"El modelado de Clases-Responsabilidades-Colaboraciones (CRC) aporta un medio sencillo de identificar y organizar las clases que resulten relevantes"* (Pressman, 2002).

*"Un modelo CRC es realmente una colección de tarjetas índice estándar que representan clases. Las tarjetas están divididas en tres secciones. A lo largo de la cabecera de la tarjeta usted escribe el nombre de la clase. En el cuerpo se listan las responsabilidades de la clase a la izquierda y a la derecha los colaboradores."* (Pressman, 2002).

Las clases de diseño utilizadas en la implementación del sistema SIECA fueron: *gtrADAutenticacion, gtrADCriterio, gtrADEvaluacionCriterio, gtrADEvaluacionSistema, gtrADIndicador, gtrADSistema, gtrADTipoProceso.*

**Tabla 5:** Tarjeta CRC para Gestionar Indicador.

| CRC "Gestionar Indicador" | |
|---|---|
| **Responsabilidad** | **Colaboradores** |
| **Registrar un nuevo indicador.** | *gtrADCriterio, gtrADTipoProceso* |
| **Asociar un listado de criterios al indicador registrado.** | *gtrADCriterio,* |
| **Eliminar un indicador determinado.** | *gtrADCriterio* |
| **Mostrar listado de indicadores.** | *gtrADCriterio, gtrADTipoProceso* |
| **Mostrar los datos de un indicador dado su código.** | *gtrADCriterio, gtrADTipoProceso* |
| **Mostrar listado de criterios.** | *gtrADCriterio,* |
| **Mostrar listado de criterios asociados al indicador.** | *gtrADCriterio, gtrADIndicador* |

**Tabla 6:** Tarjeta CRC para Gestionar Proceso

| CRC "Gestionar Proceso" | |
|---|---|
| **Responsabilidad** | **Colaboradores** |
| **Registrar un nuevo Proceso** | *gtrADTipoProceso* |
| **Listar los Procesos registrados** | *gtrTipoProceso* |
| **Modificar un Proceso** | |
| **Eliminar un Proceso determinado.** | |
| **Mostrar los datos de un Proceso dado su código.** | |

**Tabla 7:** Tarjeta CRC para Gestionar Sistema

| CRC "Gestionar Sistema" | |
|---|---|
| **Responsabilidad** | **Colaboradores** |
| **Registrar un nuevo Sistema.** | *gtrADSistema* |
| **Eliminar un Sistema determinado.** | |
| **Mostrar listado de Sistemas.** | |
| **Mostrar listado de Sistemas asociados un código.** | |
| **Evaluar un Sistema** | *gtrSistemaInformatico* |

**Tabla 8:** Tarjeta CRC para Evaluar Sistema

| CRC "gtrEvaluacionSistema" | |
|---|---|
| **Responsabilidades** | **Colaboradores** |
| ***crearEvaluacionSistema(ENEvaluacionSistema es)*** | *EvaluacionSistema* |
| ***crearEvaluacionSistema(EvaluacionSistema es)*** | *ENEvaluacionSistema* |

### 2.7- Arquitectura del sistema

#### 2.7.1- Arquitectura en tres capas

Para la implementación del SIECA se utilizó una arquitectura en n-capas, específicamente una arquitectura en basada en tres capas, cuyo objetivo fundamental consiste en separar la capa de presentación al usuario de la de lógica del negocia y esta a su vez de la capa de acceso a datos. Este estilo arquitectónico tiene como ventaja principal que el desarrollo se puede realizar en varios niveles para que en caso de que ocurra algún cambio, sólo se modifique la capa requerida sin tener que revisar todo el código.

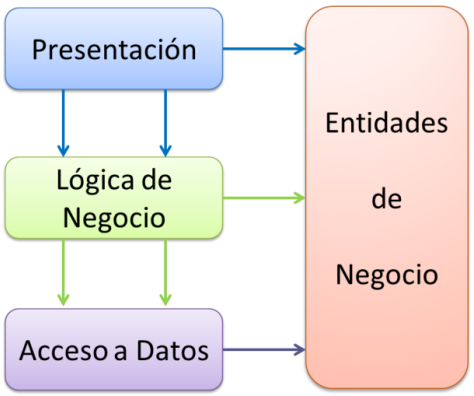



Figura 2: Ejemplo de la arquitectura de SIECA

- **Capa de Presentación:** es donde se le presenta el sistema al usuario, le presenta la información al usuario y se recibe del usuario, realizándose a la vez una serie de validaciones para asegurar la veracidad de los datos recibidos. Esta capa se comunica únicamente con la capa lógica de negocio.
- **Capa de Lógica de Negocio:** es donde residen las clases gestoras de la información, se reciben las peticiones del usuario y se envían las respuestas a la capa de presentación. Se nombra capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes del usuario y presentar los resultados obtenidos, y con la capa de acceso a datos, para enviar datos que necesitan persistirse en la base de datos o recibirlos de la misma.
- **Capa de Acceso a Datos:** está constituida por las clases gestoras del acceso a datos encargadas de acceder a los mismos y que realizan todo el almacenamiento de la información. Recibe solicitudes de almacenamiento o recuperación de información desde la capa de lógica de negocio.

### 2.8- Patrones

*"Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva y reusable. El uso de los patrones de diseño beneficia en gran medida el desarrollo y la calidad del software, pues evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y resueltos anteriormente, además formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño"* (Hernández, 2013).

### 2.8.1- Patrones Generales de Software para Asignación de Responsabilidades

*"Los Patrones Generales de Software para Asignación de Responsabilidades (GRASP, por sus siglas en inglés) describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable"* (Larman, 2009).

#### 2.8.1.2- Patrón Creador

Las clases gestoras de la capa de negocio incluyen las acciones que pueden ser realizadas en el sistema, entre ellas las relacionadas con la creación de los objetos necesarios para acceder a las funcionalidades de las clases persistentes que gestionan el acceso a la base de datos. Ejemplo: *gtrADEvaluacionSistema.java, gtrEvaluacionSistema.java*.

```
public class gtrEvaluacionSistema {

    public static final EvaluacionSistema crearEvaluacionSistema(ENEvaluacionSistema es){
        return new EvaluacionSistema(es.getEvaluacion(), es.getFecha());
    }

    public static final ENEvaluacionSistema crearEvaluacionSistema(EvaluacionSistema es){
        return new ENEvaluacionSistema(es.getEvaluacion(), es.getFecha());
    }

}
```

**Figura 3:** Ejemplo de uso en el sistema del patrón Creador.

#### 2.8.1.3- Patrón Experto

El mapeo[2] de objetos relacionales de la capa de acceso a datos de la Interfaz de Aplicación de Programación para la Persistencia Java (JPA por sus siglas en inglés) encapsula la lógica de los datos. JPA genera clases expertas en manejar la información pertinente a la entidad que representan. Ejemplo: *gtrADEvaluacionSistema.java*.

[2] El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

```
public class gtrADEvaluacionSistema {
    private EntityManagerFactory emf;
    private EntityManager em;

    public gtrADEvaluacionSistema() {
        this.emf = Persistence.createEntityManagerFactory("JavaApplication1PU");
        this.em = emf.createEntityManager();
    }
```

**Figura 4:** Ejemplo de uso en el sistema del patrón Experto.

### 2.8.1.4- Patrón Controlador

Las clases gestoras de SIECA presentes en las capas de acceso a datos y de las reglas del negocio se encargan de la gestión de las entidades. Ejemplo: *gtrADSistema.java*.

```
public class gtrADSistema {
    private EntityManagerFactory emf;
    private EntityManager em;

    public gtrADSistema() {
        this.emf = Persistence.createEntityManagerFactory("JavaApplication1PU");
        this.em = emf.createEntityManager();
    }

    public void salvarSistema(ENSistemaInformatico s){
        em.getTransaction().begin();
        SistemaInformatico sis = new SistemaInformatico();

        try{
            sis = cargarSistemaCodigo(s.getCodigo());
        }
        catch(Exception e){
            sis.setCodigo(s.getCodigo());
        }
        finally{
            sis.setActivo(s.getActivo());
            sis.setDescripcion(s.getDescripcion());
            em.persist(sis);
            em.getTransaction().commit();
            sis = new SistemaInformatico();
        }

    }
```

**Figura 5:** Ejemplo de uso en el sistema del patrón Controlador

### 2.8.1.5- Patrón Alta Cohesión

La información que controlan las clases debe de ser coherente y relacionada con la propia clase. La utilización de la arquitectura en capas permite la aplicación de este patrón. Las clases de la capa de presentación permiten la creación de objetos y de operaciones sobre los mismos, permitiendo que exista flexibilidad a cambios. Ejemplo: *NewFrmPrincipal.java*

```
EvaluacionSistema evs = new EvaluacionSistema();
gtrADIndicador gtrind = new gtrADIndicador();
gtrADSistema gtrsis = new gtrADSistema();
gtrADEvaluacionSistema gtrevs = new gtrADEvaluacionSistema();

evs.setIdSistema(gtrsis.cargarSistemaPorCodigo(jComboBox1.getSelectedItem().toString()));
evs.setIdIndicador(gtrind.cargarIndicadorPorCodigo(jComboBox2.getSelectedItem().toString()));
float evaluacion = cantCritParcInd/totalCritInd;
evs.setEvaluacion(evaluacion);
evs.setFecha(cal.getTime());
```

**Figura 6:** Ejemplo de uso en el sistema del patrón Alta Cohesión

### 2.8.1.6- Patrón Bajo Acoplamiento

Este patron se garantiza en la aplicación basándose en la propia arquitectura del sistema, lo que permite que las dependencias entre las clases se muy poca, ya que solamente las clases de una capa se pueden comunicar con las de la capa inmediatamente inferior.

```
public static final ENEvaluacionCriterio crearEvaluacionCriterio(EvaluacionCriterio ec){
    ENCriterio cri = gtrCriterio.crearCriterio(ec.getIdCriterio());
    ENSistemaInformatico sis = gtrSistemaInformatico.crearSistemaInformatico(ec.getIdSistema());
    return new ENEvaluacionCriterio(ec.getEvaluacion(), sis, cri);
}
```

**Figura 7:** Ejemplo de uso en el sistema del patrón Bajo acoplamiento

### 2.8.2- Patrones del Grupo de Cuatro

Los Patrones del Grupo de Cuatro (GoF, por sus siglas en inglés) *"resuelven problemas específicos de diseño, y vuelven al diseño orientado a objetos más flexible, elegante y extremadamente reutilizable. Ayudan a los diseñadores a reutilizar diseños exitosos basando nuevos diseños en experiencia previa"* (Boizán del Pozo, y otros, 2012).

### 2.8.2.1- Patrón Instancia Única

Instancia Única es un patrón GoF utilizado en las clases gestoras de la capa de lógica de negocio, mediante el cual se define un mecanismo para acceder a los objetos de forma global.

Ejemplo: la funcionalidad *crearCriterio* para la creación de objetos de tipo *Criterio* puede ser accedido desde cualquier lugar del sistema.

```
public class gtrCriterio {

    public static final Criterio crearCriterio(ENCriterio c){
        return new Criterio(c.getNombre(), c.getDescripcion(), c.getActivo());
    }

    public static final ENCriterio crearCriterio(Criterio c){
        return new ENCriterio(c.getNombre(), c.getDescripcion(), c.getActivo());
    }

}
```

**Figura 8:** Ejemplo de uso en el sistema del patrón Instancia Única

### 2.9- Modelo de Datos

El modelo de datos de SIECA se compone de un conjunto de tablas que contribuyen a la gestión de criterios e indicadores de evaluación como se muestra en la Figura 2, debido a las necesidades del negocio y al propio diseño de la base de datos esta se encuentra completamente en la tercera forma normal, por lo que no hay redundancia de datos, atributos nulos ni dependencias transitivas, las llaves primarias son únicas, los atributos de cada tabla dependen totalmente de su llave primaria. Las tablas *criterio*, *indicador* y *tipo_proceso* almacenan el registro de indicadores y procesos. La tabla *sistema_informático* almacena la información referente a los sistemas informáticos evaluados por SIECA o pendientes de evaluación. La tabla *evaluación_sistema* registra la información de la evaluación del sistema una vez que haya sido realizada.

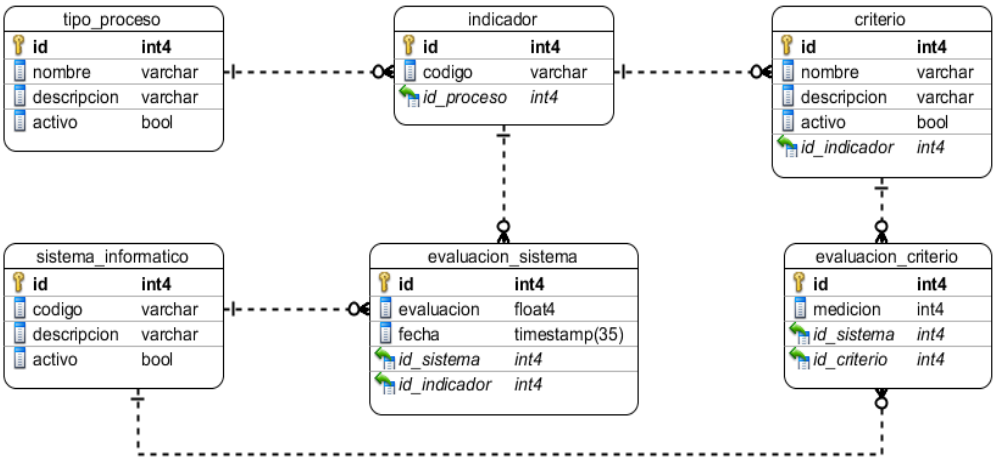



**Figura 9:** Modelo de Datos de SIECA.

**Conclusiones del Capítulo**

- La solución informática SIECA constituye una herramienta para la evaluación de la fiabilidad de la evaluación de la seguridad del control de acceso de los procesos de Identificación y Autenticación, Autorización y Auditoría de los SI.
- Las HU y las tarjetas CRC permitieron realizar el diseño y la implementación del sistema basado en los requisitos funcionales.
- Los Requisitos Funcionales y No Funcionales permitieron el desarrollo de un sistema con un diseño funcional, basado en patrones, buenas prácticas y con niveles de seguridad adecuados a las necesidades del negocio.
- Los patrones de diseño utilizados posibilitaron la correcta asignación de responsabilidades, para que el desarrollo se realizara de una forma sistemática, racional y explicable.
- El modelo de datos de SIECA cumple con las necesidades del sistema, ya que es funcional, sencillo y normalizado por lo que permite la estabilidad.

# CAPÍTULO 3: ANÁLISIS DE LOS RESULTADOS

En el presente capítulo se muestran los resultados obtenidos en las métricas y pruebas realizadas al sistema. La aplicación de la métrica de Tamaño Operacional de Clase, de las Pruebas Unitarias divididas en Pruebas del Camino Básico y Complejidad Ciclomática, además de las Pruebas de Aceptación. Todas estas pruebas se realizaron sobre las funcionalidades de SIECA, obteniéndose y mostrándose de esta forma el análisis de los resultados en correspondencia a las pruebas realizadas.

## 3.1- Métricas de Desarrollo

### 3.1.3- Tamaño Operacional de Clase

Las clases a las que se aplicó la métrica Tamaño Operacional de Clase fueron las clases del paquete *GestoresAD* cuya cantidad de procedimientos se muestra en la Tabla 9. El paquete *GestoresAD* tiene un total de 10 clases y una cantidad de procedimientos total igual a 49 por lo que según los cálculos definidos en la métrica, el promedio de procedimientos es igual a 4,9.

Para el caso de la clase *gtrADEvaluacionSistema* y el resto de las clases del paquete *GestoresAD* según los umbrales definidos por los atributos de calidad: responsabilidad, complejidad de la implementación y reutilización se considera categoría Baja a valores inferiores a 4,9, categoría Media a resultados que se encuentran en el intervalo de 4,9 a 9,8 y categoría Alta en otro caso. La tabla muestra los resultados obtenidos en la evaluación de estos atributos en cada clase del paquete.

**Tabla 9:** Resultados obtenidos en la aplicación de la prueba TOC en las clases del paquete GestoresAD.

| Clase | Cantidad de Procedimientos | Responsabilidad | Complejidad | Reutilización |
|---|---|---|---|---|
| ***gtrADAutenticacion*** | 3 | Baja | Baja | Alta |
| ***gtrADCriterio*** | 6 | Media | Media | Media |
| ***gtrADEvaluacionCriterio*** | 2 | Baja | Baja | Alta |
| ***gtrADEvaluacionSistema*** | 2 | Baja | Baja | Alta |
| ***gtrADIndicador*** | 10 | Alta | Alta | Baja |

| | | | | |
|---|---|---|---|---|
| ***gtrADSistema*** | 10 | Alta | Alta | Baja |
| ***gtrADTipoProceso*** | 5 | Media | Media | Media |
| ***gtrADAutenticacion*** | 3 | Baja | Baja | Alta |
| ***gtrADCriterio*** | 6 | Media | Media | Media |
| ***gtrADEvaluacionCriterio*** | 2 | Baja | Baja | Alta |

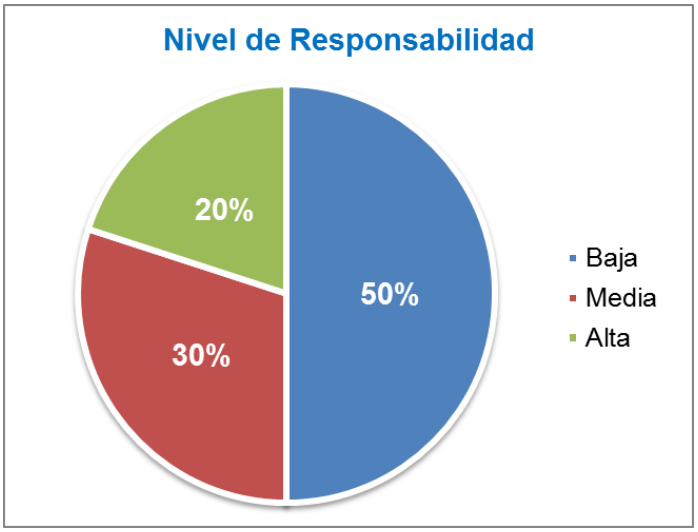



**Figura 10:** Atributo de calidad que representa la responsabilidad.

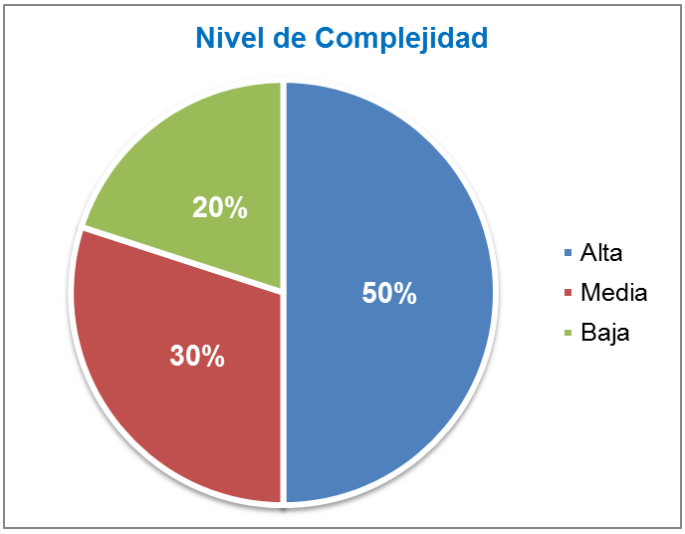



**Figura 11:** Atributo de calidad que representa la complejidad.

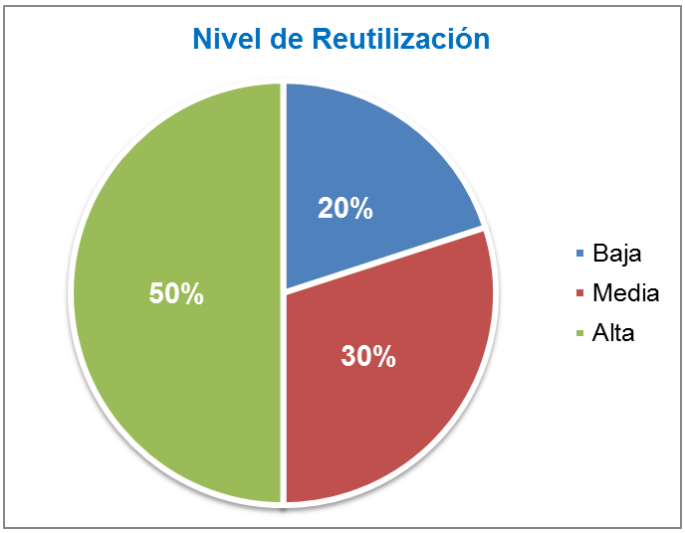



**Figura 12:** Atributo de calidad que representa la reutilización.

Tomando de referencia la información contenida en las **Figura 10**, **Figura 11** y **Figura 12** se puede demostrar que el conjunto de clases contenidas dentro del paquete *GestoresAD* poseen un grado de responsabilidad baja equivalente al 50%, complejidad baja al 50% y reutilización alta al 50% respectivamente. Lo anteriormente planteado demuestra que SIECA posee un nivel de tolerancia a los cambios y reutilización alto, atendiendo a los parámetros establecidos por la propia métrica TOC, lo que cumple con una de las metas trazadas durante la fase de desarrollo del sistema.

### 3.2- Pruebas unitarias

#### 3.2.1- Prueba de Complejidad Ciclomática

```
private void jButtonES1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        int cantP = 0;
        float razon = 0;
        EvaluacionCriterio evc = new EvaluacionCriterio();
        SistemaInformatico sis = new SistemaInformatico();
        List<EvaluacionCriterio> lecri = new ArrayList<EvaluacionCriterio>();
        gtrADCriterio gtrcri = new gtrADCriterio();
        gtrADSistema gtrsis = new gtrADSistema();
        gtrADEvaluacionCriterio gtrec = new gtrADEvaluacionCriterio();

        sis = gtrsis.cargarSistemaPorCodigo(jComboBoxES1.getSelectedItem().toString());
        evc.setIdSistema(sis);

        for (int i = 0; i < jTableRE1.getModel().getRowCount(); i++) {
            if ((Boolean) jTableRE1.getModel().getValueAt(i, 0) == Boolean.TRUE) {
                Criterio criEva = gtrcri.cargarCriterioPorNombre(jTableRE1.getModel().getValueAt(i, 1).toString());
                evc.setIdCriterio(criEva);

                if (jTableES2.getModel().getRowCount() > 0) {
                    for (int j = 0; j < jTableES2.getModel().getRowCount(); j++) {
                        if ((Boolean) jTableES2.getModel().getValueAt(j, 0) == Boolean.TRUE)
                            cantP++;
                        }
                    }
                    int total = jTableES2.getModel().getRowCount();
                    razon = (float) cantP / total;
                    cantCritParcInd += cantP;
                    totalCritInd += total;
                }
            }
        }
        evc.setEvaluacion(razon);
        evaluaciones.add(razon);
        gtrec.SalvarEvaluacionCriterio(evc);
```

**Figura 13:** Ejemplo del código utilizado para calcular la complejidad ciclomática

1. El número máximo de regiones en profundidad del grafo de flujo coincide con la complejidad ciclomática.

   El número de regiones del grafo del flujo definido es 4 lo que se representa en el siguiente resultado:

   V (G)= 4

2. V (G)=A-N+2=15-13+2=4

### 3.2.2- Prueba del Camino Básico

Una vez calculada la complejidad ciclomática define como límite superior cuatro pruebas a realizar para que todo el código se encuentre probado, no siendo complejo el código analizado por el equipo de desarrollo debido que el resultado arrojado por la métrica pertenece al intervalo entre 1-10 indicando que el código perteneciente al sistema informático propuesto en la investigación es simple y sin mucho riesgo.

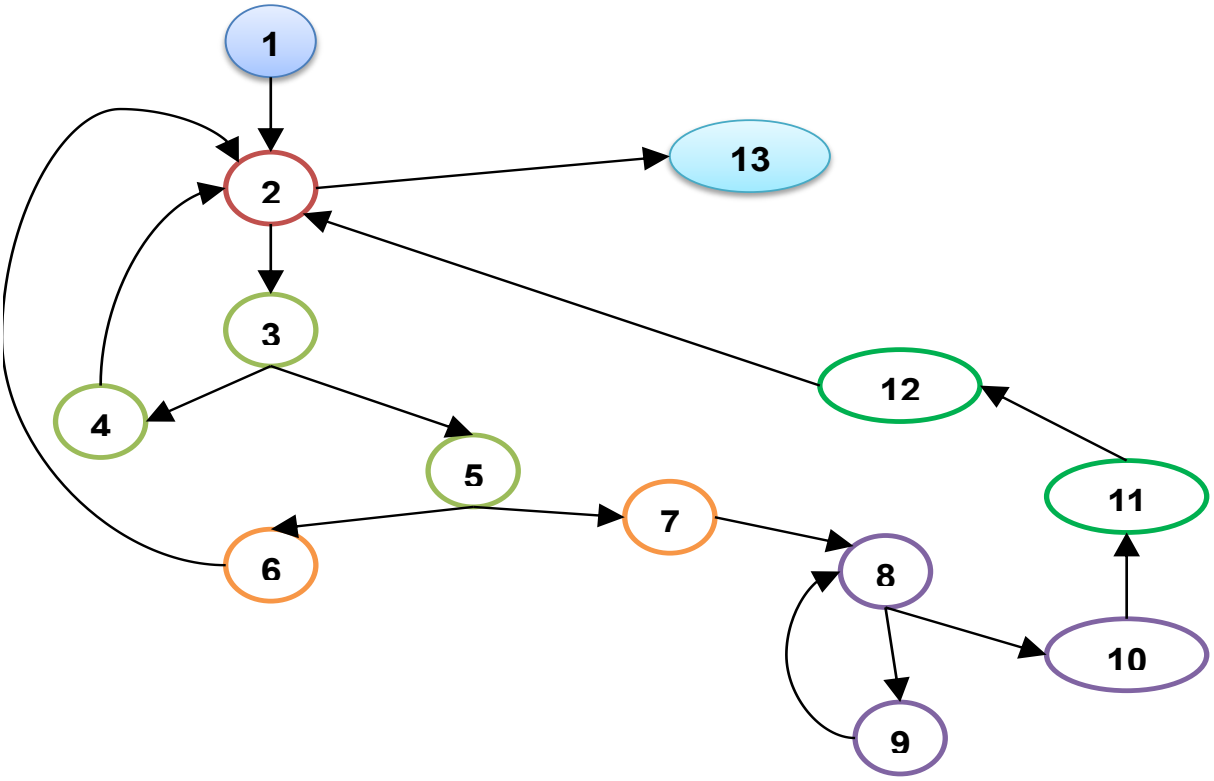



Figura 14: Grafo que muestra el camino básico de la funcionalidad *Evaluar Sistema*.

Caminos independientes establecidos:

Camino1: 1-2-13

- Comienza al inicio de la sentencia *try* y concluye lanzando una excepción si dicha sentencia no se puede ejecutar correctamente.

Camino2: 1-2-3-4-13

- Comienza al inicio de la sentencia try, ejecuta el primer bloque de instrucciones, inicia el ciclo *for* y concluye cuando ejecuta la condicional y se obtiene un resultado falso.

Camino3: 1-2-3-5-6-13

- Comienza al inicio de la sentencia try, ejecuta el primer bloque de instrucciones, inicia el ciclo *for*, ejecuta la primera condicional de dicho ciclo, ejecuta la segunda condicional y termina cuando se obtiene un resultado falso.

Camino4: 1-2-3-5-7-8-10-11-12-13

- Ejecuta exitosamente la funcionalidad *Evaluar Sistema*

Las pruebas unitarias realizadas a la funcionalidad *Evaluar Sistema* sumaron un total de 20, 18 de las cuales (para un 98%) resultaron satisfactorias. Fue evidenciada la estabilidad del código empleado durante la implementación de la funcionalidad. En la segunda iteración el 100% de las pruebas fueron satisfactorias, como se muestra en la **Tabla 10**.

**Tabla 10:** Iteraciones de pruebas unitarias.

| Iteración | Satisfactoria | Insatisfactoria | Total |
|---|---|---|---|
| **1** | 18 | 2 | 20 |
| **2** | 20 | 0 | 20 |

### 3.3- Resultados de las Pruebas de Aceptación

El caso de prueba diseñado para la **Tabla 4:** HU Evaluar Sistema incluye un conjunto de funcionalidades críticas para el desarrollo y despliegue exitoso SIECA. La Tabla 4 muestra el resultado satisfactorio obtenido en la prueba de aceptación Evaluar Sistema.

**Tabla 11:** Ejemplo del Caso de Prueba para la HU Evaluar Sistema.

| Caso de prueba de aceptación | |
|---|---|
| **Código:** HU8_P1 | **Historia de Usuario**: 8 |
| **Nombre:** Evaluar Sistema. | |
| **Descripción:** Prueba para la funcionalidad de evaluar el control de acceso de un sistema informático. | |
| **Condiciones de Ejecución:**<br>El usuario debe seleccionar un sistema a evaluar.<br>El usuario debe seleccionar un indicador del sistema.<br>El usuario debe seleccionar aquellos criterios con los que cumple el sistema que se evalúa. | |
| **Resultados esperados:** El sistema guarda la configuración seleccionada y muestra un mensaje | |

| |
|---|
| informando que la operación se ha realizado con éxito. Además se habilita en la interfaz gráfica un botón que proporciona la posibilidad de ver el resultado de la evaluación obtenida. |
| **Evaluación de la prueba:** Prueba satisfactoria. |

Después de aplicar el caso de prueba correspondiente al proceso Evaluar Sistema (**Tabla 11**), se pudo comprobar que dicha funcionalidad después de realizar los pasos necesarios para su ejecución esta funcionaba correctamente según se necesita.

Fueron diseñadas 10 pruebas funcionales. La realización de las pruebas tuvo lugar en un total de 3 iteraciones y los resultados obtenidos se muestran en la
Tabla 12.

**Tabla 12:** Iteraciones de pruebas funcionales.

| Iteración | Resultado Satisfactorio | Resultado Insatisfactorio | Total |
|---|---|---|---|
| **1** | 5 | 5 | 10 |
| **2** | 8 | 2 | 10 |
| **3** | 10 | 0 | 10 |

**Conclusiones del Capítulo**

1. Las pruebas Unitarias realizadas permitieron constatar que las funcionalidades del sistema presentan un total de caminos inferior a 6.
2. Las pruebas del Camino Básico permitieron analizar todos los posibles caminos de las funcionalidades analizadas, posibilitando que se estudiara toda posible respuesta del código, eliminando con esto los posibles problemas de ejecución de la aplicación.
3. La aplicación de las pruebas TOC arrojaron que SIECA posee un grado de responsabilidad de clases bajo equivalente al 50%, complejidad baja al 50% y reutilización alta al 50% respectivamente, por lo que posee un nivel de tolerancia a los cambios y reutilización alto.
4. Las Pruebas Funcionales demostraron que la aplicación se ejecuta de acorde a las necesidades, obteniéndose resultados satisfactorios a los casos de prueba utilizados.

# CONCLUSIONES

1. El análisis de estándares, métricas, recomendaciones, resoluciones, controles y decretos permitió la síntesis y agrupación de 10 indicadores y 24 criterios de evaluación.

2. El análisis de algoritmos de evaluación permitió adaptar uno de estos al proceso de evaluación de la seguridad del control de acceso.

3. SIECA constituye una herramienta para la evaluación, de manera fiable, de la seguridad de los procesos de Identificación y Autenticación, Autorización y Auditoría de los SI.

4. El uso de metodologías, lenguajes de modelados, herramientas de modelado, gestores de base de datos, lenguajes de programación, entorno de desarrollo integrado y pruebas contribuyó a la implementación y validación del sistema SIECA.

5. Las Pruebas Unitarias y de Aceptación realizadas al sistema fueron satisfactorias.

# REFERENCIAS BIBLIOGRÁFICAS


1. **Acosta, César. 2010.** Extreme Programing (XP). [En línea] 2010. [Citado el: 06 de diciembre de 2013.] http://www.slideshare.net/cesar_acosta/programacin-extrema-extream-programming-xp..
2. **Alarcón, Vicenç Fernández. 2006.** *Desarrollo de sistemas de información. Una metodología basada en el modelado.* Catalunya : Universitat Politècnica de Catalunya, SL, 2006. ISBN: 84-8301-862-4.
3. **Altova. 2010.** Altova UModel. [En línea] 2010. [Citado el: 25 de octubre de 2013.] http://www.altova.com/umodel.html.
4. **Alvarez Zurita, Flor María y Pamela, García Guzmán Anabel. 2007.** *Implementación de un sistema de gestión de seguridad de la información basado en la norma ISO 27001, para la intranet de la Corporación Metropolitana de Salud.* Quito : s.n., 2007.
5. **Alvear Rodriguez, Tatiana y Ronda Ceballos, Carlos. 2005.** *Sistemas de Información para el Control de Gestión. Un apoyo a la gestión empresarial.* Santiago de Chile : Universidad de Chile, 2005.
6. **Asensio, Rafael Menéndez-Barzanallana. 2014.** Ingeniería del software. *Principales herramientas CASE del mercado y su uso.* [En línea] Departamento Informática y Sistemas de la Universidad de Murcia, 2014. [Citado el: 14 de marzo de 2014.] http://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.
7. **AUDISEC. 2010.** *Guía de implantación de un sisitema de gestión de seguridad de la información UNE – ISO/IEC 27001:2007.* 2010.
8. **Baryolo, Oiner Gómez. 2012.** *CAEM: Modelo de control de acceso para sistemas de información en entornos multidominios.* La Habana : s.n., 2012.
9. **Ben-Ari, M. 1996.** *Understanding Programming Languages.* Chichester : John Wiley & Sons, 1996.
10. **Boizán del Pozo, Yurnelis y Galano González, Alexis. 2012.** *Plugin de la herramienta Visual Paradigm para la evaluación del diseño Orientado a Objeto.* La Habana : s.n., 2012.
11. **Centro Nacional de Inteligencia - Centro Nacional Criptológico. 2009.** Las amenazas y vulnerabilidades sobre los Sistemas de Información. [En línea] 2009. [Citado el: 25 de 11 de 2013.] http://www.ati.es/IMG/pdf/vulnerabilidades_press.pdf.
12. **Cid Escalona, Lilian y Sarduy Sánchez, Ismel. 2009.** *Análisis y diseño del subsistema Multimoneda del ERP-Cuba.* La Habana : s.n., 2009.
13. **Ciolli, María Elena. 2007.** *Testing de migración de aplicaciones distribuidas a entornos Web.* 2007.
14. **CMS. 2008.** Content Management System. [En línea] marzo de 2008. [Citado el: 7 de noviembre de 2013.]

http://www.cms.hhs.gov/SystemLifecycleFramework/Downloads/SelectingDevelopmentApproach.pdf.

15. **Cordero Calderón, Marcia Iveth y Ibujés Rivera, María del Carmen. 2008.** *Auditoría de riesgos informáticos del departamento de sistemas de Teleamazonas usando COBIT.* Quito : s.n., 2008.
16. **CORDERO CALDERÓN, MARCIA IVETH y IBUJÉS RIVERA, MARÍA DEL CARMEN. 2008.** *Auditoría de riesgos informáticos del departamento de sistemas de Teleamazonas usando COBIT.* Quito : s.n., 2008.
17. **Dirección de Información. 2012.** Repositorio Institucional. [En línea] 2012. [Citado el: 06 de diciembre de 2013.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04358_11/1/TD_04358_11.pdf.
18. **Domínguez, Jimmy Arturo Brito. 2004.** *Análisis y aprovechamiento de los sistemas de información para una eficiente auditoría y control de gestión.* Guayaquil : s.n., 2004.
19. **Downs, D.** *Issues in Discretionary Access Control.* Oakland, USA : IEEE Computer Society.
20. **Duharte, F. R. 2008.** *Pruebas Unitarias de Software en la Plataforma J2EE.* 2008.
21. **El País. 2013.** 38 millones de cuentas de Adobe, hackeadas. [En línea] EDICIONES EL PAÍS, S.L., 30 de octubre de 2013. [Citado el: 10 de diciembre de 2013.] http://elpais.com/tecnologia/2013/10/30/actualidad/1383137368_646223.html.
22. **—. 2013.** Ataques contra la moneda virtual bitcoin. [En línea] EDICIONES EL PAÍS, S.L., 04 de abril de 2013. [Citado el: 10 de diciembre de 2013.] http://elpais.com/tecnologia/2013/04/04/actualidad/1365085283_638364.html.
23. **—. 2013.** Holanda sufre un cibertaque contra los sitios de Internet del Gobierno. [En línea] EDICIONES EL PAÍS, S.L., 08 de mayo de 2013. [Citado el: 10 de diciembre de 2013.] http://elpais.com/internacional/2013/05/08/actualidad/1368016951_372040.html.
24. **—. 2013.** Un ataque informático tumba la web del Congreso horas antes de su “asedio”. [En línea] EDICIONES EL PAIS, S.L., 25 de abril de 2013. [Citado el: 10 de diciembre de 2013.] http://internacional.elpais.com/politica/2013/04/25/actualidad/1366885719_944004.html.
25. **—. 2013.** Venezuela corta Internet en todo el país por un ataque informático contra Maduro. [En línea] EDICIONES EL PAIS, S.L., 15 de abril de 2013. [Citado el: 10 de diciembre de 2013.] http://internacional.elpais.com/internacional/2013/04/15/actualidad/1365986694_419799.html.
26. **Escribano, G. F. 2002.** *Introducción a Extreme Programming.* 2002.
27. **Esteban, Juan Jesús Muñoz. 2004.** *Metodología para la incorporación de medidas de seguridad en sistemas de información de gran implantación.* Madrid : s.n., 2004.

28. **Fernández. 2010.** Sitio Oficial JAVA. [En línea] 2010. [Citado el: 06 de diciembre de 2013.] http://java.sun.com/people/jag/OriginalJavaWhitepaper.pdf.
29. **Figueroa, Roberth G, S., C. J. y Cabrera, A. A. 2010.** *Metodologías tradicionales vs. Metodologías ágiles.* 2010.
30. **Fritz, Walter.** Intelligent-Systems. [En línea] New Horizons Press. http://www.intelligent-systems.com.ar/intsyst/defsystSp.htm#Contents.
31. **Hernández, Suany Leyva. 2013.** *Sistema informático para la conformación automática de tribunales de tesis en la Facultad 2.* La Habana : s.n., 2013.
32. **IATAC. 2007.** *Software Security Assurance.* 2007.
33. **IBM. 2013.** Rational Rose. [En línea] IBM, 2013. [Citado el: 06 de noviembre de 2013.] http://www-03.ibm.com/software/products/es/rosetechdev/.
34. **Infobae. 2013.** *Febrero, un mes plagado de ataques informáticos en los EEUU.* [En línea] Infobae, 19 de febrero de 2013. [Citado el: 10 de diciembre de 2013.] http://www.infobae.com/2013/02/19/697237-febrero-un-mes-plagado-ataques-informaticos-los-eeuu.
35. **Internet Engineering Task Force. 2013.** Protocolo LDAP. [En línea] 2013. [Citado el: 25 de octubre de 2013.] http://www.ldap-es.org/contenido/04/12/1.-%C2%BFque-esldap%.
36. **ISO.** ISO 27000. [En línea] [Citado el: 21 de octubre de 2013.] WWW.ISO27000.ES.
37. **—. 2005.** *ISO/IEC 17799.* 2005.
38. *ISO/IEC 17799.* **International Organization for Standardization. 2005.** 2005.
39. **ISO/IEC FCD 25040. 2009.** *Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Evaluation reference model and guide.* 2009.
40. **Jiménez, Alain López. 2011.** *Desarrollo de un componente para modelación y visualización en tres dimensiones de objetos geológicos utilizando el framework VTK.* La Habana : s.n., 2011.
41. **Larman, Craig . 2009.** *UML y Patrones.* 2009.
42. **Laudon & Laudon. 2002.** *Sistemas de Información Gerencial.* 2002.
43. **Leivas, Daniel Hugo Enriquez. 2009.** Bases de Datos IBM AS / 400. [En línea] 2009. [Citado el: 06 de diciembre de 2013.] http://www.recursos-as400.com.
44. **Letelier, Patricio y Penadés, Carmen. 2008.** *Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* 2008.
45. **Llorente, Omar Macias. 2011.** *Componente visual para la representación de árboles genealógicos.* La Habana : s.n., 2011.
46. **Lobos, Maria Elena. 2013.** Aprende a Programar. [En línea] 2013. [Citado el: 06 de diciembre de 2013.] http://www.emagister.com/curso-aprende-programar/concepto-lenguaje-programacion.

47. **Martínez Díaz, Yisel y Moreno Rodríguez, Dailier. 2010.** *Módulo Medios de Diagnóstico del Subsistema Web del Sistema Integral para la Atención Primaria.* La Habana : s.n., 2010.
48. **Mendez, Zoilibeth Moreno. 2011.** Importancia y Prevención de La Seguridad de la Información . [En línea] 19 de junio de 2011. [Citado el: 17 de octubre de 2013.] http://controltecnologico.com/2011/06/importancia-y-prevencion-de-la.html.
49. **Mestras Pavón, Juan. 2007.** [En línea] 2007. [Citado el: 25 de octubre de 2013.] http://www.fdi.ucm.es/profesor/jpavon/is2/Lab01RationalRose.pdf.
50. **Microsoft. 2008.** Control de acceso. [En línea] febrero de 2008. [Citado el: 16 de octubre de 2013.] http://technet.microsoft.com/es-es/library/cc732699%28v=ws.10%29.aspx.
51. **—. 2011.** Las medidas de seguridad en la protección de datos. [En línea] 7 de julio de 2011. [Citado el: 2013 de octubre de 2013.] http://www.microsoft.com/business/es-es/Content/Paginas/article.aspx?cbcid=323.
52. **Netbeans. 2014.** Netbeans Comunity. [En línea] 2014. [Citado el: 14 de marzo de 2014.] www.netbeans.org/index_es.html.
53. **Noble, Angela Martin y de Robert Biddle, James. 2004.** *The XP Customer Role in Practice: Three Studies Agile.* 2004.
54. *Nuevas Tendencias en Sistemas de Información: Procesos y Servicios.* **R. de Soto, Adolfo y Cuervo Fernández, Eva.**
55. **Object Management Group. 2014.** Object Management Group Business Process Model and Notation. [En línea] Object Management Group, Inc., 2014. [Citado el: 14 de marzo de 2014.] http://www.bpmn.org/.
56. **O'Brien, J. A. 2001.** *Sistemas de Información Gerencial.* Bogotá : Irwin McGraw-Hill, 2001.
57. **Oficial, Gaceta. 2011.** *Decreto Ley 281. Principios de organización y funcionamiento del Sistema de Información del Gobierno.* La Habana : Ministerio de Justicia, 2011. págs. 1-6.
58. **Oracle. 2010.** Oracle Identity Management. [En línea] 2010. [Citado el: 25 de noviembre de 2013.] http://www.oracle.com.
59. **Ortiz, Yudisney Vázquez. 2007.** *Análisis y diseño del sistema de información y gestión para la calidad del Software Educativo.* La Habana : s.n., 2007.
60. **Pinagapani, S. 2009.** *A Comparative Study of Access Control Languages.* Shanghai : IEEE Computer Society, 2009.
61. **Popa, Marius. 2011.** *Techniques and Methods to Improve the Audit Process of the Distributed Informatics Systems Based on Metric System.* Bucarest : s.n., 2011.
62. **PostgreSQL Core Team. 2013.** PostgreSQL. [En línea] 2013. [Citado el: 25 de octubre de 2013.] http://www.postgresql.org/.

63. **Pressman, Roger. 2002.** *Ingeniería de Software "Un enfoque práctico" 5ta ed.* s.l. : Graw Hill, 2002.
64. **Ranchal, Juan. 2012.** Gestión de sesiones web: ataques y medidas de seguridad. [En línea] 2012 de marzo de 2012. [Citado el: 2013 de octubre de 17.] http://muyseguridad.net/2012/03/29/gestion-sesiones-web-ataques-medidas-seguridad/.
65. *Resolución No. 127 /2007.* **Ministerio de la Informática y las Comunicaciones. 2007.** La Habana : s.n., 2007.
66. **Robles, R. 2008.** *Application of Role-Based Access Control for Web Environment.* s.l. : IEEE Computer Society, 2008.
67. **Sánchez, Alice Naranjo. 2009.** *Administración y control - evaluación de controles y seguridades del sistema de cartera de las empresas.* Guayaquil : Universidad de Guayaquil, 2009.
68. **Santiago Zaragoza, María de Lourdes . 2014.** Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado (RUP). [En línea] 2014. [Citado el: 14 de marzo de 2014.] http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.
69. *Seguridad de la Información.* **Centro Nacional de Información de la Calidad. 2012.** España : s.n., 2012.
70. *Seguridad de la información en un mundo sin fronteras.* **Ernst & Young Global Limited. 2011.** Mancera : s.n., 2011.
71. **Solis, E. 2008.** IBM: Seguridaden ambientes SOA. Retos, patrones y soluciones. [En línea] 2008. [Citado el: 11 de noviembre de 2013.] www.cert.uy/historico/pdf/IBMSOASecurity.pdf.
72. **Sparks, Geoffrey. 2006.** Craftware. [En línea] 2006. [Citado el: 25 de 10 de 2013.] http://www.craftware.net/es/products/ea/modelo_de_proceso_de_negocio.pdf..
73. **Synergix. 2013.** Tipos de requisitos: Funcional vs. No Funcional. [En línea] Synergix, 2013. [Citado el: 05 de marzo de 2014.] http://synergix.wordpress.com/2008/07/07/requisito-funcional-y-no-funcional/.
74. **Tao, W. 2010.** *RBAC Permission Consistency Static Analysis Framework.* Jiangsu : IEEE Computer Society, 2010.
75. **Trasobares, Alejandro Hernandez.** *Los sistemas de información: evolución y desarrollo.* Zaragoza : s.n.
76. **Vance, A y Molyneux, B. 2012.** *Reducing Unauthorized Access by Insiders through User Interface Design: Making End Users Accountable.* Hawaii, USA : IEEE Computer Society, 2012.
77. **Visual Paradigm For UML. 2012.** Visual Paradigm For UML. [En línea] 2012. [Citado el: 06 de diciembre de 2013.] http://www.visual-paradigm.com/product/vpsuite.
78. **Visual Paradigm International. 2013.** Visual Paradigm. [En línea] 2013. [Citado el: 25 de octubre de 2013.] http://www.visual-paradigm.com.

79. **Wordpress. 2013.** Ayuda de Wordpress. [En línea] Wordpress, 2013. [Citado el: 10 de diciembre de 2013.] http://ayudawordpress.com/ataque-masivo-de-fuerza-bruta-para-acceder-a-sitios-wordpress/.

# ANEXOS

**Tabla 13:** HU Adicionar Criterio.

| HU "Adicionar Criterio" | |
|---|---|
| **Número: 1** | **Nombre HU**: Adicionar Criterio |
| **Modificación de Historia de Usuario Número:** | |
| **Referencia: No** | |
| **Programador: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |
| **Prioridad en Negocio: Alta (Alta/Media/Baja)** | **Puntos Estimados(días)**: 10 |
| **Riesgo en Desarrollo: Bajo** | **Puntos Reales(días)**: 12 |
| **Descripción:** **Comienza en el instante en que el administrador del sistema elige la pestaña Adicionar y pulsa la opción Indicador donde selecciona el sistema que desea evaluar. Posteriormente escribe el código de identificación del indicador y selecciona el tipo de proceso al que pertenece, además rellena los campos de Nombre y Descripción. Consecutivamente realiza este procedimiento para cada uno de los indicadores que desee añadir y finalmente hace clic en el botón Aceptar.** | |
| **Observaciones:** | |

**Tabla 14:** HU Adicionar Proceso.

| HU "Adicionar Proceso" | |
|---|---|
| Número**: 2** | **Nombre HU**: Adicionar Proceso |
| Modificación de Historia de Usuario Número**:** | |
| Referencia**: No** | |
| Programador**: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |
| Prioridad en Negocio**: Alta (Alta/Media/Baja)** | **Puntos Estimados(días)**: 1115 |
| Riesgo en Desarrollo**: Bajo** | **Puntos Reales(días)**: 16 |
| Descripción**:** **Comienza en el instante en que el administrador del sistema elige la pestaña Adicionar y pulsa la opción Proceso. Posteriormente rellena los campos de Nombre y Descripción.** | |

| |
|---|
| **Consecutivamente realiza este procedimiento para cada uno de los procesos que desee añadir y finalmente hace clic en el botón Aceptar.** |
| Observaciones**:** |

**Tabla 15:** HU Adicionar Sistema.

| **HU "Adicionar Sistema"** | |
|---|---|
| Número**: 3** | **Nombre HU**: Adicionar Sistema |
| Modificación de Historia de Usuario Número**:** | |
| Referencia**: No** | |
| Programador**: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |
| Prioridad en Negocio**: Alta (Alta/Media/Baja)** | **Puntos Estimados(días)**: 15 |
| Riesgo en Desarrollo**: Bajo** | **Puntos Reales(días)**: 9 |
| Descripción**:** **Comienza en el instante en que el administrador del sistema elige la pestaña Adicionar y pulsa la opción Sistema. Posteriormente rellena los campos de Nombre y Descripción. Consecutivamente realiza este procedimiento para cada uno de los sistemas que desee añadir y finalmente hace clic en el botón Aceptar.** | |
| Observaciones**:** | |

**Tabla 16:** HU Modificar Indicador

| **HU "Modificar Indicador"** | |
|---|---|
| Número**: 4** | **Nombre HU**: Modificar Indicador |
| Modificación de Historia de Usuario Número**:** | |
| Referencia**: No** | |
| Programador**: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |
| Prioridad en Negocio**: Alta (Alta/Media/Baja)** | **Puntos Estimados(días)**: 12 |
| Riesgo en Desarrollo**: Bajo** | **Puntos Reales(días)**: 16 |
| Descripción**:** **Comienza en el instante en que el administrador del sistema elige la pestaña Modificar y pulsa la opción Indicador. Posteriormente selecciona el Indicador que desea modificar utilizando para esto el código de identificación del mismo. Después modifica los parámetros** | |

| |
|---|
| **existentes del indicador cambiándolos por otros. Consecutivamente realiza este procedimiento para cada uno de los indicadores que desee modificar y finalmente hace clic en el botón Aceptar.** |
| Observaciones**:** |

**Tabla 17:** HU Modificar Proceso

| **HU "Modificar Proceso"** | |
|---|---|
| Número**: 5** | **Nombre HU**: Modificar Proceso |
| Modificación de Historia de Usuario Número**:** | |
| Referencia**: No** | |
| Programador**: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |
| Prioridad en Negocio**: Alta (Alta/Media/Baja)** | **Puntos Estimados(días)**: 10 |
| Riesgo en Desarrollo**: Bajo** | **Puntos Reales(días)**: 9 |
| Descripción**:** **Comienza en el instante en que el administrador del sistema elige la pestaña Modificar y pulsa la opción Proceso. Posteriormente selecciona el Proceso que desea modificar utilizando para esto el código de identificación del mismo. Después modifica los parámetros existentes del Proceso cambiándolos por otros. Consecutivamente realiza este procedimiento para cada uno de los procesos que desee modificar y finalmente hace clic en el botón Aceptar.** | |
| Observaciones**:** | |

**Tabla 18:** HU Modificar Sistema

| **HU "Modificar Sistema"** | |
|---|---|
| Número**: 6** | **Nombre HU**: Modificar Sistema |
| Modificación de Historia de Usuario Número**:** | |
| Referencia**: No** | |
| Programador**: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |
| Prioridad en Negocio**: Alta (Alta/Media/Baja)** | **Puntos Estimados(días)**: 9 |
| Riesgo en Desarrollo**: Bajo** | **Puntos Reales(días)**: 10 |
| Descripción**:** | |

| |
|---|
| **Comienza en el instante en que el administrador del sistema elige la pestaña Modificar y pulsa la opción Sistema. Posteriormente selecciona el Sistema que desea modificar utilizando para esto el código de identificación del mismo. Después modifica los parámetros existentes del Sistema cambiándolos por otros. Consecutivamente realiza este procedimiento para cada uno de los Sistema que desee modificar y finalmente hace clic en el botón Aceptar.** |
| Observaciones**:** |

**Tabla 19:** HU Eliminar Indicador.

| **HU "Eliminar Indicador"** | |
|---|---|
| Número**: 7** | **Nombre HU**: Eliminar Indicador |
| Modificación de Historia de Usuario Número**:** | |
| Referencia**: No** | |
| Programador**: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |
| Prioridad en Negocio**: Alta (Alta/Media/Baja)** | **Puntos Estimados(días)**: 12 |
| Riesgo en Desarrollo**: Bajo** | **Puntos Reales(días)**: 5 |
| Descripción**:** **Comienza en el instante en que el administrador del sistema elige la pestaña Modificar y pulsa la opción Indicador. Posteriormente selecciona el Indicador que desea modificar utilizando para esto el código de identificación del mismo. Después elimina los parámetros existentes del indicador seleccionado. Consecutivamente realiza este procedimiento para cada uno de los indicadores que desee eliminar y finalmente hace clic en el botón Aceptar.** | |
| **Observaciones** | |

**Tabla 20:** HU Eliminar Sistema.

| **HU "Eliminar Sistema"** | |
|---|---|
| Número**: 8** | **Nombre HU**: Eliminar Sistema |
| Modificación de Historia de Usuario Número**:** | |
| Referencia**: No** | |
| Programador**: Wilfredo Palma Pérez** | **Iteración Asignada**: 1 |

| | |
|---|---|
| Prioridad en Negocio**: Alta**<br>**(Alta/Media/Baja)** | **Puntos Estimados(días)**: 6 |
| Riesgo en Desarrollo**: Bajo** | **Puntos Reales(días)**: 7 |
| Descripción**:**<br>**Comienza en el instante en que el administrador del sistema elige la pestaña Modificar y pulsa la opción Sistema. Posteriormente selecciona el Sistema que desea eliminar utilizando para esto el código de identificación del mismo y después pulsa sobre el botón Eliminar. Consecutivamente realiza este procedimiento para cada uno de los Sistema que desee modificar y finalmente hace clic en el botón Aceptar.** | |
| Observaciones**:** | |

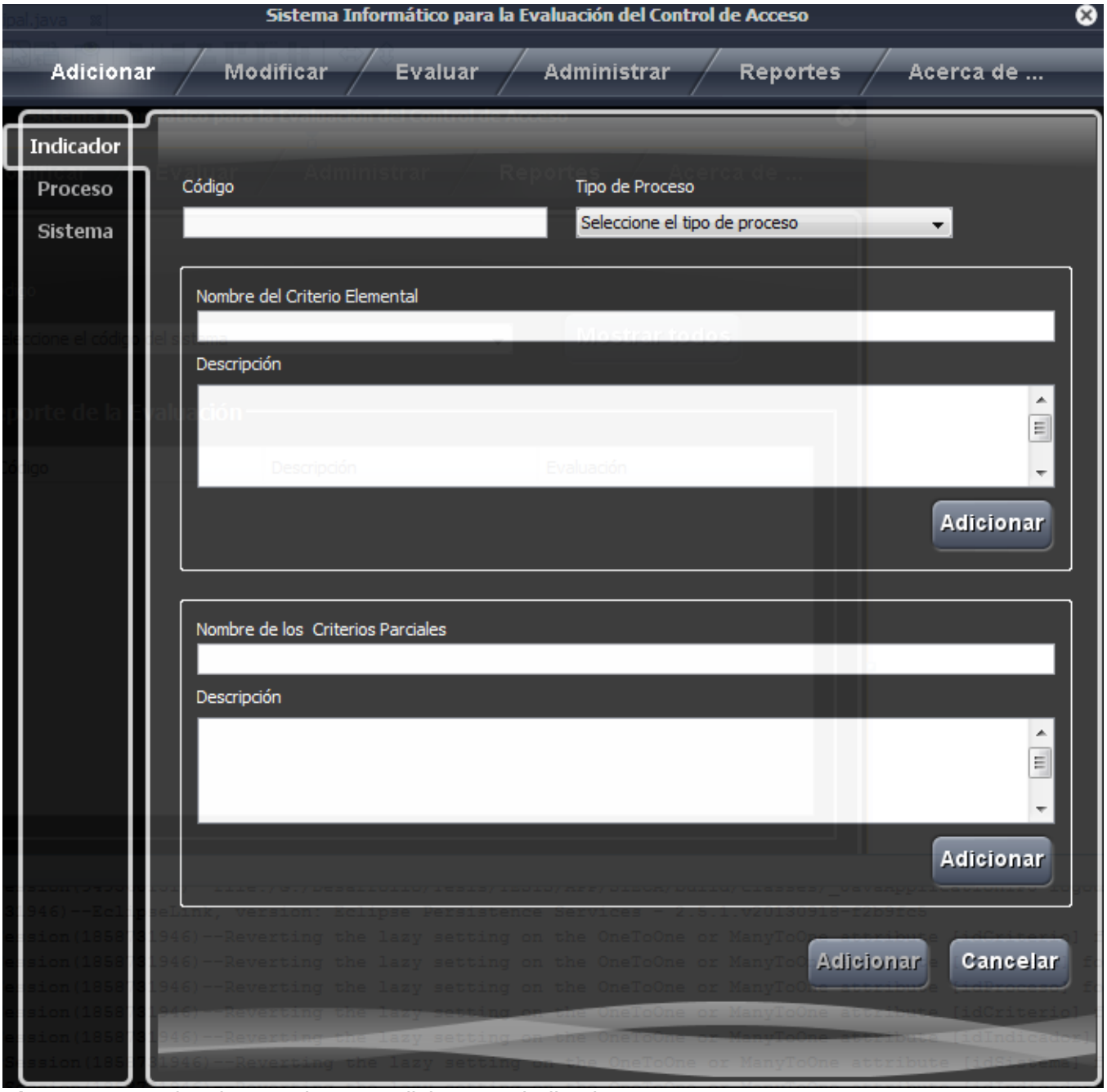



**Figura 15:** Interfaz de usuario para adicionar un indicador.

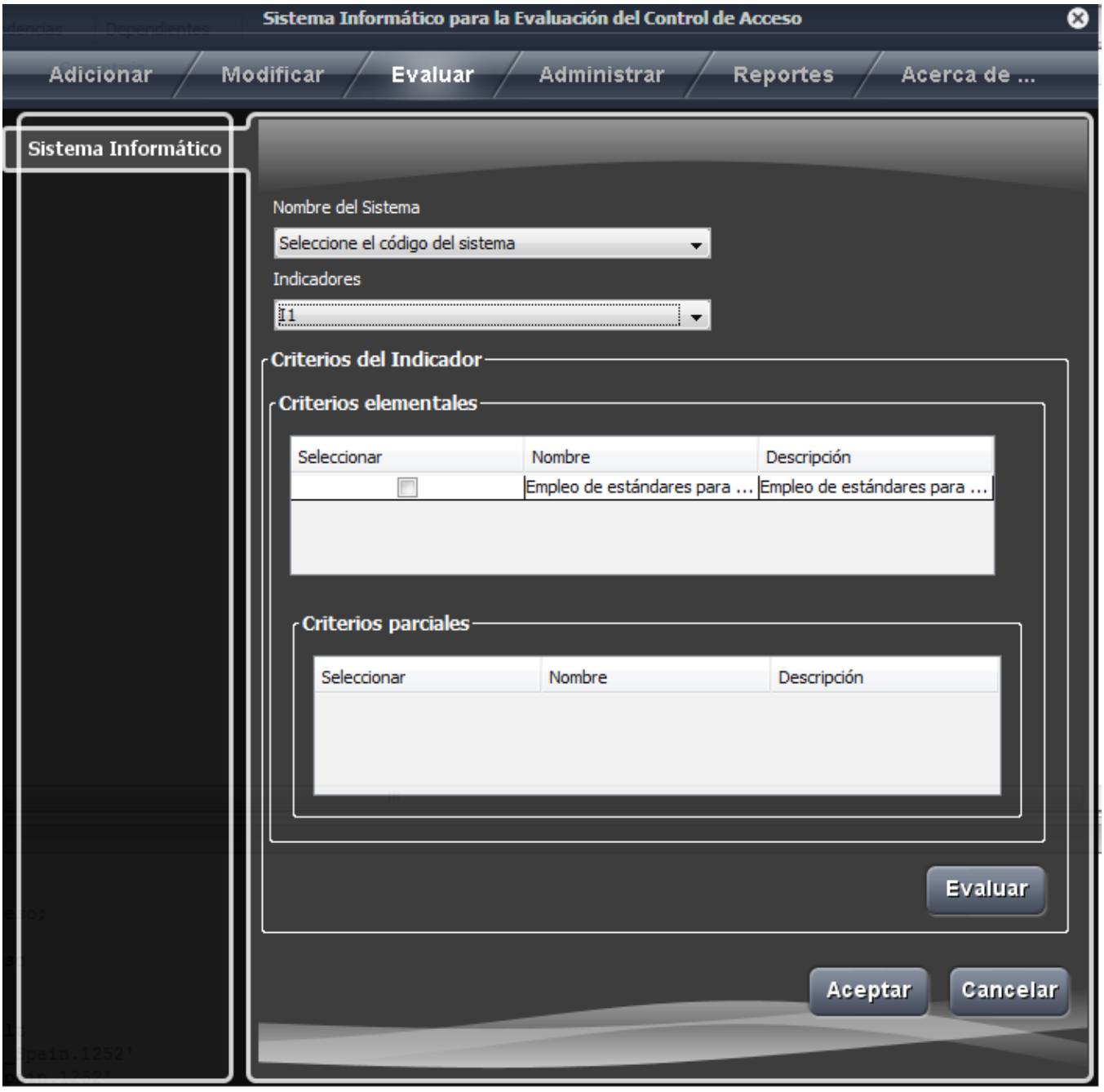



**Figura 16:** Interfaz de usuario para evaluar un sistema.


**Figura 17:** Interfaz de usuario de contacto

# GLOSARIO DE TÉRMINOS

**API:** Applications Programing Interfaces. Conjunto de subrutinas que aportan los sistemas operativos a los programas de aplicación para el acceso a los recursos y servicios prestados por el ordenador.

**Base de Datos:** Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Una base de datos relacional es una base de datos que cumple con el modelo relacional, que no es más que un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos que su idea fundamental es el uso de "relaciones".

**CASE:** Acrónimo de Computer Aided Software Engineering (Ingeniería de Software Asistida por Ordenador), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

**Clases:** Conjunto de objetos que comparten atributos, operaciones, relaciones y semántica, las mismas representan los conceptos fundamentales del sistema.

**Empresa:** Una empresa es el organismo social integrado por elementos humanos, técnicos y materiales cuyo objetivo natural y principal es la obtención de utilidades.

**GRASP:** General Responsibility Assignment Software Patterns de sus siglas en inglés, traducido como patrones generales de asignación de responsabilidades.

**IDE:** Integrated Development Environment (Ambiente Integrado de Desarrollo). Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

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

**International Electrotechnical Commission:** La Comisión Electrotécnica Internacional (CEI o IEC, por sus siglas del idioma inglés International Electrotechnical Commission) es una organización de normalización en los campos eléctrico, electrónico y tecnologías relacionadas. Desarrollan Numerosas normas conjuntamente con la ISO (normas ISO/IEC).

**Lenguaje de Programación:** Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis.

**Linux:** Sistema operativo libre tipo Unix.

**Multiplataforma:** Término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en Windows en un procesador x86, en GNU/Linux en un procesador x86, y en Mac OS X en uno x86.

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

**Requisitos Funcionales:** los requisitos funcionales van a ser las capacidades o condiciones que deberá tener el sistema a construir.

**Requisitos no Funcionales:** los requisitos no funcionales son las cualidades o propiedades que deberá tener el producto.

**Software Libre:** Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

**SQL:** Son las siglas de Structured Query Language, Lenguaje de consulta estructurado, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre la misma.

**TIC:** Las tecnologías de la información y la comunicación (TIC) agrupan los elementos y las técnicas utilizadas en el tratamiento y la transmisión de las informaciones, principalmente de informática, Internet y telecomunicaciones.

**UML:** lenguaje de modelado visual consistente en el cual se expresan los resultados de numerosas metodologías de orientación a objetos existentes.

**UNIX:** Sistema operativo multitarea, multiusuario. Gran parte de las características de otros sistemas más conocidos como MS-DOS están basadas en este sistema, muy extendido para grandes servidores.