Universidad de las Ciencias Informáticas

Facultad 6

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

Herramienta para la Gestión de No Conformidades para el Centro GEYSED

Autor: Wilfredo Carballo Rodríguez.

Tutor: Ing. Frank Alain Castro Sierra.

Ciudad de La Habana, Junio 2011

"Año del 53 de la Revolución"

"La juventud tiene que crear. Una juventud que no crea es una anomalía realmente."

"Ernesto Che Guevara"

Mi tesis la dedico con todo mi amor y cariño.

A ti DIOS que me diste la oportunidad de vivir y de regalarme una familia maravillosa.

Con mucho cariño principalmente a mis padres que me dieron la vida y han estado conmigo en todo momento. Gracias por todo papá y mamá por darme una carrera para mi futuro y por creer en mí, mami hemos pasado momentos difíciles siempre me has estado apoyando y brindando todo tu amor, por todo esto te agradezco de todo corazón. Este trabajo me llevo un año hacerlo y es para ustedes, por ser su único hijo aquí esta lo que ustedes me brindaron, solamente les estoy devolviendo lo que ustedes me dieron en un principio.

A mi Abuela Soledad que es también como una madre para mi, a ella le debo mucho.

A mi novia que a pesar de llevar 2 meses juntos me ha ayudado mucho en este trabajo y gracias a ella logré enderezar un poco mi vida. "Te quiero mucho"

A mis hermanos Reinaldo y Yasmany gracias por estar conmigo y apoyarme siempre.

A todos mis amigos Liudmila, Karelia, Arianna, Lillian, Osmel, Diosbel, Rodolfo y Reyalex, muchas gracias por estar conmigo en todo este tiempo donde he vivido momentos felices y tristes, gracias por ser mis amigos y recuerden que siempre los llevaré en mi corazón.

Y no me puedo ir sin antes decirles, que sin ustedes a mi lado no lo hubiera logrado, tantas desveladas sirvieron de algo y aquí está el fruto. Les agradezco a todos ustedes con toda mi alma el haber llegado a mi vida y el compartir momentos agradables, esos momentos son los que nos hacen crecer y valorar a las personas que nos rodean. Los quiero mucho y nunca los olvidaré.

Agradecimientos

"Mientras el río corra, los montes hagan sombra y en el cielo haya estrellas, debe durar la memoria del beneficio recibido en la mente del hombre agradecido."

Les agradezco a todos mis amigos por todo este tiempo juntos el cual nunca olvidaré, a todos ustedes porque estuvieron conmigo en los momentos más difíciles y siempre me hicieron mirar hacia otros horizontes, por eso muchas gracias Liudmila, Karelia, Arianna, Lillian, Osmel, Diosbel, Rodolfo y Reyalex.

Le agradezco especialmente a mi familia que sin ellos todo este sueño nunca se hubiese hecho realidad:

A mi primos Daniela, Merlin, Jose, Lizandra, el Koki.

A mis tías Zunilda, Nieves, Maridelvis.

A mi Abuela Soledad.

A mi novia que a pesar de llevar 2 meses juntos me ha ayudado mucho en este trabajo y gracias a ella logré normalizar un poco mi vida. "Te quiero mucho"

Le agradezco a mi mamá porque has compartido conmigo todos los valores que te hacen tan especial: bondad, perdón, honestidad, perseverancia, consideración y paciencia. Lo que trato de decirte mamá, es que eres la base sobre lo que se ha formado mi personalidad y sólo me queda una palabra: GRACIAS!

A todos, ¡Muchas Gracias! Eternamente agradecido.

Wilfredo Carballo Rodriguez.

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro Desarrollo Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Wilfredo Carballo Rodríguez Autor	Ing. Frank A. Castro Sierra Tutor

Tutor:

Ing. Frank Alain Fernandez Sierra. Ingeniero en Ciencias Informáticas, UCI, 2007. Profesor del Departamento de Ciencias Básicas, Facultad 6.

Correo electrónico: fcastro@uci.cu.

Teléfono: 837 3140.

Co-Tutor

Ing. Carlos De Jesús Andrés González. Ingeniero en Ciencias Informáticas, UCI, 2010. Jefe del Subsistema de administración de PRIMICIA, Facultad 6.

Correo electrónico: candres@uci.cu.

Teléfono: 837 3093

Resumen

La investigación está asociada al grupo de calidad que se dedica a examinar los artefactos de los proyectos que pertenecen al departamento de Señales Digitales el cual forma parte del Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED). Se realizaron entrevistas las cuales permitieron tener una comunicación con otras personas para obtener ideas, que permitieron desarrollar una herramienta capaz de gestionar las No Conformidades del Centro GEYSED. Se analizaron otras soluciones existentes llegando a la conclusión de que ninguna se ajusta a las necesidades del grupo de calidad. Garantizando la soberanía tecnológica se fundamentó la selección de herramientas y tecnologías que permitieron obtener los artefactos del producto final y se aseguró el correcto funcionamiento de la aplicación desarrollada realizándole las pruebas pertinentes.

Palabras Claves

Calidad de software, Gestión, No Conformidades.

Introducción

El poderoso auge de las Tecnologías de la Información y las Comunicaciones (TIC) ha cambiado los paradigmas, estrategias reconocidas y establecidas por muchos años. Dentro de las TIC la industria del software alcanza una posición relevante por su característica de controlar o hacer accesible en la mayoría de los casos, adelantos electrónicos. En la sociedad moderna el uso de las tecnologías se ha convertido en parte fundamental de la vida cotidiana. Teléfonos celulares, robots industriales, satélites, fibra óptica y otros adelantos científicos-técnicos permiten que problemas anteriormente complicados y engorrosos puedan solucionarse en poco tiempo y con gran eficiencia.

No existe una forma de organización global definida para la industria del software que rija su desarrollo o forma de implementación. Cada país, según sus características y posibilidades adopta la vía que considera más favorable en este sentido. Las TIC están fuertemente sujetas al desarrollo económico de cada nación. La industria del software como apéndice a las nuevas tecnologías se encuentra bajo las mismas condiciones.

La calidad en el desarrollo y mantenimiento de las aplicaciones de software se ha convertido actualmente en uno de los principales objetivos estratégicos de las entidades que se dedican a la producción debido a que cada vez más los procesos principales de dichas empresas dependen del sistema informático para su buen funcionamiento.

Calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenimiento, portabilidad, usabilidad, seguridad e integridad. Ésta es medible y varía de un sistema a otro. (1)

En la evolución experimentada por la calidad de software se ha transitado de un tratamiento centrado fundamentalmente en la inspección y detección de errores, a una aproximación más sistemática, dada la importancia que ha adquirido la calidad en la ingeniería del software. (2)

Una de las formas de medir la calidad son los modelos de Calidad que guían a las Organizaciones a la mejora continua y la competitividad dándoles especificaciones de qué tipo de requisito debe de implementar para poder brindar productos y servicios de alto nivel. Existen varios tipos de modelos de

calidad, tales como: Modelo de Implementación (Métrica3), Modelo Orientado al proceso del ciclo de vida del software (Norma ISO/IEC 12007), Capability Maturity Model Integration (CMMI) y muchos más. (1)

Uno de los métodos más importantes para crear un software con eficiencia es la Norma Internacional ISO 9001:2008 promueve, la oportunidad, detección y disposición de las No Conformidades, con la finalidad de identificar, segregar y tomar una disposición de productos no conformes y de este modo evitar su uso intencionado.

En Cuba se vienen desarrollando soluciones informáticas para el beneficio de toda la sociedad y para la exportación a otros países, principalmente en Latinoamérica, de ahí la importancia de concretar esfuerzos e ideas conjuntas entre centros investigativos, empresas y universidades, para la estandarización y mejora de los procesos de ingeniería de software y la creación de sistemas.

Entre las principales instituciones fundadas para apoyar la creación de software en Cuba, la Universidad de las Ciencias Informáticas (UCI) juega un papel decisivo ya que tiene la responsabilidad de formar profesionales preparados en la rama de la informática. Dentro ella se encuentra el Grupo de Calidad del departamento de Señales Digitales que forma parte del Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED), como parte de sus tareas gestiona y controla la calidad durante cada una de las fases de desarrollo. También desarrolla un conjunto de acciones necesarias y sistemáticas para garantizar que las exigencias del cliente satisfagan los requisitos de calidad establecidos. Estas tareas son esenciales para asegurar que las actividades técnicas se lleven a cabo correctamente dentro de restricciones de costo, tiempo y calidad. Estas restricciones se aplican a lo largo de todo el proceso de Ingeniería de Software ya que comprenden métodos, herramientas de análisis, diseño, codificación, pruebas, revisión de técnicas formales, control de ajuste a los estándares de desarrollo del software, mecanismos de medidas y de información entre otras tareas.

Actualmente todo ese proceso se desarrolla manualmente, recopilándose un gran volumen de información, el cual se almacena en documentos, para luego ser calculada manualmente; lo que provoca la generación de un número considerable de errores mediante su análisis, además de que en ocasiones el proceso puede demorar en dependencia del volumen de datos con el que se trabaje. Como consecuencia de estos retrasos en ocasiones no se puede conocer con exactitud la situación actual en que se encuentra

un determinado indicador de calidad que se desee, además de esto, los integrantes del equipo de calidad se ven en la necesidad de realizar grandes esfuerzos para entregar los resultados de su trabajo en el tiempo definido, que al final atentan contra la calidad y determinación del producto final que debe ser entregado.

De lo anterior puede plantearse la necesidad de registrar las No Conformidades de forma centralizada siguiendo un estándar que permita tabular esta información y obtener reportes, que posibiliten la toma de decisiones sobre qué acciones realizar en aras de resolver los errores detectados en el menor período de tiempo posible.

A partir de lo estudiado, y teniendo en cuenta la situación problemática existente se plantea como ***problema a resolver*:**

¿Cómo perfeccionar los procesos de gestión de No Conformidades en el grupo de Calidad del centro GEYSED?

Para dar solución al problema anterior se propone como ***objetivo general*** desarrollar una herramienta que gestione las No Conformidades del Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED).

Se define como ***objeto de estudio*** el proceso de gestión de No Conformidades y como ***campo de acción*** la informatización de los procesos de Gestión de No Conformidades en el Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED).

Para dar cumplimiento al objetivo general se definieron las siguientes ***tareas***

1. Caracterizar el Proceso de gestión de No Conformidades.
2. Definir las técnicas, lenguajes y plataformas de programación.
3. Modelar los procesos de Negocio, Diseño e Implementación del sistema.
4. Validar la herramienta implementada.

Teniendo en cuenta lo anterior, se plantea la siguiente ***idea a defender*** si se realiza una herramienta para la gestión de No Conformidades se mejorará la gestión de los procesos del departamento de señales digitales.

A lo largo del proceso investigativo se utilizaron una serie de métodos científicos que serán explicados a continuación:

Métodos Teóricos

Analítico-sintético: Es utilizado para estudiar y analizar documentos y bibliografías de diferentes autores para poder realizar una amplia investigación sobre los elementos que se relacionan con el objeto de estudio.

Análisis histórico lógico: Se utiliza para hacer un estudio sobre la evolución y desarrollo que han tenido los sistemas de gestión de No Conformidades, lo que permite ver en qué etapa se encuentran dichos sistemas y cuál es la tecnología más factible para el desarrollo de los mismos.

Métodos Empíricos

Observación: Permite realizar valoraciones y obtener información a partir de lo observado. Esto se manifiesta principalmente cuando se realizan observaciones sobre el funcionamiento de sistemas similares al desarrollado, lo que da una visión de cómo tiene que ser el sistema a realizar en su forma externa.

Entrevista: Permite tener una comunicación con otras personas que pueden servir para apoyar y dar ideas en cuanto al sistema que se quiere construir.

El contenido de este documento está estructurado de la siguiente manera:

Capítulo 1: Fundamentación Teórica de la investigación.

Se especifican los elementos teóricos que sustentan al problema científico, además de analizarse todo lo referente a los procesos de Gestión de las No Conformidades y también se analizan soluciones existentes.

Capítulo 2: Tendencias y tecnologías a desarrollar.

Centra su atención en la descripción de la metodología que guiará el desarrollo de la investigación y las herramientas usadas para el cumplimiento de los objetos.

Capítulo 3: Características del Sistema.

Se define el Modelo de negocio, Especificación de los Requisitos del Sistema y Casos de Uso.

Capítulo 4: Implementación y Pruebas.

Está enfocado a la implementación del sistema con el objetivo de darle solución a los requisitos funcionales. Se realiza el modelo de Implementación y se aplican patrones de diseño y de arquitectura a cada una de las clases del diseño. Además de probar todas las funcionalidades del software.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se abordará los conceptos fundamentales que se trabajaran durante el desarrollo de la investigación y que serán referenciados en el resto del documento. Así como características y ventajas de los modelos de calidad de software que se harán uso. También se afrontará todo lo referente al estado actual de la calidad de software. Se dará una breve valoración de algunos de los sistemas automatizados que existen y que están vinculados al campo acción. Finalmente se especifica el proceso de gestión de No Conformidades que se tendrá en cuenta a la hora de realizar la herramienta.

1.2 Conceptos asociados al dominio del problema

1.2.1 Definición de Calidad

En la actualidad se pueden encontrar muchas definiciones en los textos de calidad, todas muy similares, según el Diccionario de la Real Academia Española (DRAE) es la propiedad o conjunto de propiedades inherentes a un objeto que permiten apreciarlo como mejor, igual o peor que otros objetos de su especie.

Pressman[1] plantea que la calidad es la concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente que desea el usuario. Es un conjunto de propiedades inherentes a un objeto que le confieren capacidad para satisfacer necesidades implícitas o explícitas. (3)

La calidad puede definirse como la conformidad relativa con las especificaciones y satisfacciones en un producto cumpliendo todas las expectativas que busca algún cliente. Existen varios autores calificados como Gurús de la Calidad que definen calidad como sigue:

Los requisitos del software son la base de las medidas de calidad, si falta concordancia con los requisitos faltará a su vez calidad. Los estándares o metodologías definen un conjunto de criterios de desarrollo que

[1] Roger S. Pressman es un ingeniero de software americano, autor y consultor y presidente de R.S. Pressman & Associates

guían la forma en que se aplica la ingeniería del software, si no se sigue ninguna metodología siempre habrá falta de calidad. (5)

En cualquier caso, el incumplimiento de los requisitos de calidad en cualquiera de las dos dimensiones puede tener consecuencias negativas graves para cualquiera o todos los interesados en el proyecto. Por ejemplo:

- Cumplir con los requisitos del cliente haciendo trabajar en exceso al equipo del proyecto puede producir consecuencias negativas, tales como un desgaste elevado de los empleados, errores involuntarios.
- Cumplir con los objetivos del cronograma del proyecto ejecutando apresuradamente las inspecciones de calidad planificadas puede producir consecuencias negativas cuando los errores no se detectan. (6)

"*Calidad: grado en el que un conjunto de características inherentes cumple con los requisitos*" (6)

Según Luis Andres Arnauda Sequera[2] Define la norma ISO 9000 "Conjunto de normas y directrices de calidad que se deben llevar a cabo en un proceso".

Se entendió por calidad y en lo que se basó este trabajo es en aportar valor al cliente, esto es, ofrecer condiciones de uso del producto lo cual el cliente espera recibir y a un precio accesible. También, la calidad se refiere a minimizar las pérdidas que un producto pueda causar a la sociedad humana mostrando cierto interés por parte de la empresa a mantener la satisfacción del cliente.

1.2.2 Calidad de Producto de Software

El objetivo principal de la calidad a nivel de producto es conocer el nivel de satisfacción de los clientes con el producto evaluado y con la empresa desarrolladora, también facilita ofrecerle al software la opinión de usuarios con experiencia en la utilización del producto.

Los sujetos que intervienen son:

[2] Luis Andres Arnauda Sequera: Define la norma ISO 9000.

- Cliente o usuario
- Empresario o empresa desarrolladora
- Personal técnico (6)

La calidad que pueden alcanzar los productos de software, y en general cualquier producto, está sometida a cómo se desarrollan cada una de las etapas de la vida del producto, partiendo por la definición de la idea del producto hasta la entrega y mantención del mismo.

Así la estrategia de calidad a un producto considera actividades tales como:

- Administración de la calidad, asegurando minimizar las diferencias entre los recursos presupuestados y los recursos realmente utilizados en las distintas etapas, equipamiento y tiempo de desarrollo.
- Uso de tecnología de Ingeniería de Software eficiente, considerando métodos de desarrollo y herramientas.
- Aplicación de técnicas formales a lo largo de todo el proceso.
- Minimización de las variaciones entre los productos, disminuyendo las diferencias y defectos entre versiones.
- Testeo acucioso en diferentes etapas del desarrollo.
- Control de la documentación, tanto de apoyo al desarrollo como la entregada al usuario final, generado en cada etapa, y verificación de los posibles cambios y modificaciones que pudiera sufrir.
- Correcta mantención y servicios de post-venta. (8)

Las normas en las que se apoya y definen las características de calidad de software que deben estar presentes en todos los productos son:

- **Mantenibilidad:** La capacidad del software de ser modificado. Las modificaciones pueden incluir las correcciones, mejoras o adaptación del software a los cambios en el ambiente, y en los requisitos y las especificaciones funcionales.
- **Funcionalidad:** La capacidad del software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas cuando el software se usa bajo las condiciones especificadas.

- **Portabilidad:** La capacidad del software debe ser transferido de un ambiente a otro.
- **Eficiencia:** La capacidad del software para proporcionar la requerida ejecución, en relación con la cantidad de recursos usados, bajo las condiciones declaradas.
- **Confiabilidad:** La capacidad del software para mantener su nivel de ejecución cuando se usa bajo las condiciones especificadas.
- **Usabilidad:** La capacidad del software de ser comprendido, aprendido, utilizado y de ser amigable para el usuario, cuando se emplee bajo las condiciones especificadas. (7)

Los factores que determinan la calidad del software se clasifican en tres grupos:

- Operaciones del producto: características operativas.
  - Corrección (¿Hace lo que se le pide?)
  - El grado en que una aplicación satisface sus especificaciones y consigue los objetivos encomendados por el cliente.
  - Fiabilidad (¿Lo hace de forma fiable todo el tiempo?)
  - El grado que se puede esperar de una aplicación lleve a cabo las operaciones especificadas y con la precisión requerida.
  - Eficiencia (¿Qué recursos hardware y software necesito?)
  - La cantidad de recursos hardware y software que necesita una aplicación para realizar las operaciones con los tiempos de respuesta adecuados.
  - Integridad (¿Puedo controlar su uso?)
  - El grado con que puede controlarse el acceso al software o a los datos y a personal no autorizado.
  - Facilidad de uso (¿Es fácil y cómodo de manejar?)
  - El esfuerzo requerido para aprender el manejo de una aplicación, trabajar con ella, introducir datos y conseguir resultados.

- Revisión del producto: capacidad para soportar cambios.
  - Facilidad de mantenimiento (¿Puedo localizar los fallos?)
  - El esfuerzo requerido para localizar y reparar errores.
  - Flexibilidad (¿Puedo añadir nuevas opciones?)

  - El esfuerzo requerido para modificar una aplicación en funcionamiento.
  - Facilidad de prueba (¿Puedo probar todas las opciones?)
  - El esfuerzo requerido para probar una aplicación de forma que cumpla con lo especificado en los requisitos.

- Transición del producto: adaptabilidad a nuevos entornos.
  - Portabilidad (¿Podré usarlo en otra máquina?)
  - El esfuerzo requerido para transferir la aplicación a otro hardware o sistema operativo.
  - Reusabilidad (¿Podré utilizar alguna parte del software en otra aplicación?)
  - Grado en que partes de una aplicación pueden utilizarse en otras aplicaciones.
  - Interoperabilidad (¿Podrá comunicarse con otras aplicaciones o sistemas informáticos?)
  - El esfuerzo necesario para comunicar la aplicación con otras aplicaciones o sistemas informáticos. (9)

1.2.3 Gestión de la calidad de software

La gestión de la calidad del software es el conjunto de actividades que determinan la calidad, los objetivos y responsabilidades, se implanta por los cuatro medios que se mencionan a continuación**:**

- Planificación de calidad.
- Control de la calidad.
- Aseguramiento (garantía) de la calidad.
- Mejora de la calidad. (10)

La gestión de la calidad se aplica normalmente a nivel de empresa, también puede haber una gestión de calidad dentro de la gestión de cada proyecto. (5)

La política de calidad son las directrices y objetivos generales de una organización, relativas a la calidad, tal como se expresan formalmente por la alta dirección. (9)

La gestión de calidad moderna complementa la dirección de proyectos. Por ejemplo, ambas disciplinas reconocen la importancia de:

- Satisfacción del cliente. Entender, evaluar, definir y gestionar las expectativas, de modo que se cumplan los requisitos del cliente. Esto requiere una combinación de conformidad con los requisitos (el proyecto debe producir lo que dijo que produciría) y ser adecuado para su uso (el producto o servicio debe satisfacer las necesidades reales).
- La prevención sobre la inspección. El coste de prevenir errores es generalmente mucho menor que el coste de corregirlos cuando son detectados por una inspección.
- Responsabilidad de la dirección. El éxito requiere la participación de todos los miembros del equipo, pero proporcionar los recursos necesarios para lograr dicho éxito sigue siendo responsabilidad de la dirección.
- Mejora continua. El ciclo planificar-hacer-revisar-actuar es la base para la mejora de la calidad (según la definición de Shewhart, modificada por Deming, en el Manual de la ASQ, páginas 13–14, American Society for Quality, 1999). Los modelos de mejora de procesos incluyen CMM. (6)

1.2.4 Planificación de la Calidad

La planificación de calidad implica identificar qué normas de calidad son relevantes para el proyecto y determinar cómo satisfacerlas. Es uno de los procesos clave a la hora de llevar a cabo el Grupo de Procesos de Planificación y durante el desarrollo del plan de gestión del proyecto, y debería realizarse de forma paralela a los demás procesos de planificación del proyecto. (6)

Uno de los principios fundamentales de la gestión de calidad moderna es la calidad. Se planifica, se diseña e incorpora; no se incluye mediante inspección.

Las herramientas y técnicas principales son:

- Análisis Coste-Beneficio

La planificación de calidad debe tener en cuenta las concesiones entre costes y beneficios. El principal beneficio de cumplir con los requisitos de calidad es un menor reproceso, lo cual significa mayor productividad, menores costes y mayor satisfacción de los interesados. El coste principal de cumplir con los requisitos de calidad son los gastos asociados con las actividades de Gestión de la Calidad del Proyecto.

- Estudios Comparativos

Un estudio comparativo implica comparar prácticas del proyecto reales o planificadas con las de otros proyectos, a fin de generar ideas de mejoras y de proporcionar una base respecto a la cual medir el rendimiento. Estos otros proyectos pueden estar dentro o fuera de la organización ejecutante y pueden encontrarse dentro de la misma área de aplicación o en otra.

- Diseño de Experimentos

El diseño de experimentos (DOE) es un método estadístico que ayuda a identificar qué factores pueden influir sobre variables específicas de un producto o proceso en desarrollo o en producción. También desempeña un rol en la optimización de productos o procesos. El aspecto más importante de esta técnica es que proporciona un marco estadístico para cambiar sistemáticamente todos los factores importantes. (6)

1.2.5 Control de la Calidad (QC)

Son las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en dos objetivos fundamentales:

- Mantener bajo control un proceso.
- Eliminar las causas de los defectos en las diferentes fases del ciclo de vida. (5)

Control de Calidad es realizar una observación constante acerca del cumplimiento de las tareas que pueden ofrecer una calidad objetiva a la forma en cómo se está desarrollando un proyecto de Ingeniería de Software, es decir una vigilancia permanente a todo el proceso de desarrollo y ciclo de vida del software.

El control de la calidad permite realizar las rectificaciones pertinentes al desarrollo en cuanto este empieza a desviarse de sus objetivos, alejando la inclusión de la calidad al trabajo. Estas rectificaciones son posibles gracias a una retroalimentación de las etapas superiores. (8)

En el control de calidad se debe tener presente los costos que esta involucra. Si se piensa en las tareas que se debe realizar en este control, puede observase que es necesario llevar a cabo tareas de búsqueda de problemas, testeo, realimentación, rectificación, elaboración, modificación y estudio de la documentación; entre otras actividades.

Todas ellas tienen costos involucrados (incluso puede darse la inclusión de equipos destinados al aseguramiento de la calidad). Pero debe existir un compromiso, ya que un excesivo costo en el control de la calidad puede hacer que este proceso se torne ineficiente. (8)

Los diagramas de control determinan si el proceso es estable o no, o si tiene un rendimiento predecible. Los diagramas de control pueden servir como una herramienta de obtención de datos para mostrar cuándo un proceso está sujeto a una variación por una causa especial, que crea una condición fuera de control. (6)

La revisión de reparación de defectos es una acción llevada a cabo por el departamento de control de calidad o por una organización con una denominación similar, para asegurar que los defectos de productos se reparen y cumplan con los requisitos o especificaciones. (6)

Las mediciones de Control de Calidad (QC) representan los resultados de las actividades de control de calidad que se retroalimentan del aseguramiento de la calidad para revaluar y analizar las normas y procesos de calidad de la organización ejecutante. Las acciones correctivas implican acciones llevadas a cabo como resultado de una medición de QC que indica que el proceso de fabricación o desarrollo excede los parámetros establecidos.

El plan de gestión del proyecto se actualiza a fin de reflejar los cambios en el plan de gestión de calidad que resultan de los cambios al realizar el proceso de QC. Los cambios solicitados (adiciones, modificaciones o supresiones) al plan de gestión del proyecto y sus planes subsidiarios se procesan mediante revisión y disposición a través del proceso de control integrado de cambios. (6)

1.2.6 Aseguramiento de la Calidad

El aseguramiento de la calidad es la aplicación de actividades planificadas y sistemáticas relativas a la calidad, para asegurar que el proyecto emplee todos los procesos necesarios para cumplir con los requisitos. (6)

Algunos autores prefieren decir garantía de calidad en vez de aseguramiento, se considera en este trabajo de diploma que el término aseguramiento es más adecuado ya que pretende dar confianza en que el producto tiene calidad.

El aseguramiento de calidad del software está presente en:

- Métodos y herramientas de análisis, diseño, programación y prueba.
- Inspecciones técnicas formales en todos los pasos del proceso de desarrollo del software.
- Estrategias de prueba multiescala.
- Control de la documentación del software y de los cambios realizados.
- Procedimientos para ajustarse a los estándares (y dejar claro cuando se está fuera de ellos).
- Mecanismos de medida (métricas).
- Registro de auditorías y realización de informes. (10)

Una de las actividades para el aseguramiento de calidad del software es la verificación y validación del software a lo largo del ciclo de vida, incluyendo las pruebas y los procesos de revisión e inspección.

El análisis del proceso de aseguramiento de la calidad incluye el análisis causal, una técnica específica para analizar un problema / situación, determinar las causas subyacentes que lo provocan y crear acciones preventivas para problemas similares. (6)

El aseguramiento de calidad del software se diseña para cada aplicación antes de comenzar a desarrollarla y no después. (2)

1.2.7 Mejora de la Calidad

La mejora de la calidad es la parte de la Gestión de la Calidad que contribuye, por medio de las mediciones, a los análisis de los datos y auditorías, a efectuar mejoras en la calidad del software. Orientada a aumentar la capacidad de cumplir con los requisitos de la calidad. Los requisitos pueden estar relacionados con cualquier aspecto tal como la eficacia, la eficiencia o la trazabilidad. (1)

La organización debe mejorar continuamente la eficacia del sistema de gestión de la calidad por medio de la utilización de la política de la calidad, objetivos de la calidad, resultados de las auditorías, análisis de datos, acciones correctivas y preventivas y la revisión por la dirección.

Para implementar un programa de mejoras es necesario definir procesos, decidir qué se quiere mejorar, definir qué medidas serán necesarias recoger, cómo y dónde tomarlas, gestionarlas mediante herramientas, utilizarlas para la toma de decisiones y reconocer las mejoras.

Cuando el proceso a mejorar es el de desarrollo del software, es importante definir qué objetivos se quieren alcanzar, para reducir el número de medidas y, en consecuencia, el coste de recopilarlas y el impacto sobre la actividad de producción de software. (11)

1.2.8 No Conformidades

Las pruebas de software no garantizan que un software esté libre de errores, sino que se detecten la mayor cantidad de No Conformidades posibles en el mismo para su debida corrección. De acuerdo a la norma ISO 9000 una No Conformidad es el incumplimiento de un requisito. (19)

1.3 Modelos y Estándares de Calidad de Software

El objetivo de las organizaciones fabricantes del software es producir software de buena calidad de una manera sistemática y predecible, para lograr esto es necesario aplicar modelos y estándares que aseguren y guíen la calidad del software.

Los modelos y estándares de calidad de software forman parte de la ingeniería del software, es la disciplina cuyo fin es la producción de software, libre de fallas, entregado a tiempo, dentro del presupuesto y que satisfaga las necesidades del cliente. (12)

La IEEE[3] define los modelos y estándares de calidad como la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software, es decir la aplicación de ingeniería al software. Cuando se cumplen con las normas y parámetros se consigue una disminución de los defectos, satisfacción del cliente, y la ansiada calidad.

1.3.1 Serie ISO 9000

La Organización Internacional para la Estandarización estipula que sus estándares son producidos de acuerdo a los siguientes principios:

- **Consenso:** Son tenidos en cuenta los puntos de vistas de todos los interesados: fabricantes, vendedores, usuarios, grupos de consumidores, laboratorios de análisis, gobiernos, especialistas y organizaciones de investigación.
- **Aplicación Industrial Global:** Soluciones globales para satisfacer a las industrias y a los clientes mundiales.
- **Voluntario:** La estandarización internacional es conducida por el mercado y por consiguiente basada en el compromiso voluntario de todos los interesados del mercado. (12)

Los estándares de ISO contribuyen a hacer el desarrollo, la fabricación y la fuente de los productos y de los servicios más eficientes, más seguros y más limpios. El plan estratégico 2005-2010 de ISO conforma la visión global de la organización en 2010, junto con los siete objetivos estratégicos precisados para resolver las expectativas de los miembros y accionistas de la ISO. (4)

Las normas se revisan cada 5 años para garantizar la adecuación a las tendencias y dinámica del contexto mundial. En el año 2000 cobraron vigencia los cambios propuestos para las ISO 9000, los que se tradujeron en las actuales Normas ISO 9000 versión 2000. (13)

[3] IEEE (**Instituto de Ingenieros Eléctricos y Electrónicos**) Una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

Cuando una organización cuenta con una Certificación en ISO 9000 generalmente experimenta:

- Aumento en la aceptación de los clientes.
- Reducciones en los costos de operación.
- Implementación de un modelo o sistema controlado y documentado.
- Un mejoramiento extiendo en la manera de trabajar. (13)

La norma ISO 9000 contiene las directrices para seleccionar y utilizar las normas para el aseguramiento de la calidad, es decir, es la que permite seleccionar un modelo de aseguramiento de calidad, entre las que se describen las ISO 9001/9002/9003.

La norma ISO 9004 establece directrices relativas a los factores técnicos, administrativos y humanos que afectan a la calidad del producto, es decir, establece directrices para la gestión de la calidad.

La norma ISO 9004-2 establece directrices relativas a los factores técnicos, administrativos y humanos que afectan a la calidad de los servicios, es decir, se refiere especialmente a los servicio.

Las normas ISO 9001/9002/9003 establecen requisitos que determinan los elementos. Tienen que comprender los sistemas de calidad, pero no es el propósito imponer uniformidad en los sistemas de calidad. Son genéricas e independientes de cualquier industria o sector económico concreto. (13)

1.3.2 Modelo de Madurez y Capacidad Integrado (CMMI)

Capability Maturity Model Integration (CMMI) es un modelo de Aseguramiento de la Calidad que busca la mejora continua de las organizaciones mediante el análisis y re-diseño de los procesos que subyacen en la organización. Fue creado por el SEI (Software Engineering Institute) de la Universidad de Carnegie-Mellon y patrocinado por el Ministerio de Defensa de los Estados Unidos. (14)

Con el propósito de lograr la mejora de los procesos, CMMI provee:

- Una forma de integrar los elementos funcionales de una organización.

- Un conjunto de mejores prácticas basadas en casos de éxito probado de organizaciones experimentadas en la mejora de procesos.
- Ayuda para identificar objetivos y prioridades para mejorar los procesos de la organización, dependiendo de las fortalezas y debilidades de la organización que son obtenidas mediante un método de evaluación.
- Un apoyo para que las empresas complejas en actividades productivas puedan coordinar sus actividades en la mejora de los procesos.
- Un punto de referencia para evaluar los procesos actuales de la organización.

El modelo CMMI contiene las siguientes 22 áreas de proceso:

- Análisis de Causas y Resolución (CAR).
- Gestión de la Configuración (CM).
- Análisis de Decisiones y Resolución (DAR).
- Gestión Integrada de Proyectos (IPM).
- Medición y Análisis (MA).
- Innovación y Despliegue Organizacionales (OID).
- Definición de Procesos Organizacionales (OPD).
- Enfoque Organizacional en Procesos (OPF).
- Rendimiento de Procesos Organizacionales (OPP).
- Formación Organizacional (OT).
- Monitorización y Control de Proyecto (PMC).
- Planificación de Proyecto (PP).
- Aseguramiento de Calidad de Procesos y Productos (PPQA).
- Integración de Producto (PI).
- Gestión Cuantitativa de Proyectos (QPM).
- Gestión de Requerimientos (REQM).
- Desarrollo de Requerimientos (RD).
- Gestión de Riesgos (RSKM).
- Gestión de Acuerdos con Proveedores (SAM).

- Solución Técnica (TS).
- Validación (VAL).
- Verificación (VER).

El modelo para ingeniería de sistemas (SE-CMM) establece 6 Niveles de Capacidad posibles para cada una de las 22 áreas de proceso implicadas en la ingeniería de sistemas. La organización puede decidir cuáles son las Áreas de Proceso (PA) que quiere mejorar determinando así su perfil de capacidad. El modelo presenta dos representaciones:

- La visión continua de una organización mostrará la representación de nivel de capacidad de cada una de las áreas de proceso del modelo.
- La visión escalonada definirá a la organización dándole en su conjunto un nivel de madurez del 1 al 5.

Los 6 niveles definidos en CMMI para medir la capacidad de los procesos son:

- **Incompleto:** El proceso no se realiza, o no se consiguen sus objetivos.
- **Ejecutado:** El proceso se ejecuta y se logra su objetivo.
- **Gestionado:** Además de ejecutarse, el proceso se planifica, se revisa y se evalúa para comprobar que cumple los requisitos.
- **Definido:** Además de ser un proceso gestionado se ajusta a la política de procesos que existe en la organización, alineada con las directivas de la empresa.
- **Cuantitativamente Gestionado:** Además de ser un proceso definido se controla utilizando técnicas cuantitativas.
- **Optimizando:** Además de ser un proceso Cuantitativamente Gestionado, de forma sistemática se revisa y modifica o cambia para adaptarlo a los objetivos del negocio. Mejora continua.

Los componentes principales de CMMI son:

Área de proceso: Conjunto de prácticas relacionadas que son ejecutadas de forma conjunta para conseguir un conjunto de objetivos.

Componentes Requeridos

- **Objetivo genérico**: Los objetivos genéricos asociados a un nivel de capacidad establecen lo que una organización debe alcanzar en ese nivel de capacidad.
- **Objetivo específico**: Los objetivos específicos se aplican a una única área de proceso y localizan las particularidades que describen qué se debe implementar para satisfacer el propósito del área de proceso.

El logro de cada uno de esos objetivos en un área de proceso significa mejorar el control en la ejecución del área de proceso.

Componentes Esperados

- **Práctica genérica**: Una práctica genérica se aplica a cualquier área de proceso porque puede mejorar el funcionamiento y el control de cualquier proceso.
- **Práctica específica**: Una práctica específica es una actividad que se considera importante en la realización del objetivo específico al cual está asociado.

Las prácticas específicas describen las actividades esperadas para lograr la meta específica de un área de proceso.

Componentes Informativos

- Propósito
- Notas introductorias
- Nombres
- Tablas de relaciones práctica - objetivo
- Prácticas
- Productos típicos
- Sub-prácticas: Una sub-práctica es una descripción detallada que sirve como guía para la interpretación de una práctica genérica o específica.

- Ampliaciones de disciplina: Las ampliaciones contienen información relevante de una disciplina particular y relacionada con una práctica específica.
- Elaboraciones de prácticas genéricas: Una elaboración de una práctica genérica es una guía de cómo la práctica genérica debe aplicarse al área de proceso. (16)

Evaluaciones

Una evaluación de CMMI corresponde al estudio y análisis de uno o más procesos realizado por un equipo capacitado de profesionales, utilizando un modelo de referencia de evaluación como base para determinar, a lo menos, fortalezas y debilidades dentro de una organización. Un método de evaluación puede ser aplicado para distintos propósitos, incluyendo evaluaciones internas para mejora de los procesos, evaluaciones de capacidad de selección de proveedores, evaluaciones de monitoreo de procesos, entre otros enfoques. (15)
El SEI[4] ha publicado dos documentos guías que actualmente son utilizados para realizar una evaluación de CMMI:

- Appraisal Requirements for CMMI (ARC).
- Standard CMMI Appraisal Method for Process Improvement (SCAMPI).

ARC define un conjunto de requerimientos considerados esenciales para realizar una evaluación CMMI mientras que SCAMPI es la referencia para la evaluación. Se definen en ARC tres clases de evaluaciones: clase A, clase B y clase C. Las clases definen los requerimientos que debe cumplir una evaluación de cierta complejidad.

La clase A de ARC corresponde al método de evaluación que satisface el 100% de los requerimientos que el documento define y es la única evaluación que se considera oficial para otorgar un nivel de certificación de CMMI en una organización. Se denomina SCAMPI clase A.

Este método permite comprender de mejor forma las capacidades de la organización, identificando fortalezas y debilidades en sus procesos y relacionar estas fortalezas y debilidades con el modelo de

[4] **SEI** es un instituto federal estadounidense de investigación y desarrollo, fundado por el Congreso de los Estados Unidos en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software.

referencia CMMI. El método permite además enfocar la organización en el mejoramiento continuo de procesos y priorizar los planes de mejora; finalmente permite evaluar con una nota el nivel de madurez en el cual se encuentra una organización.

SCAMPI clase A consta de tres fases:

- Planificar y preparar la evaluación.
- Llevar a cabo la evaluación.
- Reportar resultados de la evaluación.

Dentro de estas fases se ejecutan una serie de procesos. Algunos de ellos incluyen asignar responsabilidades, documentar el proceso, entrevistar a personas de la organización, agrupar los datos que se utilizarán, verificar y validar los procesos con el estándar, asignar notas o ratings, crear reportes. Se espera contar con un equipo evaluador que cuente como mínimo de cuatro personas y un máximo recomendado de nueve, incluyendo al evaluador líder certificado en CMMI por el SEI.

Las evaluación clase B está basada en la evaluación clase A. La evaluación clase B ayuda a una organización a comprender, con relativamente alto grado de confianza, el estado de los procesos relativos a CMMI. Generalmente se ejecuta una evaluación clase B cuando la organización necesita auto-evaluar sus procesos, con miras a una evaluación clase A para lograr el objetivo de la certificación. Esta clase de evaluación debe ser ejecutada por dos personas, incluyendo a un líder de CMMI y requiere mucho menos información que la evaluación clase A. Menos formal aún, de menor duración y con menos información requerida es la evaluación clase C que además es realizada por sólo una persona y tiene por objetivo evaluar pequeños aspectos de la organización que quieren apoyarse. (15)

1.4 Calidad del Software en la actualidad

1.4.1 Calidad del Software a nivel Internacional

Con la llegada de Internet y la tecnología, se hace cada vez más esencial estar al mando de la información, la competencia mundial cada vez se hace más pronunciada. En algunas ocasiones el éxito o

fracaso del proyecto no solamente está en la tecnología disponible y en el conocimiento que se tenga de ella, sino también en la forma en que el proyecto no lleva un control de calidad durante su desarrollo.

Es notable cómo los países desarrollados siguen llevando el liderazgo en cuanto a desarrollo de las nuevas tecnologías y como puede apreciarse el ranking de disponibilidad de las nuevas tecnologías lanzado por el Foro Económico Mundial (2009) para el año comprendido desde 2008 hasta 2009, los diez primero lugares, de 134 países incluidos, son ocupados por economías del primer mundo. (16)

Aún cuando Estados Unidos es el mayor productor y consumidor de servicios informáticos y de software, en las últimas dos décadas tres naciones han tenido un alto nivel de desarrollo en esta industria. Se trata de las tres famosas "I": India, Irlanda e Israel. Estos países proporcionan una gran variedad de servicios a la industria, inclusive realizan la parte más sofisticada del desarrollo del software.

La India es uno de los países con más empresas con certificación CMM (*Capability Maturity Model*) nivel cinco (se trata del nivel de capacidad y madurez en todas las áreas de servicios de desarrollo y prueba de software). Contar con una certificación de nivel tres al menos, es importante para participar en el mercado mundial de software. (17)

Latinoamérica

Los países de América Latina no han tenido políticas estatales hechas públicas con vistas a conocer sus pronunciamientos sobre la informatización social, excepto México que presenta un trabajo consecuente desde la década de los 90. Sin embargo, la actualidad está marcada por numerosos convenios y alianzas para el desarrollo de las nuevas tecnologías en el continente. Desde 1997, como uno de los primeros pasos, el gobierno federal de Brasil creó 20 centros de capacitación, Asesoría en certificación de calidad y contaba con 25 empresas certificadas ISO 9001. Ya en septiembre de 2006 Brasil contaba con 39 evaluaciones CMMI, según Marcelo (2007).

Hasta Septiembre de 2006, México y Chile contaban con menos de 10 evaluaciones CMMI y Costa Rica no reporta evaluaciones CMMI pero cuenta con 2 evaluaciones SW-CMM desde Julio de 2004 y 2 empresas de software certificadas con la norma ISO 9000. (18)

En Argentina el 90% de las 229 empresas que se acogieron a la Ley de Software, disponen de una certificación de calidad internacional, hecho muy auspicioso que revela la creciente necesidad de actuar en un mercado altamente competitivo, donde los precios no son la única variante para ganar mercados. (18)

Estas cifras mejorables antes mencionadas para alcanzar una certificación o cumplir con ciertos estatus de normas, no sólo es una meta, sino que las empresas latinoamericanas tienen la capacidad de competir con empresas de economía de primer mundo. Si se quiere contar con certificaciones de reconocimiento mundial, sólo basta comprometerse.

1.4.2 Calidad del Software en Cuba

Un universo de posibilidades abren las comunicaciones satelitales, la telefonía inalámbrica, Internet, la televisión digital, la computación, a un país como Cuba, a pesar del bloqueo económico, comercial y financiero con el que Estados Unidos, le impide a la Isla el acceso a su mercado y la obliga a invertir varias veces más recursos al tener que recurrir a mercados muy distantes.

A pesar de eso, basándose sobre todo en sus recursos humanos y optimizando sus recursos materiales y financieros, Cuba avanza en su informatización, priorizando el uso social y colectivo, de las TICS lo que da al país una connotación diferenciada al resto de los países del mundo en cuanto a Tecnologías de la Información se trata.

Como antecedente histórico María Vidal Ledo[5] concibe la primera aproximación de la estrategia cubana, fue presentada al Organismo encargado de regir el proceso de Informatización Social en 1999 lo cual sitúa a Cuba entre los primeros países en organizar un trabajo coherente en este sentido. Trabajando a partir de 5 estrategias maestras:

- Formación, preparación y perfeccionamiento de los Recursos Humanos.
- Red Telemática de la Salud.
- Seguridad Informática.

[5] **María Vidal Ledo:** Profesora Consultante y Auxiliar del Centro de Cibernética Aplicado a la Medicina del Instituto Superior de Ciencias Médicas de La Habana y de la Escuela Nacional de Salud Pública.

- Desarrollo Informático Territorial. Proyectos Sectoriales e Intersectoriales.
- Alianzas externas para el desarrollo. (18)

En el país existen empresas productoras de software, ejemplo Softel, la cual ofrece soluciones informáticas para el Sistema de Salud, con experiencia en diseño, implantación y gestión de estas soluciones. La mayoría de estas empresas no poseen una estrategia de Aseguramiento de la Calidad definida, los procesos son inestables e inmaduros, debido a que son empresas con poca experiencia en la producción del Software y no se encuentran certificadas con ningún modelo de calidad.

1.4.3 Calidad del Software en la UCI

La UCI es una universidad productiva, cuya misión es producir software y servicios informáticos a partir de la vinculación estudio – trabajo como modelo de formación. La Producción de Software y Servicios Informáticos se basa en la integración de los procesos de formación, investigación y producción en torno a una temática para convertirla en una rama productiva.

La universidad se encuentra inmersa desde octubre del 2008 en el proceso de mejora de la calidad, llevada a cabo por el centro de calidad (Calisoft) asistido por el SIE Center[6] el cual contribuye al desarrollo de la industria, incrementando su competitividad a través de la difusión, la mejora continua y el conocimiento en tecnologías de información.

El servicio de consultoría que SIE Center permite a la UCI:

- Revisar su estrategia de mejora de procesos de software, para asegurar que su organización está basada en procesos y con un programa de mejora continua alineado con sus objetivos de negocio.
- Establecer las bases y fundamentos para seguir mejorando sus procesos y fortalecer su cultura de calidad en el desarrollo de software.
- Alinear los procesos de desarrollo de software con los principios y requisitos del modelo CMMI, estableciendo planes de mejora con los que la organización oriente sus procesos hacia la consecución de sus metas.

[6] El SIE Center es un Centro Tecnológico de Monterrey que apoya al desarrollo de la industria del software por medio de los servicios de: Capacitación, Consultoría, Evaluación e Investigación.

En la UCI se han dado importantes pasos relacionados con la superación, como son diversos cursos que se han impartido con el fin de mejorar el trabajo:

- Curso de Ingeniería de Requisitos.
- Curso de postgrado dedicado a la capacitación de analistas, líderes de proyecto y otros agentes involucrados con el área de proceso de Administración de Requisitos.

También es válido mencionar que se ha logrado establecer una serie de documentos y plantillas que tributan a la organización interna de cada proyecto, donde se registran todas las actividades, artefactos y procesos realizados en un proyecto dando lugar al Expediente del Proyecto.

La estrategia seguida para el Programa de Mejoras de los procesos basado en el modelo CMMI (Capability Madurity Model Integration) que se lleva a cabo en la Universidad, incluye fases de capacitación en las áreas de procesos del Nivel 2 de CMMI.

1.5 Sistemas automatizados para la Gestión de No Conformidades

1.5.1 SIGNO – Software para la gestión de No Conformidades

Este sistema es desarrollado en Colombia en el año 2005. Es una herramienta informática de gestión que permite hacer el registro y la gestión total de las No Conformidades, tanto internas como externas, para un sistema de gestión de la calidad ISO 9000:2000 SIGNO permite registrar toda la información de una No Conformidad, asignar responsables (automáticamente o manualmente), reportar la respuesta, reportar el problema y sus causas, definir el plan de acción (acciones correctivas/preventivas), visualizar mediante semáforos la posible demora en ejecutar acciones, reportar la ejecución de acciones, reportar la auditoría realizada, cerrar No Conformidades y realizar consultas específicas por diferentes criterios. Adicionalmente permite todo el manejo de usuarios y motivos para hacer fácil adaptación a las necesidades de su organización.

Limitaciones

El costo, va más allá de las posibilidades, el cual asciende a 100 000 euros, además de no ser configurable a la hora de decidir qué pruebas hacer. Las pruebas en la Universidad tienen características específicas a la hora de hacer alguna revisión en dependencia del cronograma de trabajo y las necesidades del cliente.

1.5.2 KMKey Quality

KMKey Quality es un software de gestión de la calidad ideal para la implementación y mantenimiento de un sistema de gestión de calidad (SGC) de cualquier tipo: ISO 9001, ISO 14001, entre otros, o de una combinación de los mismos, facilitando su integración. Mediante KMKey Quality podrá gestionar y mantener la documentación del sistema, los registros, y los flujos de información propios que se generan en acciones como la gestión de No Conformidades, Acciones Correctivas/preventivas, Reclamaciones, Auditoría, Indicadores, Evaluaciones, entre otros. Todo ello adaptado al enfoque propio de su organización.

Dentro de sus funcionalidades de encuentra:

- No Conformidades y acciones correctivas/preventivas: Desde el mismo gestor de expedientes cualquier usuario autorizado puede introducir No Conformidades, así como planificar las correspondientes Acciones Correctivas o preventivas, repartiendo las tareas a cada uno de los responsables, y controlando plazos y acciones realizadas.

Limitaciones

El software KMKey Quality dentro de sus funcionalidades se encuentra la Gestión Documental, la gestión de expedientes específicos para Planes de Formación, planes de Auditoras o Planes de Mejora Continua, así como el seguimiento y Evaluación de proveedores o la Gestión y Control de Indicadores, a pesar de ser un software libre estas funcionalidades están todas correlacionadas, por lo que este software no se ajusta a las necesidades de la gestión de No Conformidades en el grupo de calidad.

Las herramientas descritas anteriormente no satisfacen las necesidades existentes en el Grupo de Calidad del departamento de Señales Digitales, de ahí surge la necesidad de crear el sistema informático

para la gestión de No Conformidades, la cual será desarrollada para facilitar el trabajo del grupo de calidad de la facultad, dentro de sus funcionalidades se encuentran la gestión de No Conformidades.

1.6 Proceso de Gestión de No Conformidad

El enunciado de la No Conformidad lleva a la organización y el análisis de la causa, la corrección y la acción correctiva.

El enunciado de la No Conformidad debería:

- Ser auto explicable y relacionado con el punto del sistema.
- No ser ambiguo con una correcta lingüística y tan concisa como sea posible.
- El enunciado de la No Conformidad no debe ser una repetición de la evidencia de la auditoría o usado en lugar de la evidencia de auditoría.

Una No Conformidad bien documentada tendrá tres partes:

- La evidencia de la auditoría.
- El requisito.
- El enunciado de la No Conformidad.

El formato para redactar las No Conformidades no tiene reglas fijas pero sí, en la redacción de la misma, se debe dejar claro lo siguiente:

- El problema (reportar lo que está mal).
- El área (dónde está lo que está mal).
- El requisito que incumple (referir concretamente el criterio de auditoría aplicable).

Si todas estas tres partes de la No Conformidad están bien documentadas, el auditado o, cualquier otra persona con conocimientos suficientes podrá ser capaz de leer y entender la No Conformidad.

Este proceso se realiza con la idea de detectar la mayor cantidad de errores en el menor tiempo posible y corregirlos de manera eficiente evitando la inserción de nuevos defectos, para ello se realizan las siguientes tareas:

- Entrada al Laboratorio.
- Distribución del trabajo entre los probadores.
- Realización de las Pruebas.
- Clasificación de las No Conformidades.
- Generación de reportes.

1.6.1 Entrada al Laboratorio

Una vez concluido el proceso inicial se informa al especialista la realización de la prueba y se procede a la entrega que se realizará en el servidor donde cada persona tiene los privilegios necesarios según la tarea que le corresponda. El proyecto informa por vía e-mail el momento justo de la entrega al Líder del Laboratorio de prueba. El cual contiene los elementos que se entregan, una estructura determinada con las especificaciones necesarias para el eficiente desarrollo de la prueba, además un lugar determinado para la colocación de las No Conformidades que se detecten, en caso de que se entreguen varios artefactos a la vez se debe especificar una serie de elementos a tener en cuenta a la hora de probar. El Líder del Laboratorio de prueba, verifica que estén todos los elementos previamente pactados y genera un documento de Aceptación o de No Conformidades de acuerdo a la evaluación de los elementos recibidos y el estado de éstos. De acuerdo a la valoración del Líder del Laboratorio de prueba si se decide que no se procederá a probar se indicaría en el mismo.

1.6.2 Distribución del trabajo entre los probadores

Cuando ya se tienen todos los elementos a revisar se pasa a la asignación del trabajo a los probadores, se elabora un informe con la distribución que se irá haciendo por nivel de complejidad y en dependencia de la cantidad de personas o turnos que se hayan decidido. Los probadores deberán conocer la distribución así como la etapa de detección y en caso de que la etapa de detección sea

posterior a las Pruebas Exploratorias (PE), debe existir la carpeta de Referencias para Calidad, donde se recogen las No Conformidades anteriormente detectadas.

1.6.3 Realización de las Pruebas

Los probadores comienzan a examinar los elementos que pueden ser documentación o aplicación, dentro de la documentación se encuentran los Diseños de caso de prueba (DCP), los Diseños de Caso de Uso (DCU), descripción de requisitos y los manuales. En la aplicación se encuentra el código fuente estructurado de manera funcional y lógica organizado arquitectónicamente. Estos elementos son revisados por el probador quien irá detectando los defectos a partir de la comparación con estándares previamente establecidos. Si la tapa de detección es mayor que las PE, entonces antes de detectar cualquier No Conformidad nueva, debe verificar que las anteriormente detectadas estén verdaderamente resultas.

1.6.4 Clasificación de las No Conformidades

La clasificaciones de las No Conformidades varían de la documentación a la aplicación aunque existen similitudes como los defectos categorizados como **Ortografía**, en esta clasificación se recoge todo lo referente a los errores ortográficos, gramaticales y de concordancia, éstas son siempre de tipo significativa; **Validación** se refiere a los mensajes que deben ser mostrados en la aplicación que no aparecen recogidos en los DCP y viceversa, incluye también los campos que no están correctamente validados según la entrada de datos que deben tener y son de tipo significativa; **Funcionalidad** en la documentación recoge los errores en los pasos del flujo central que guían lo que debe hacer el usuario y aparecen en la sección Requisitos a probar del DCP, en la aplicación concierne a los errores que no permiten que una funcionalidad determinada se ejecute, son de tipo significativa; **No correspondencia con los casos de uso** o los requisitos trata los errores cuando algún detalle no coincide entre DCU o Diseño de requerimientos (DR), aplicación y DCP, estas son de tipo significativa; la clasificación **Otros** recoge los errores relacionados con estándares de diseño o plantillas y son de tipo no significativas a excepción de los errores idiomáticos que son la mezcla de dos o más idiomas estos son de tipo significativa. Propias de la aplicación son **Opciones que no funcionan** que agrupa los errores de componentes, opciones o funciones que aparecen sin

desarrollar ninguna operación, son de tipo significativa; **Excepciones** por su parte recoge los errores de excepciones controladas o no controladas por el sistema que no se corresponden con los errores cometidos y son significativa; de tipo **Interfaz** son los errores de incumplimiento con los estándares para el diseño de interfaces y son de tipo significativa. El probador cuando está efectuando las pruebas puede hacer recomendaciones sobre los aspectos que estime conveniente, pero a su vez **Recomendación** es otra clasificación para No Conformidades que son poco relevantes y no entran en la clasificación de significativa o no significativa.

1.6.5 Generación de reportes

Una vez terminado el trabajo por parte de los probadores se sube al servidor para ser revisado por el Líder del laboratorio quien debe emitir un reporte con las No Conformidades clasificadas y contabilizadas que será recibido por el Equipo de desarrollo que al igual debe registrar el trabajo en el historial de las No Conformidades. Además al final del día se crea un reporte donde se controla todo el trabajo realizado durante la jornada.

1.7 Conclusiones parciales

En este capítulo se trató lo referente a la calidad de software lo cual creó la base teórica de la investigación. Teniendo en cuenta los modelos calidad se logrará un mejoramiento continúo en la manera de trabajar y un punto de referencia para evaluar los procesos actuales de la organización. Ningún Sistema informático para la gestión de No Conformidades existente se ajusta a las necesidades del grupo de calidad. Aunque se tomaron algunos elementos para el desarrollo de la solución que se pueden integrar los procesos asociados a la gestión de No Conformidades.

Capítulo 2: Herramientas y Tecnologías actuales

2.1 Introducción

Este capítulo recoge una descripción de las herramientas y tecnologías actuales relacionadas con el objeto de estudio teniendo en cuenta las necesidades vistas y las características del entorno donde se aplicará la solución propuesta. Se hace énfasis en las comparaciones de las metodologías, IDE de Desarrollo, Gestor de Base de Datos, Framework, para luego realizar la selección de los que se utilizarán en el desarrollo del Software.

2.2 Arquitectura establecida para los productos de GEYSED

Para los Productos de GEYSED se utiliza una arquitectura Modelo Vista Controlador para el trabajo en Web empleando el framework Symfony.

Symfony se basa en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por tres niveles:

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. (20)

El centro GEYSED también cuenta con una arquitectura de Escritorio la cual se basa en 3 Capas. La programación por capas es un estilo en el cual el objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos.

Las ventajas principales de este estilo son:

- Reutilización de capas.
- Facilita la estandarización.
- Dependencias se limitan a intra-capas.
- Contención de cambios a una o pocas capas.

En el diseño de sistemas informáticos actual se suele usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten). (21)

El diseño más utilizado actualmente es el diseño en tres niveles (o en tres capas):

Capa de presentación: es la que ve el usuario (también se le denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. (21)

Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del 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 y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación. (21)

Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. (21)

Todas estas capas pueden residir en un único ordenador, si bien lo más usual es que haya una multitud de ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si por el contrario, fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de negocio, y otra serie de ordenadores sobre los cuales corre la base de datos. (21)

Esta arquitectura se aplica en el centro GEYSED auxiliándose del framework Qt creator:

- No es más que un entorno de desarrollo integrado multiplataforma implementado específicamente para facilitar el proceso de codificación de aplicaciones haciendo uso del marco Qt.

2.3 Metodologías de Desarrollo

Una Metodología de desarrollo es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un producto de software. Una metodología representa el camino para desarrollar software de una manera sistémica. (36)

2.3.1 Rational Unified Process (RUP)

El Proceso Unificado RUP, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. Proporciona una aproximación disciplinada a la asignación de tareas y responsabilidades. RUP actúa como modelo y puede ser adaptado y extendido. (37)

El ciclo de vida de RUP se caracteriza por:

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura.

- **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son mini proyectos.

Algunas de las principales fases de RUP son:

- **Concepción o inicio:** Comprender los requisitos y determinar visión y alcance del proyecto.
- **Elaboración:** Asignar recursos, especificar las características y definir la arquitectura.
- **Construcción*:*** Implementación, construir el producto operacional.
- **Transición:** Hacerlo operativo para los usuarios, nivel correcto de calidad para entregar. En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

RUP cuenta con Flujos de trabajo descritos a continuación:

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe como el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

- **Implementación:** Define como se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba (Testeo):** Busca los defectos a lo largo del ciclo de vida.
- **Instalación:** Produce la liberación del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe como controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

RUP se basa mucho en la documentación, que aunque se ve afectada con los posibles cambios volátiles que los clientes soliciten en cuanto a funcionalidades del software, representa una gran ventaja, ya que gracias a su plan de desarrollo se pueden reconocer los problemas y fallos de forma temprana y corregirlos. Pero además, RUP no es un sistema con pasos firmemente establecidos, sino una metodología adaptable al contexto y necesidades.

2.3.2 Extreme Programming (XP)

Programación Extrema (Extreme Programming, XP). Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. (38)

Las principales características son:

- Centrado en resolver el problema lo más rápido posible.
- Cada miembro del equipo debe estar listo para enfrentar cualquier problema.
- El cliente se introduce en el equipo de desarrollo.

- Hago algo y lo pruebo.
- Termino todo y después integro.

Los obstáculos más comunes surgidos en proyectos XP son la “fantasía” de pretender que el cliente se quede en el sitio y la resistencia de muchos programadores a trabajar en pares.

2.3.3 OpenUP

OpenUP es una versión más ágil de lo que es RUP. Éste plantea que se debe tener un software ya funcional o lo que es lo mismo un proyecto ejecutable en un lapso de tiempo corto. Principalmente plantea que se debe utilizar solo los procesos que sean necesarios y en este caso se necesita de personas y profesionales que sean capaces de distinguir entre lo necesario y lo que no es necesario para que el proyecto no tenga errores y con eso evitar entregar al usuario final un producto de mala calidad, además plantea que no se deben utilizar demasiados artefactos y sobre todo que el proyecto debe acoplarse a las necesidades del usuario pudiendo ser modificado, mejorado y extendido.

2.3.4 Metodología Seleccionada

Anteriormente se analizaron algunas metodologías como XP, que a pesar de ser muy ágil en el proceso de desarrollo de software debido a que utiliza el mínimo de documentación, tiene como desventaja que el cliente debe de estar fuertemente ligado al equipo de desarrollo, en este caso el cliente no se encuentra dentro del equipo de desarrollo.

Otra metodología estudiada fue OpenUP, la cual es muy simple en el proceso de desarrollo de software generando poca documentación, por lo que no se adapta a las necesidades del equipo de desarrollo. Después de haber analizado estas tres metodologías de desarrollo de software se decidió utilizar RUP, ya que es la que más se adapta a las necesidades del equipo de desarrollo. Además constituye un marco de trabajo genérico que puede ser adaptado a una gran diversidad de sistemas de software independientemente del tamaño del proyecto, el tipo de organización y los diferentes niveles de aptitud.

2.4 Lenguaje de Modelado

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar (parte de) un diseño de software orientado a objetos. (32)

2.4.1 Unified Modeling Language (UML)

Debido a la selección de la metodología de desarrollo de software se hace necesario utilizar como Lenguaje Unificado de Modelado. UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Está pensado para poder aplicarse en cualquier medio que necesite capturar requerimientos y comportamientos del sistema que se desee construir. Ayuda a comprender y a mantener de una mejor forma un sistema basado en un área que el analista o desarrollador puede desconocer. UML surgió con el propósito de lograr una estandarización universal al crecido número de metodologías de desarrollo que habían estado apareciendo. UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. La decisión de utilizar UML como notación para el desarrollo del software se debe a que se ha convertido en un estándar que tiene las siguientes características:

- Permite modelar sistemas utilizando técnicas orientadas a objetos (OO).
- Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar. (33)

UML es independiente del proceso, aunque para utilizarlo óptimamente se debería emplear en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. (33)

2.5 Herramientas de modelado

Las herramientas de modelado de sistemas informáticos se emplean para la creación de modelos de sistemas que ya existen o que se desarrollarán. (34)

Las buenas herramientas de modelado cumplen con determinadas características:

- Permiten una visión descendente del sistema.
- Poseen componentes gráficos con algo de apoyo textual.
- El modelo resultado debe ser fácil de comprender.
- Poseen mínima redundancia.

2.5.1 Visual Paradigm

Visual Paradigm es una herramienta UML profesional fácil de usar. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (34)

Dentro de las principales características de la herramienta están:

- Soporte de UML versión 2.0.
- Disponibilidad de integrarse en los principales IDEs.
- Soporta una gama de lenguajes en la Generación de Código e Ingeniería Inversa en Java, C++, CORBA IDL, PHP, Ada y Python.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio.
- Diagramas de flujo de datos.

- Generador de informes para generación de documentación.
- Distribución automática de diagramas.
- Importación y exportación de ficheros.
- Editor de figuras.

Visual Paradigm ofrece un entorno amigable para el usuario, sugiere nuevos posibles componentes a utilizar, por lo que ya no es necesario localizarlos en la barra y así se crea fácilmente cualquier tipo de diagrama. Incluye gran variedad de estereotipos para la creación de diagramas de fácil entendimiento, los que organiza automáticamente.

2.5.2 Rational Rose

Rational Rose se ha convertido en una herramienta de modelado visual para el análisis y diseño de sistemas basados en objetos. Con el uso del lenguaje común de modelado UML, facilita el trabajo para el equipo de desarrollo y garantiza mayor eficiencia y calidad del trabajo. Permite generar código en diferentes lenguajes de programación partiendo de modelado UML. Además, hace posible efectuar la ingeniería inversa, es decir, obtener información del diseño de un software partiendo de su código. Rational Rose permite mantener la consistencia de los modelos del sistema software, realiza chequeo de la sintaxis UML y genera documentos automáticamente. (35)

2.5.3 Herramienta Seleccionada

Durante el estudio realizado sobre las herramientas de modelado se ha podido determinar que ambas poseen muchas ventajas a la hora de realizar el modelado de un software, sin embargo Rational Rose presenta como desventaja que es propietario, por lo que no sería recomendable utilizarlo para el desarrollo de la aplicación, pues en el departamento de señales digitales se aboga por la soberanía tecnológica y la utilización de software gratuitos. Visual Paradigm es la herramienta que se utilizará en la modelación de este producto debido a que es multiplataforma, además de ser una potente herramienta con licencia libre para uso de la comunidad de desarrolladores. Permite modelar UML, con 13 tipos diferentes de diagramas que pueden ser exportados como imágenes en formato JPG, PNG, entre otros.

Proporciona a los desarrolladores una plataforma con interfaz amigable que les permite diseñar un producto con calidad de forma muy rápida. Puede ser extendido, pues presenta soporte al diseño personalizado, permitiendo incorporar nuevas formas y notaciones, mediante el uso de imágenes o íconos importados.

2.6 Lenguaje de Programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. (22)

Los lenguajes de programación pueden clasificarse por diversos criterios siendo el más común, su nivel de semejanza con el lenguaje natural y su capacidad de manejo de niveles internos de la máquina.

Los principales tipos de lenguajes son tres:

- Lenguaje máquina.
- Lenguaje de bajo nivel.
- Lenguaje de alto nivel.

Lenguaje de máquina: Son aquéllos que están escritos directamente inteligible por la máquina (computadoras), ya que sus instrucciones son cadenas binarias (cadenas o series de caracteres de dígitos 0 y 1) que especifican una operación y las posiciones (dirección) de memoria implicadas en la operación se denominan instrucciones de máquina o código máquina. El código máquina es el conocido código binario.

Lenguajes de Bajo nivel: Son más fáciles de utilizar, dependen de la máquina en particular. El lenguaje de bajo nivel por excelencia es el ensamblador. Las instrucciones en lenguajes ensamblador son instrucciones conocidas como nemotécnicos. Por ejemplo, nemotécnicos típicos de operaciones aritméticas son: SUM, RES, DIV.

Lenguaje de Alto nivel: Son los más utilizados por los programadores. Están diseñados para que las personas escriban y entiendan los programas de un modo mucho más fácil que los lenguajes de máquina y ensambladores. Un programa escrito en alto nivel es independiente de la máquina, por lo que estos programas son portables o transportables. También pueden ejecutarse con pocas o ninguna modificación en diferentes tipos de computadoras.

2.6.1 Hypertext Preprocessor (PHP)

PHP es un lenguaje script para el desarrollo de páginas web dinámicas del lado del servidor, cuyos fragmentos de código se intercalan fácilmente en páginas HTML, debido a esto, y a que es de Open Source (código abierto), es el más popular y extendido en la web. (23)

PHP tiene muchas ventajas algunas de ellas son:

Multiplataforma: Inicialmente fue diseñado para entornos UNIX por lo que ofrece más prestaciones en este sistema operativo, pero es perfectamente compatible con Windows.

- Soporte para varios servidores Web.
- Mucha documentación (Ejemplos, manuales).
- Posee una sintaxis bastante clara.
- Fácil aprendizaje.
- Seguro.

PHP posee capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL. Además PHP está orientado a objetos. Por las potencialidades que ofrece, se decidió utilizar como lenguaje de programación PHP, debido a su amplia distribución está perfectamente soportado por una gran comunidad de desarrolladores. Como producto de código abierto goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparen rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.

2.7 Framework de Desarrollo

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (26)

2.7.1 Symfony

Symfony es un framework de PHP que simplifica el trabajo con páginas Web dado a la automatización de algunos patrones para resolver tareas frecuentes. Este framework tiene una estructura de código que fuerza al desarrollo de código más legible. También encapsula operaciones complejas en instrucciones sencillas. Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones Web. Este framework trabaja con el patrón de arquitectura Modelo Vista Controlador y la herramienta Propel para el mapeo de objetos a bases de datos. Es un framework sencillo para cualquier programador independientemente de que sea principiante o un experto en PHP. Symfony está programado en PHP5 y enfocado al desarrollo en el mismo lenguaje. Este framework hace un uso completo de los mecanismos de POO (Programación Orientada a Objetos) disponibles en PHP5. (27)

Symfony proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Algunas de las principales características son:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de Base de Datos.
- Utiliza programación orientada a objetos, de ahí que sea imprescindible PHP 5.

- Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Aunque utiliza MVC (Modelo vista controlador), tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de base de datos, el controlador frontal y las acciones.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.
- Es una estructura de librerías y clases para programar aplicaciones web. (27)

2.7.2 Kumbia

KumbiaPHP es un framework para aplicaciones web libre escrito en PHP5. Basado en las prácticas de desarrollo web. Kumbia fomenta la velocidad y eficiencia en la creación y mantenimiento de aplicaciones web. Intenta proporcionar facilidades para construir aplicaciones robustas para entornos comerciales. Además es un esfuerzo por producir un framework que ayude a reducir el tiempo de desarrollo de una aplicación web sin producir efectos sobre los programadores. (28)

Algunas de las principales características son:

- Mapeo Objeto Relacional (ORM) y Separación MVC.
- Soporte para AJAX.
- Generación de Formularios.
- Componentes Gráficos.
- URL amigables

2.7.3 Framework Seleccionado

Dada la necesidad de aumentar la productividad a la hora de implementar la solución resultante de la investigación, así como la importancia de garantizar un buen mantenimiento y seguridad a la misma, se ha decidido utilizar Symfony como framework de desarrollo, con la idea de liberar al programador de las

operaciones básicas y rudimentarias de la programación sobre el lenguaje seleccionado. A continuación se describen sus características más generales:

- Fácil de instalar y configurar en sistemas Windows, Mac y Linux
- Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server)
- Compatible solamente con PHP 5 desde hace años, para asegurar el mayor rendimiento y acceso a las características más avanzadas de PHP
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador sólo debe configurar aquello que no es convencional
- Preparado para aplicaciones empresariales, ya que se puede adaptar con facilidad a las políticas y arquitecturas propias de cada empresa u organización
- Flexible hasta cualquier límite y extensible mediante un completo mecanismo de plugins
- Publicado bajo licencia MIT de software libre y apoyado por una empresa comprometida con su desarrollo
- Traducido a más de 40 idiomas y fácilmente traducible a cualquier otro idioma.

2.8 IDEs de Desarrollo

Un entorno de desarrollo integrado o IDE es un programa informático compuesto por un conjunto de herramientas de programación. 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. (39)

2.8.1 Zend Studio

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene la interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados,

los configura para trabajar juntos en depuración. Zend Studio fue diseñado para usarse con el lenguaje PHP; sin embargo ofrece soporte básico para otros lenguajes Web, como HTML, Java script y XML. (40)

2.8.2 NetBeans

El IDE NetBeans una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java y soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). NetBeans IDE 6.8 Beta proporciona herramientas para la programación en el lenguaje PHP e incluye resaltado de sintaxis, auto-completado de código, documentación de PHP integrada, entre otras funcionalidades. Provee soporte para el framework Symfony y cuenta con un módulo de subversión para facilitar la manipulación de las versiones del código durante la implementación. (41)

2.8.3 IDE Seleccionado

Después de haber realizado esta comparación entre los IDE de Desarrollo se puede apreciar que ambos son potentes en el desarrollo de aplicaciones web, sin embargo el ZendStudio tiene como desventaja el rendimiento, que se va más allá de las posibilidades, requiere licencia de pago y no incluye editor visual HTML, por lo que se decidió utilizar como IDE de Desarrollo NetBeans ya que es un reconocido entorno de desarrollo integrado disponible en múltiples plataformas, permite desarrollar aplicaciones web con gran rapidez, además es un producto libre y gratuito sin restricciones de uso.

2.9 Gestor de Base de Datos

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. (29)

Ayuda a realizar las siguientes acciones:

- Definición de los datos.
- Mantenimiento de la integridad de los datos dentro de la base de datos.
- Control de la seguridad y privacidad de los datos.

- Manipulación de los datos.

2.9.1 PostGreeSQL

PostgreeSQL es un sistema de gestión de bases de datos objeto-relacional y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreeSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (30)

Algunas de las elementales características son:

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes, cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

2.9.2 MySQL

MySQL es un sistema de gestión de base de datos relacional. Opera en una arquitectura cliente/servidor, de tal manera que el servidor sólo tiene que enviarle una cadena de caracteres y esperar la devolución de los datos. MySQL ha pasado de ser una pequeña base de datos a una completa herramienta. Es una tecnología útil para la creación de bases de datos seguras al poseer un sistema de privilegios y contraseñas que es muy flexible y seguro,

que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de ellas a través de la Web está encriptado al conectarse con un servidor.

Las principales características de este gestor de bases de datos son las siguientes:

- Soporta gran cantidad de tipos de datos para las columnas. (31)
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP).
- Relativamente sencillo de añadir otro sistema de almacenamiento. Es útil si desea añadir una interfaz SQL para una base de datos propia.
- Gran portabilidad entre sistemas.

Algunas de las principales desventajas son:

- Carece de soporte para transacciones y subconsultas.
- El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí poseen esta característica.
- No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

2.9.3 Gestor de Base de Datos Seleccionado

Se decidió utilizar como Gestor de Base de Datos PostGreeSQL ya que posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos benchmarks se dice que ha llegado a soportar el triple de carga de lo que soporta MySQL). Además implementa el uso de subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz.

2.10 Servidor Apache

El servidor Apache es un software que está estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo.

Los módulos del Apache se pueden clasificar en tres categorías:

- **Módulos Base:** Módulo con las funciones básicas del Apache.
- **Módulos Multiproceso:** Son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones.
- **Módulos Adicionales:** Cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código. El resto de funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software. (42)

Algunas de las principales características son:

- Funciona sobre muchas plataformas.
- Módulos cargados dinámicamente.
- Php3 + Bases de datos.
- SSL: transacciones seguras.
- Alto desempeño.

2.11 Conclusiones parciales

A partir de la arquitectura planteada en el Centro de GEYSED y teniendo en cuenta las mejores opciones que cumplan con las políticas de desarrollo de software que se pretenden alcanzar tanto en Cuba como

en la Universidad de la Ciencias Informáticas, con el objetivo de lograr la soberanía e independencia tecnológica, se seleccionaron herramientas, lenguajes y tecnologías que permitirán darle solución a la problemática. Se empleará como metodología de desarrollo RUP, la cual utiliza como lenguaje de modelado UML. Para obtener un mejor diseño se hará uso de la herramienta de modelado Visual Paradigm. Además para lograr una mayor facilidad a la hora de desarrollar la herramienta se utilizará PHP como lenguaje de programación donde el entorno de desarrollo a utilizar será NetBeans, el cual será integrado al framework de desarrollo Symfony y PostGreeSQL como gestor de base de datos.

CAPÍTULO 3: Características del Sistema

3.1 Introducción

En el presente capítulo se describe el Modelo del Negocio y se enumeran las características básicas del sistema listando los requisitos funcionales y no funcionales. Se definen además los casos de uso, actores y trabajadores del sistema. Así como los patrones arquitectónicos, diagrama de diseño web, modelo de datos y modelo de despliegue.

3.2 Modelo del negocio

El modelo de negocio que se describe responde a un conjunto de actividades realizadas dentro de los procesos que se desarrollan en el grupo de calidad del Centro GEYSED. Con el objetivo de comprender la estructura y la dinámica en el cual se va a implantar el sistema y obtener una visión de la organización se realiza la modelación del negocio. El modelo de negocio permite definir los procesos, roles, responsabilidades del centro. Propicia además que los usuarios finales y los desarrolladores tengan un entendimiento común de la organización del sistema.

3.2.1 Actores del negocio

Se define como actor del negocio cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. En la siguiente tabla se definen los actores del negocio y su justificación. (43)

Actor del Negocio	Descripción
Solicitante	Jefe de proyecto que realiza la solicitud para la revisión de sus productos.

Tabla 1: Actor del negocio

3.2.2 Trabajadores del negocio

Un trabajador del negocio es una abstracción de una persona que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando entidades del negocio. (44)

Trabajadores del negocio	Descripción
Responsable de Calidad	Persona responsable de gestionar los proyectos que se prueban, registrar los usuarios, asignar el trabajo a los probadores y obtener reportes de la actividad del sistema.
Responsable del Proyecto	Persona que interactúa con el sistema cuando envía los artefactos a revisar, también cuando recibe las No Conformidades, y por último cuando desea ver algún reporte.
Probador	Persona responsable de realizar las pruebas a los artefactos y detectar las No Conformidades en los mismos.

Tabla 2: Descripción de los trabajadores del negocio

3.2.3 Diagrama de casos de uso del negocio

Describe los procesos de negocio de una empresa en términos de casos de uso y actores del negocio y sus relaciones, que se corresponden con los procesos del negocio y los clientes, respectivamente.

3.2.4 Descripción de los casos de uso del negocio

Un Caso de Uso del negocio representa a un proceso de negocio, por lo que se corresponde con una secuencia de acciones que producen un resultado observable para ciertos actores del negocio. Produce resultados deseables o mitigar los costos del negocio. (45)

3.2.5 Casos de Uso del Negocio

- Gestionar No Conformidades.

- Obtener Reportes.

Gestionar No Conformidades: El solicitante inicia el Caso de Uso al informar al Responsable de Calidad que los elementos a probar se encuentran en el lugar de la entrega. El Responsable de calidad informa al líder de los probadores que ya se encuentra el contenido de trabajo actualizado y este lo reparte entre los probadores, éstos entregan al líder las No Conformidades que encuentran, el Caso de Uso finaliza cuando el líder realiza la entrega de las No Conformidades detectadas al solicitante.
Obtener Reportes: El Caso de Uso comienza cuando el directivo decide solicitar un reporte, el asesor de calidad recoge toda la información necesaria y finaliza el Caso de Uso cuando el asesor de calidad emite el informe sobre el reporte solicitado.

3.2.6 Reglas del Negocio

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio. El proceso de especificación implica que hay que identificarlas dentro del negocio, evaluar si son relevantes dentro del campo de acción que se está modelando e implementando en la propuesta de solución. (46)

- El sistema automatizará las siguientes funcionalidades.
    - Control de la gestión de las No Conformidades encontradas al software.
    - Control de los reportes de las No Conformidades.
    - Control de los reportes de la cantidad de No Conformidades encontradas.
    - Control de la información.
- El sistema sólo será utilizado por el laboratorio de calidad de la facultad.
- Pueden acceder al sistema el Responsable de Calidad, Revisor Líder, Probadores, Responsable de Proyecto.
- En caso que un proyecto entre por segunda vez en fase de prueba debe especificar la versión.

3.3 Especificación de los requisitos de software

Los requisitos del software definen de forma precisa el producto de software que se va a construir pues no son más que características que va a poseer el producto. Se dividen en dos grupos, requerimientos funcionales y no funcionales.

3.3.1 Requerimientos funcionales del sistema

Según la IEEE *Standard Glossary of Software Engineering Terminology* 610, define un requerimiento como:

"Condición o capacidad necesaria para que un usuario resuelva su problema o alcanzar un objetivo que debe cumplirse o poseído por un sistema o componente del sistema para satisfacer un contrato, norma, especificación o formalmente impuesto en un documento". (47)

Los requerimientos funcionales son las características básicas del sistema que definen el comportamiento interno del mismo, en ellos se recogen las diferentes operaciones que debe realizar el producto.

- RF 1.1: Adicionar No Conformidades
- RF 1.2: Eliminar No Conformidades
- RF 1.3: Modificar No Conformidad
- RF 2.1: Adicionar Proyectos
- RF 2.2: Modificar Proyectos
- RF 2.3: Eliminar Proyecto
- RF 3.1: Obtener reporte de Tareas
- RF 4.1: Obtener reporte de No Conformidades
- RF 5.1: Entrar al sistema
- RF 6.1: Genera PDF
- RF 7.1: Reporte de Evaluación
- RF 8.1: Adicionar usuario
- RF 8.2: Modificar usuario
- RF 8.3: Eliminar usuario

- RF 9.1: Adicionar Tareas
- RF 9.2: Modificar Tareas
- RF 9.3: Eliminar Tareas
- RF 10.1: Adicionar Artefacto
- RF 10.2: Modificar Artefacto
- RF 10.3: Eliminar Artefacto
- RF 11.1: Adicionar Revisión
- RF 11.2: Modificar Revisión
- RF 11.3: Eliminar Revisión

3.3.2 Requerimientos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son características que hacen al producto, atractivo, usable, rápido y confiable. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir una vez que se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto. Las propiedades no funcionales, como cuán usable, seguro y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. (48)

- **Apariencia o interfaz externa:**
  - Debe ser una interfaz amigable, legible, interactiva, fácil de usar, profesional, clara, sencilla y debe mantener el mismo formato en todas las páginas.
  - Diseño para la resolución 1024 de ancho y el largo varía en dependencia de la cantidad de datos que contenga el sitio, aunque puede verse en otras resoluciones.
- **Usabilidad**
  - El sistema podrá ser usado por cualquier persona, que posea conocimientos básicos en el manejo de la computadora.
- **Rendimiento**
  - Rápido acceso de búsqueda a la información en tiempos relativamente cortos.

- El sistema deberá de ser lo más estable y confiable posible.

- **Soporte:**
  - Se requiere un navegador que interprete Java Script.
  - Versión de PHP 5.0 o superior.
  - Se requiere un servidor de bases de datos que soporte grandes volúmenes de información y velocidad de procesamiento.
- **Portabilidad y operatividad**
  - El software podrá ser usado bajo los sistemas operativos Windows 7 y Ubuntu 10.10.
- **Seguridad**
  - Establecer permisos para garantizar que sólo se acceda a la información quien tenga permiso.
  - Garantizar la protección ante acciones no autorizadas.
  - Verificación sobre acciones irreversibles (eliminación, modificación).
- **Confiabilidad**
  - Garantizar un tratamiento adecuado de excepciones y validaciones de las entradas de los usuarios.
  - Garantizar la recuperación del sistema ante fallos y errores.

3.4 Actores del Sistema

Cada trabajador del negocio (incluso si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema.

Actores del sistema	Justificación
Responsable de Calidad	Persona responsable de gestionar los proyectos que se prueban, registrar los usuarios, asignar el trabajo a los probadores y obtener reportes de la actividad del sistema.

Responsable de Proyecto	Persona que interactúa con el sistema cuando envía los artefactos a revisar, también cuando recibe las No Conformidades, y por último cuando desea ver algún reporte.
Probador	Persona responsable de realizar las pruebas a los artefactos y detectar las No Conformidades en los mismos.

Tabla 3: Trabajadores de negocio

3.4.1 Listado de Casos de Uso

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.

3.5 Patrones

Pareja de problema / solución con un nombre que codifica (estandariza) buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. (49)

Según Christopher Alexander[7] "Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma". Dentro de la rama informática los patrones se clasifican en:

- Patrones de Caso de Uso.
- Patrones Arquitectónicos.

3.5.1 Patrones de Caso de Uso

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requisitos reales, haciendo más fácil el trabajo con los sistemas, y mucho más simple su mantenimiento. Dado un contexto y un problema a resolver, estas técnicas han mostrado

[7] Christopher Alexander: Destacado Arquitecto reconocido internacionalmente por sus numerosos aportes en la Teoría de Patrones.

ser la solución adoptada en la comunidad del desarrollo de software. Se presentan a modo de herramientas que permiten resolver los problemas que se les planteen a los desarrolladores de una forma ágil y sistemática. Estos patrones se enfocan hacia el diseño y las técnicas utilizadas en modelos de alta calidad, y no en cómo modelar casos específicos. Utilizando estos patrones, arquitectos, analistas, ingenieros y gerentes pueden lograr mejores resultados de forma más rápida. (49)

Los patrones de Caso de Uso que se utilizarán son:

- Patrón CRUD (Completo).
- Patrón Múltiples Actores (Rol Común).

Patrón CRUD

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

Patrón Múltiples Actores

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

3.5.2 Patrones de Diseño y Patrones de Arquitectura

Dentro del marco de la arquitectura de software se insertan lógicamente los elementos relacionados al diseño del sistema en cuestión, en este caso, una aplicación para gestión de No Conformidades, y análogamente a los patrones de arquitectura, también existen los patrones de diseño.

Patrones GoF

Se utilizará el framework Symfony para el desarrollo del sistema informático, este framework utiliza una serie de patrones GOF como son:

En la categoría Creacionales:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal hay una llamada a sfContext: getInstance (). En una acción, el método getContext (), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony.

En la categoría Estructurales:

Decorador (Envoltorio): Añade funcionalidad a una clase, dinámicamente. El archivo layout.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación. (50)

Patrones GRASP

Creador: En la clase Actions se encuentran las acciones definidas para el sistema y se ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Actions es "creador" de dichas entidades.

Experto: Es uno de los más utilizados, puesto que Propel es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Alta Cohesión: Symfony permite asignar responsabilidades con una alta cohesión, por ejemplo la clase Actions tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por

diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

Controlador: Todas las peticiones web son manejadas por un solo controlador frontal (sf Actions), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Patrones de Arquitectura: En la última década cambió la visión que los desarrolladores tienen de los sistemas de software. Esta nueva visión se llamó "arquitectura". La arquitectura de software básicamente indica la estructura, funcionamiento e interacción entre las partes del software sistemas más grandes poseen una estructura y un comportamiento que los hace clasificables según su arquitectura. Dentro de los patrones de arquitectura se encuentran el patrón Modelo Vista Controlador (MVC) y el patrón modelo de tres capas. Para el desarrollo de este trabajo de diploma como fue anteriormente especificado se hará uso del framework Symfony, el cual está basado en el patrón Modelo Vista Controlador. MVC es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Utiliza las siguientes abstracciones:

- Modelo: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- Vista: Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- Controlador: Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón y pulsaciones de teclas. Los eventos son traducidos a solicitudes de servicio para el modelo o la vista.

3.6 Diseño

El diseño facilita la compresión del sistema que se implementará en el cual se recoge los diagramas correspondientes que contribuirán a la posterior implementación del proyecto. Es la parte fundamental en

la construcción del sistema que se pretende crear, pues es una abstracción de cómo quedara el producto final. A partir del diseño se obtendrá un modelo cercano a la versión final del software que se construye.

3.6.1 Diagrama de clases del diseño web

Mediante los diagramas de clases de diseño web es posible obtener una abstracción de cómo quedará la implementación del sistema, los diagramas especifican la relación entre los distintos elementos lo que hace que sea más fácil la implementación, facilitan y proveen una abstracción de lo que será el producto final.

3.7 Flujo de trabajo de implementación

Posterior a la fase de Análisis y Diseño se sucede la fase de Implementación del sistema. Dentro de esta fase existen una serie de artefactos que tienen como elementos de entrada para su confección, los desarrollados en la fase anterior, con el objetivo de ir conformando el proyecto como un sistema completo y con un acabado a la altura de los requerimientos previamente definidos.

3.7.1 Modelado de Datos

Colección de conceptos que sirven para describir la estructura de una base de datos, además de proporcionar los medios necesarios para conseguir la abstracción de los datos.

3.7.2 Modelo de Despliegue

El diagrama de despliegue ilustra la representación física de los elementos con los que deberá contar el sistema, los componentes de este diagrama son los nodos y las relaciones entre ellos, estas relaciones pueden ser de diferentes tipos dependiendo de lo que represente el nodo.

3.7.3 Ambiente de implementación

Para el correcto funcionamiento de la aplicación son necesarias una serie de características para el ambiente donde será implantada, a continuación se precisan las particularidades:

Para el trabajo con el sitio web se precisa una Computadora Personal (PC) servidor de 512 MB de memoria RAM y un procesador de 3.00GHZ, con Sistema Operativo Linux o Microsoft Windows Server 2003, o superior. En la PC servidor se necesita además la instalación del Gestor de Datos PostgreSQL en su versión 8.3 y el Servidor Web Apache en su versión 2.0.

La PC cliente para un buen rendimiento se aconseja tener como mínimo 1.3 GHZ de velocidad de procesamiento y 256 MB de RAM, podrán tener Sistema Operativo Windows 7 y Ubuntu 10.10 siempre y cuando se tenga instalado y configurado el navegador web Mozilla Firefox en la versión 3.0. Se recomienda para la utilización del sistema por parte de la administración, personas con conocimientos informáticos, los clientes del sistema por su parte deberán tener conocimientos básicos sobre ambientes Web así como de la importancia de las prueba dentro del proceso de desarrollo de software

3.8 Conclusiones parciales

En este capítulo se abordaron los elementos fundamentales de la implementación, donde juega un papel fundamental los artefactos generados durante la fase de diseño ya que sirven de entrada para la implementación del sistema, donde entra el diagrama de despliegue el cual muestra una visión de cómo quedará el sistema físicamente, el modelo de datos el cual es imprescindible para el desempeño de las consultas sobre la base de datos, los diagramas de clases de los principales requisitos funcionales de la aplicación, dando como resultado un sistema el cual realiza las funciones necesarias para el buen funcionamiento y obtener idea más detallada y precisa de lo que será el sistema que se llevará a cabo en la implementación.

Capítulo 4: Implementación y Prueba

4.1 Introducción

Durante este capítulo se realiza el flujo de trabajo de implementación, se analizan los principales artefactos como Modelo de Implementación que incluye componentes, subsistemas de implementación y diagramas de componentes. Se comienza a implementar el sistema en términos de componentes mediante las clases del diseño. Además cuando se haya terminado el software se realizarán las pruebas.

4.2 Modelo de Implementación

El Modelo de implementación representa la composición física de la implementación en términos de subsistemas de implementación y elementos de implementación. Describe como los elementos de diseño se implementan en componentes.

Se considera el artefacto más significativo del flujo de trabajo de Implementación, debido a la importancia que tiene para los desarrolladores comprender el funcionamiento del sistema desde el punto de vista de componentes y sus relaciones. Este modelo está conformado por el diagrama de componentes. (35)

4.2.1 Diagrama de Componentes

Dentro del Modelo de Implementación se encuentran los diagramas de componentes. Un componente es la parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces que proporciona la realización de los mismos. El diagrama de componentes describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros.

Código Fuente

Para obtener una versión funcional de la aplicación se deben implementar los componentes que se han definido, como resultado se obtienen archivos que contienen el código fuente de la aplicación. Es un

conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está descrito por completo su funcionamiento.

Estas instrucciones son escritas en un lenguaje de programación que consiste en un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

4.2.2 Estándares de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Usar técnica de codificación sólida y realizar buenas prácticas de programación con vistas a generar un código de alta calidad, es de gran importancia para la calidad del software y para obtener un buen rendimiento. (35)

Estilo de Codificación Utilizado

- Todas las etiquetas php deben ser completas (<?php?>)…no reducidas(<? ?>).
- Los bloques de código siempre deben estar encerrados por llaves (incluso si solo constan de una línea).
- Tamaño = 4 (espacios) para:
    - Declaraciones dentro de las clases.
    - Enunciado dentro de métodos y funciones.
    - Enunciados dentro de bloques de comandos.

4.2.3 Ejemplo de Código Fuente

La mayoría de los métodos implementados responden a funciones básicas de inserción, actualización, eliminación, que se vuelven realmente sencillos mediante el uso que hace Symphony de Propel para realizar el mapeo objeto-relacional. Genera la mayor parte del código de acceso a datos proporcionados una abstracción al punto que permite que los implementadores tengan que utilizar muy pocas consultas a la base de datos.

El método Buscar permite realizar una búsqueda de una No Conformidad según el número de la misma, mostrando así como resultado la No Conformidad que coincida con el número especificado.

4.3 Validación de datos

Symfony como framework empleado propone la validación de datos a través de la acción, que normalmente se corresponden con los parámetros de la petición. Symfony incluye un sistema de validación, utilizando métodos de la clase acción. Cuando un usuario realiza una petición a una acción, Symfony siempre busca primero en la clase llamado validateNombredelaAccion(), si lo encuentra ejecuta esa clase. El valor de retorno de esta validación determina el siguiente método que se ejecuta: si devuelve true, entonces se ejecuta el método executeNombredelaAccion (); si el resultado es false entonces se ejecuta handleErrorNombredelaAccion (). En el caso de que handleNombredelaAccion () no exista, Symfony busca un método genérico llamado handleError (). Si tampoco existe, simplemente devuelve el valor sfView: ERROR para producir la plantilla NombredelaAccionError.php.

4.4 Pruebas

La fase de pruebas es una de las más costosas del ciclo de vida del software. En sentido estricto, deben realizarse pruebas a todos los artefactos generados durante la construcción de un producto, lo que incluye especificaciones de requisitos, casos de uso, diagramas de diversos tipos y, por supuesto, el código fuente y el resto de productos que forman parte de la aplicación. En la investigación se realizaron Revisiones a los documentos Modelo de Negocio, Especificación de Requisitos entre otros, además Pruebas Funcionales, para éstas se diseñaron casos de pruebas a través de la descripción de los casos de usos. En el proceso de pruebas se detectaron varias No Conformidades, las cuales fueron corregidas y gestionadas correctamente.

El siguiente es un ejemplo de un Caso de Prueba realizado al Caso de Uso Gestionar No Conformidades.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central

Crear	EC 1.1: Insertar los datos correctamente	Se deben insertar los datos correctamente	http://signc/noconformidad/new
No Conformidad	EC 1.2: Existencia de algún campo requerido vacío o incorrecto.	Se deben dejar campos vacíos o introducir datos incorrectos.	http://signc/noconformidad/new

Tabla 4: Caso de Pruebas realizado al Caso de Uso "Gestionar No Conformidades"

A continuación un ejemplo de la matriz de datos del Caso de Uso Gestionar No Conformidades.

ID del escenario	Escenario	V1	V2	V3	V4	V5	V6	Respuesta del sistema	Resultado de la prueba
EC 1.1	Insertar los datos Correctamente	v	v	v	v	v	v	El sistema no muestra mensajes de error ya que no existen variables inválidas.	Satisfactorios.
EC 1.2	Existencia de algún campo requerido vacío o incorrecto.	v	v	v	v	i	v	El sistema muestra un mensaje de error ya que existe una variable inválida.	Satisfactorios.

Tabla 5: Matriz de datos del Caso de Uso "Gestionar No Conformidad"

4.5 Conclusiones parciales

En este capítulo se elaboró el diagrama de componente el cual permitió organizar los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizado. Utilizando el estilo de codificación se obtuvo un código más entendible y organizado. Se realizaron pruebas funcionales a la herramienta con la técnica de caja negra, basadas en diseños de Casos de Pruebas las cuales resultaron satisfactorias.

Conclusiones generales

- El desarrollo de la herramienta informática, creada a partir de la necesidad de automatizar los procesos que se llevan a cabo en el laboratorio de calidad del Centro GEYSED permitirá elevar la eficiencia en el proceso de revisión de los productos.
- La implementación de esta propuesta, posibilitará dar respuesta a las solicitudes de los clientes en el menor tiempo posible.
- El sistema fue implementado utilizando herramientas, lenguajes y tecnologías propuestas por el grupo de desarrollo, en su mayoría distribuidas bajo licencias de software libre en correspondencia con las políticas de la Universidad y del país.
- Se diseñaron casos de pruebas basados en los casos de uso del sistema, además de realizar pruebas de caja negra para verificar la eficacia de la aplicación.
- La herramienta es segura y posee un mecanismo de autenticación tipo usuario/contraseña.
- Tiene en cuenta el control de los usuarios, de manera que cada persona puede acceder, adicionar, modificar y/o eliminar información de acuerdo a lo que está autorizado a hacer.

Después de haber logrado los objetivos que se trazaron al principio de este trabajo y como producto informático se encuentra en su primera versión se plantean las siguientes recomendaciones:

- Implementar una nueva versión que incluya la automatización de las respuestas a las No Conformidades.
- Aplicar esta herramienta en otros laboratorios de calidad de otras facultades con el objetivo de generalizar la propuesta para elevar la eficiencia de estos procesos y evaluar las mejoras posibles para una nueva versión.
- Tener en cuenta este trabajo para el futuro desarrollo de sistemas similares.

Referencias Bibliográficas

1. **Informática, Departamento de Control de Calidad y Auditoría.** *Control de Calidad en los Sistemas.* 2000.

2. **Miranda, Sandor Luis y Romero, Arturo Luis.** *La calidad, su evolución histórica y algunos conceptos y términos asociados.* 2006.

3. *ISO 8042.* 1994.

4. **ISO.** *ISO 9000: "Sistema de Gestión de la Calidad".* 2001.

5. **Lovelle, Juan Manuel Cueva.** *Conferencia Calidad del Software.* España : s.n., 2000.

6. *Guía de los fundamentos de la dirección de proyectos (PMBOK).* 2004.

7. **Grosso, Luis Alberto.** [En línea] Marzo de 2006. http://gridtics.frm.utn.edu.ar.

8. **Monsalve, Luis.** [En línea] Agosto de 2002. http://www.inf.udec.cl/revista/edicion1/lmonsalve.htm.

9. *ISO.9000.* 2000.

10. **Lovelle, Luis Manuel Cueva.** [En línea] 2000. http://gidis.ing.unlpam.edu.ar.

11. **SCALONE, FERNANDA.** *ESTUDIO COMPARATIVO DE LOS MODELOS Y ESTANDARES DE CALIDAD DEL SOFTWARE.* Buenos Aires : s.n., 2006.

12. **Garcia, Romero.** [En línea] 2001. http://catarina.udlap.mx.

13. **ISO.** *ISO/IEC 90003: "Guía para la Aplicación de la ISO 9000 al Software de Computación".* 2004.

14. **CMMI.** *CMMI for Development, Version 1.2. .* 2006.

15. **Quiñones, Ernesto.** *Modelos de Calidad de Software y Software libre.* s.l. : Asociación Peruana de Software Libre.

16. eumed.net. [En línea] http://www.eumed.ne.

17. **Martín, Gabriela Seoane San y Ramírez, Rodolfo Cano.** ENTER@TEenlinea. [En línea] 2008. http://www.enterate.unam.mx.

18. **Hernández, Vismar Santos.** eumed.net. [En línea] http://www.eumed.net.

19. *ISO 9000.* 2000.

20. **Fabien Potencier, François Zaninotto.** Symfony la guía definitiva. [En línea] 30 de diciembre de 2008. [Citado el: 15 de Enero de 2011.] www.librosweb.es.

21. **Fanjul, Alejandro.** *Arquitectura en tres capas.* s.l. : E.T.S de Ingenieros de Telecomunicación(Universidad Pública de Navarra).

22. **Neumman, Jhon Luis Von.** Historia de la programacion. *http://ei.cs.vt.edu.* [En línea] 7 de Octubre de 2005. [Citado el: 20 de Marzo de 2011.]

23. Breve historia de PHP. [En línea] [Citado el: 20 de Enero de 2011.]

24. Programación Web: PHP + Base de Datos. [En línea] [Citado el: 20 de Enero de 2011.] <http:www.ecicomputacion.com.ar/php___bases_de_datos.html>.

25. *C++.* **Hopper, Grace.** s.l. : ENIAC Museum, 2005.

26. **Gutiérrez, Javier J.** *Que es un framework web.*

27. **Fabien Potencier, Francois Zaninotto.** Symfony la guía definitiva. s.l. : ISBN-13. , 2008.

28. **LouderTechnology.** *Kumbia Enterprise Framework.* 2009.

29. **Martin, Alicia Ramos.** *Operaciones con bases de datos ofimática y corporativas.* 2007.

30. **Martinez, Rafael.** PostgreSQL. [En línea] 2009-2011. [Citado el: 20 de Marzo de 2011.] http://www.postgresql.org.es/sobre_postgresql.

31. **Victoria.** Tecnología. [En línea] 25 de Febrero de 2009. [Citado el: 20 de Marzo de 2011.] http://www.definicionabc.com/tecnologia/mysql.php.

32. *Definición del lenguaje de Modelado Unificado (UML).* **Guillen, Paola Romero.** s.l. : Instituto Tecnológico de la laguna, 2010, Vol. Capitulo IV.

33. **Cornejo, Jose Enrique Gonzalez.** Que es UML. [En línea] 10 de Agosto de 2008. [Citado el: 20 de marzo de 2011.] http://www.docirs.cl/uml.htm.

34. **Ramirez, Jose Ramon Salazar.** Herramienta para modelado. [En línea] Noviembre de 2009. [Citado el: 20 de Marzo de 2011.] 101.

35. **Booch, Jacobson.** *El Proceso unificado de desarrollo de Software.* La habana : Felix Varela, 2004.

36. *Postgrado Metodología de la Investigación Sistema para la Gestión de Opiniones.* **Rodríguez, Yadany Betancourt.** Ciudad de la Habana : s.n., 2008.

37. *Ayuda Extendida del Rational Unificado Process.* **Rumbaugh, James.** 2003.

38. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* **Penadés, Patricio Letelier y Mª Carmen.** Valencia : Universidad Politécnica de Valencia.

39. **Amelot, Michele.** *El entorno de desarrollo IDE.* 2007.

40. Mapping Interactivo. Revista Internacional de Ciencias de la Tierra. *NORMAS SOBRE METADATOS (ISO 19115, ISO 19115-2, ISO 19139, ISO 15836).* [En línea] 2008. [Citado el: 20 de 10 de 2010.] http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=1455..

41. netbeans.org. [En línea] Oracle, 2011. [Citado el: 15 de Febrero de 2011.] http://netbeans.org/features/ide/index.html.

42. **Kabir, Mohammed J.** *La biblia de Servidr Apache.* 2003.

44. **Dapena, MSc. Martha D. Delgado.** Definición del modelo del negocio y del dominio utilizando. [En línea] [Citado el: 7 de mayo de 2011.] http://www.inf.udec.cl/~revista/ediciones/edicion8/Rbc.pdf.

45. *Material de estudios Sistema por computadora.* **Castellano, Escofet.** Páginas 22 a 27, 2005, Vol. Capítulo 3.

46. *Sistema Integral para la Gestión de Procesos.* **Jones, Melvin.** Argentina : s.n., 2002.

47. *Patrones "Gang of Four".* **informatica, Facultad de.** Madrid : Unidad Docente de Ingenieria del Software.

48. *Flujo de trabajo de requerimientos.* **Informaticas, Universidad de las Ciencias.** Ciudad de la Habana : s.n., 2007.

49. **Bicego, A., Krzanik, L. y Kuvaja, P.** *Tutorial: BOOTSTRAP – Assessment and Improvement Methodology.* San Francisco USA : s.n., June 1995.

50. **Orallo, Enrique Hernandez Orallo y Jose Hernandez.** *Programacion con el Estandar ISO y la biblioteca de Plantillas (STL).* 2001.

51. **Zibert van Gricken, Carolina.** carolina.terna.net. [En línea] 2004. [Citado el: 2010 de noviembre de 9.] http://carolina.terna.net/ingsw3/datos/Semana7_CMMI2.0.pdf.

52. **Hernández, Vismar Santos.** *Tecnología e internet.* Cuba : s.n.

53. **Vega Lebrún, Carlos, Rivera Prieto, Laura Susana y García Santillán, Arturo.** *MEJORES PRÁCTICAS PARA EL ESTABLECIMIENTO Y ASEGURAMIENTO DE LA CALIDAD DE SOFTWARE.* Boca del Río, Ver : s.n., 2008.

54. *Los sistemas de información geográficas SIG:Definición, características,estado actual y tendencias de desarrollo.* **Diaz Cisneros, Luís R y Candeaux, Rafael Duffatt.** Julio de 1994, Revista Internacional de Ciencias de la Tierra, págs. 100 -131.

55. **Santos Hernández, Vismar.** *Estudio sobre la industria del Software a nivel mundial. Caracterización en América Latina y Cuba.* La Habana, Cuba : s.n., 2009.

56. **Vázquez, Roberto Hugo.** *Taller de Calidad de Software.*

57. **Pressman, Roger S.** *Ingeniería de Software. Un Enfoque Práctico.* 2002.

59. **MINGUET MELIÁN, JESÚS M.** *LA CALIDAD DEL SOFTWARE Y SU MEDIDA.* Madrid : EDITORIAL CENTRO DE ESTUDIOS RAMON ARECES, S.A., 2003.