**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

# Facultad 3


TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

**Título**

Herramienta para la gestión del proceso de revisión de Aseguramiento de la Calidad a Proceso y Producto en el Grupo de Calidad de Software del Centro de Gobierno Electrónico.

**Autores**

Yarinet Tomacen Mustelier.

Orlando Haza Medina.

**Tutor**

Ing. Raúl Velázquez Alvarez.

**Co-Tutor**

Ing. Hernán Antonio Sánchez Guzmán.

**La Habana, Junio 2014**

**"Año 56 de la Revolución"**

*"Para ser exitoso no tienes que hacer cosas extraordinarias, haz cosas ordinarias extraordinariamente bien".*

*Ernesto Che Guevara*

## Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

| | |
|---|---|
| Yarinet Tomacen Mustelier | Orlando Haza Medina |
| Firma del Autor | Firma del Autor |
| Ing. Raúl Velázquez Alvarez | Ing. Hernán Antonio Sánchez Guzmán |
| Firma del Tutor | Firma del Co-Tutor |

## Datos de Contacto

**Autor:** Yarinet Tomacen Mustelier

Correo electrónico: ymustelier@estudiantes.uci.cu

**Autor:** Orlando Haza Medina

Correo electrónico: ohaza@estudiantes.uci.cu

**Tutor:** Raúl Velázquez Alvarez

Correo electrónico: rvelazquez@uci.cu

**Co-tutor:** Ing. Hernán Antonio Sánchez Guzmán

Correo electrónico: hasanchez@uci.cu

## Agradecimientos

*A mis padres y a mi hermana, por todo el sacrificio, las preocupaciones, la confianza, la dedicación, la comprensión y sobre todo muchas gracias por su amor incondicional.*

*A mi familia, por preocuparse y apoyarme siempre, esté donde esté.*

*A Gisell que aunque esté lejos, siempre me ha apoyado en todo y sin ella no hubiera sido posible la realización de esta tesis.*

*A la gente del barrio, por estar siempre pendientes de mí y desearme lo mejor.*

*A mi hermano Adrian que siempre ha sido un ejemplo para mí.*

*A mis amigos Andy y Hermes.*

*A mis amistades de la uci, en especial a Kiko porque sin su ayuda no hubiese llegado hasta aquí, a Osmail, a Yasel, al Yipi, a Alejandro, a Félix, a Saname, a Nicolau, a Pedro, a Ledian, a Luis, al Metra, a mi amiga Sarisleydi y a todos en general por estar siempre para mí, por soportarme y quererme como soy.*

*A la gente del grupo, los que ya no están aquí y los que como yo siguieron adelante, luchando por llegar hasta el final.*

*A mis tutores Raúl y Hernán, por ayudarme a lograr mis metas y por estar siempre ahí para mí.*

*A mi compañera de tesis y mi madrina Yarinet por ser la mejor compañera de tesis que he tenido, si te encuentras otra mejor que ella túmbala que es de cartón.*

*A Dailin Galafet, a mi ahijado Juan Enidio y a Nailin Pérez.*

*A los profes que me dieron clases y a los que en algún momento me ayudaron, incluyendo a mi tribunal, por apoyarme y por creer en nosotros.*

*A la UCI por todos los momentos que viví aquí y por la oportunidad de crecer como persona y como profesional.*

*A Yelienny sin la cual no hubiera sido posible estos agradecimientos.*

*A cada una de las personas que conocí durante estos cinco años.*

***Orlando Haza***

*Primero que todo agradecer a mi abuela Ana por su confianza en mí, por darme el apoyo, el cariño y por darlo todo por tal que hoy estuviera aquí, a ella le debo mi vida, mi carrera, mi todo.*

*A mis padres y hermanos que siempre me guiaron por el buen camino, por su amor, cariño, confianza y apoyo, por inculcarme la forma de hacer las cosas bien.*

*A mi tío Nelson, por su ayuda incondicional y dedicación que a pesar de su locura siempre ha estado al tanto de mí portándose como un padre, por motivarme a seguir adelante.*

*A mis abuelos Juanito, Miguel Ángel, Santiago y Butti por su preocupación y apoyo. De forma general a toda mi familia, que al ser tan grande no puedo mencionarlos a todos, gracias porque siempre me tienen presente y pidiéndole mucho a Dios para que yo alcanzara la meta.*

*A los tutores por su labor tan importante en el desarrollo de este trabajo, Raúl a pesar de ser "insoportable" se portó muy bien con nosotros guiándonos y ayudándonos en lo que fuera necesario y a Hernán por el esfuerzo, ayuda incondicional y sobre todo por su entrega total sin el cual no hubiera sido posible la realización de este trabajo. Muchísimas gracias.*

*Al tribunal de tesis por su preocupación y paciencia, especialmente a Mailen y Darián.*

*A mi novio Carlos Enrique por haber llegado a mi vida en el momento menos esperado pero sí en el que más lo necesitaba, por aceptarme como soy y darme su apoyo en todo. Gracias por tu amor y comprensión y sobre todo por la paciencia que has tenido conmigo.*

*A mis suegros Bárbara Álvarez y Carlos de Armas que siempre estuvieron apoyándome, dándome ánimos para llegar a este momento y por hacerme sentir como en mi casa.*

*A mi compañero de tesis que al principio, la verdad no sabía ni qué iba a hacer con él, pero después me demostró que sí podía y trabajó mucho en base para lograr este momento. A mi sobrina Luisa, la cual fue fundamental para el cumplimiento de esta tarea, se lo debemos a ella.*

*A mis amigas, sobre todo mi fiel amiga Mariosky por inculcar en mí valores de estudio y dedicación, por tenerme siempre muy presente y ha sido como un ejemplo a seguir. A Aymara Carla, Rosana, Yarissel, Yelienny, Yoelsy que me ayudaron en lo que hiciera falta y me apoyaron en cada momento y decisión tomada, en los buenos y malos ratos, siempre estuvieron ahí. A las niñas del apartamento: Lorena, Eimé, Marieni, Sari y a los compañeros de grupo por compartir juntos momentos de preocupación y alegría. Agradezco a todas las personas que de una forma u otra se preocuparon por el desarrollo de este trabajo, a todos aquellos que contribuyeron en mi formación como profesional y como ser humano. A todos ustedes muchas gracias.*

***Yarinet Tomacen***

## Dedicatoria

*A mi mamá, por su apoyo incondicional, por ayudarme a levantar después de cada tropiezo y por los consejos que tanto me han ayudado a seguir adelante cuando las cosas se tornaban imposibles.*
*A mi papá, por cada gota de sudor que ha derramado para que yo pueda cumplir mis sueños, por la confianza y el apoyo.*
*A mi hermana, por ser mi sustento, mi razón y por darme la fuerza para intentar ser un ejemplo para ella.*

***Orlando Haza***

*A mi abuela Ana por ser la razón de mi existir.*
*A mi tutor Hernán Antonio por ser un factor clave en el transcurso de mi carrera.*

***Yarinet Tomacen***

## Resumen

Las revisiones constituyen una de las actividades que se establecen para llevar a cabo el Control de la Calidad del software, su planificación, realización y seguimiento, inciden directamente en la calidad del producto final; por lo que el Centro de Gobierno Electrónico realiza el proceso de revisión de Aseguramiento de la Calidad a Proceso y Producto (PPQA, por sus siglas en inglés, Process and Product Quality Assurance) durante todo el ciclo de vida de los sistemas que desarrolla. La presente investigación está encaminada a facilitar la ejecución de las actividades del mismo a través de una herramienta que lo informatice.

Se realiza un estudio de algunas soluciones desarrolladas con fines similares, pero que no resuelven la problemática planteada, por lo que se realiza una selección de la metodología, herramientas y tecnologías a utilizar en el desarrollo de la propuesta de solución. Para la comprensión del negocio se modela el proceso y luego se definen los requisitos del sistema. Además se define la arquitectura, así como un conjunto de artefactos necesarios para lograr la implementación. Para evitar inconsistencias durante el desarrollo y cumplir con las expectativas del cliente se validan los requisitos y el diseño. Por último se realizan pruebas al sistema que verifican su correcto funcionamiento y se demuestra que la solución propuesta resuelve todas las necesidades del Centro de Gobierno Electrónico.

## Palabras Claves

Aseguramiento de la Calidad a Proceso y Producto, Proceso de revisión

# Tabla de contenido

**Índice de Figuras**

**Índice de tablas**

# Introducción

En los últimos años la industria del software ha experimentado un desarrollo vertiginoso. Esto se refleja en un constante crecimiento de la complejidad del software que se demanda en el mercado, existiendo una tendencia a que los clientes necesiten dichos productos en un tiempo relativamente corto y a un menor costo. Esto trae consigo que no siempre el software que se presenta en el mercado tenga la calidad esperada por el usuario. Sin embargo cada día aumenta el número de organizaciones que deciden mejorar la calidad de sus productos y servicios, apuntando a la satisfacción de sus clientes. La calidad es uno de los requisitos esenciales de cualquier producto y en la actualidad es un factor estratégico del cual dependen la mayor parte de las organizaciones.

Para lograr un producto que satisfaga en mayor medida los requisitos del cliente, son necesarias las revisiones de software, estas son un filtro para el proceso de desarrollo, se aplican en varios puntos durante la ingeniería del software y sirven para descubrir errores y defectos que luego pueden eliminarse. (1)

Las revisiones constituyen una de las actividades que se establecen para llevar a cabo el Control de la Calidad del software, su planificación, realización y seguimiento, inciden directamente en la calidad del producto final. La calidad del software incorpora el aseguramiento de la calidad, aplicándose en cada paso del proceso del software. El mismo es vital para lograr el cumplimiento de las prácticas y el establecimiento de la cultura de procesos en la organización, destacando la importancia de alcanzar la objetividad en la revisión, particularmente en el uso de un criterio definido, garantizando eventualmente, la independencia de las actividades que son revisadas. (2)

Como parte del aseguramiento de la calidad del software es altamente necesario realizar en determinadas fases del proceso de desarrollo del proyecto, evaluaciones de adherencia a procesos y productos que evidencien que se está cumpliendo con los procedimientos establecidos. (3)

Cuba no está ajena en cuanto a desarrollo de soluciones informáticas se trata, la Universidad de las Ciencias Informáticas (UCI) es uno de los máximos exponentes en el desarrollo de la industria del software, siendo uno de sus objetivos, informatizar la sociedad cubana; factor importante para la economía del país.

La UCI es una universidad productiva, la cual pretende servir de soporte para la industria cubana de la informática,(4) para ello se trabaja en la producción de software y servicios

informáticos, cuyo proceso de desarrollo debe cumplir con cada una de las prácticas genéricas y específicas establecidas para el área de Aseguramiento de la Calidad a Proceso y Producto (PPQA). En la universidad se encuentra institucionalizado el nivel dos del Modelo Integrado de Capacidad y Madurez (CMMI, del inglés Capability Maturity Model Integration), el cual propone una administración disciplinada de los proyectos donde se establezcan y sigan políticas organizacionales. De ahí que se ha hecho necesario en los proyectos de la Universidad, realizar revisiones al proceso de desarrollo de software para identificar los defectos que se generan durante todo el ciclo de vida de los mismos.

Uno de los centros de desarrollo de la Universidad es el Centro de Gobierno Electrónico (CEGEL), dedicado al desarrollo de proyectos y soluciones informáticas para la gestión de las áreas de gobierno. Dentro del mismo se encuentra el Grupo de Calidad de Software, el cual tiene el objetivo de asegurar la calidad tanto en el proceso de desarrollo como en el producto obtenido, logrando que el resultado final cumpla las expectativas y necesidades del cliente.

El proceso de revisiones de software realizado en CEGEL presenta una serie de deficiencias que a continuación se describen.

- ✓ El proceso de revisión de PPQA que se lleva a cabo en el Centro de Gobierno Electrónico se realiza de manera desorganizada desde la notificación de revisión hasta la generación de reportes. En él se encuentran involucrados los miembros del Grupo de Calidad y el equipo de proyecto al cual se le hará la revisión. Durante este proceso el Grupo de Calidad redacta todas las No Conformidades (NC) que son encontradas durante la revisión obteniendo como resultado el Registro de Evaluaciones del Proyecto, para lo cual se tienen como máximo 3 días. En este periodo de tiempo el equipo de proyecto desconoce el estado de la revisión y por tanto no tiene la posibilidad de ver las NC provocando así demora en la resolución de las mismas y por consiguiente el equipo revisor no puede ver las respuestas de las mismas hasta que el proyecto no envíe el Registro de Evaluaciones con los datos actualizados y se realice la actividad de seguimiento donde se comprueba si fueron cerradas. Además cuando la cantidad de revisiones aumenta, al Asesor de la Calidad se le dificulta la realización de reportes de los resultados obtenidos.
- ✓ Para la revisión el equipo revisor hace uso de una lista de chequeo pero de manera separada, por lo que los resultados no se encuentran centralizados y al

final de la revisión hay que unirlos todos en un solo documento provocando en varias ocasiones la duplicidad o ausencia de información.

- ✓ La citación para las revisiones se realiza a través del correo electrónico y en la mayoría de los casos se hace de manera personal, por lo que muchas no se tienen registradas, esto provoca que no se tenga el control del número de revisiones efectuadas durante un periodo determinado.
- ✓ La generación de los gráficos para representar los resultados de los indicadores *Adherencia a criterios de evaluación y Distribución de impacto y tipo de no conformidades* se realiza a través de un Excel que se llena de forma manual, así como el Plan Anual de Revisiones, los cuales en ocasiones suelen perderse provocando pérdida de información y afectando el control sobre la planificación de las revisiones.
- ✓ El Grupo de Calidad no evalúa el servicio que brinda, por lo que carece de la posibilidad de conocer el grado de satisfacción de los clientes, los cuales aportan elementos fundamentales a tener en cuenta para ofrecer un mejor servicio.

Teniendo en cuenta la situación problemática se identificó el siguiente **problema a resolver:** ¿Cómo contribuir a la mejora del proceso de revisión de PPQA en el Grupo de Calidad del Centro de Gobierno Electrónico de manera que se facilite el control de las actividades del mismo?

Para la solución del problema planteado se define como **objeto de estudio:** el proceso de Aseguramiento de la Calidad de Software, tomando como **campo de acción:** la revisión de adherencia a proceso y a producto.

Se traza como **objetivo general:** desarrollar una herramienta para la gestión del proceso de revisión de PPQA en el Grupo de Calidad del Centro CEGEL que facilite el control de las actividades que en este se llevan a cabo.

Para guiar el desarrollo de este trabajo se plantea como **idea a defender:** con el desarrollo de la herramienta para la gestión del proceso de revisión de PPQA en el Grupo de Calidad del Centro CEGEL se contribuye a facilitar el control de las actividades que en este se llevan a cabo.

Para lograr el objetivo trazado, se han derivado un conjunto de **objetivos específicos** los cuales se listan a continuación:

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Desarrollar la solución propuesta.
- ✓ Validar los resultados obtenidos.

Para dar cumplimiento al objetivo general se plantearon las siguientes **tareas de investigación:**

- ✓ Caracterización de las soluciones informáticas que implementan el proceso de revisiones.
- ✓ Selección de la metodología de desarrollo de software, tecnologías y herramientas necesarias para el desarrollo del sistema que se propone.
- ✓ Realización de la modelación del negocio describiendo la propuesta de solución de manera que facilite la captura de los requisitos que debe tener el sistema.
- ✓ Realización de la captura de los requisitos para obtener las especificaciones funcionales y no funcionales que debe cumplir el sistema.
- ✓ Elaboración del análisis y diseño del sistema.
- ✓ Implementación de los requisitos definidos para darle solución al problema planteado.
- ✓ Validación de la investigación realizada para demostrar que la propuesta de solución resuelve el problema planteado.

Los métodos científicos utilizados para realizar la investigación se citan a continuación:

- ✓ **Teóricos**
    - ✓ **Analítico-Sintético**: se realiza la búsqueda y análisis de los conceptos y documentos necesarios que permitieron la obtención de los elementos para la comprensión del objeto de estudio. Además de un estudio de las diferentes herramientas para la gestión del proceso de revisiones de PPQA, determinando su influencia en el problema actual de la investigación con el objetivo de optimizar la implementación de la herramienta que el presente trabajo propone.
    - ✓ **Modelación:** se utiliza en el diseño del sistema mediante el esbozo de los diferentes diagramas definidos en la metodología escogida, permitiendo así un mejor entendimiento de las funcionalidades de la aplicación.
- ✓ **Empíricos**
    - ✓ **Entrevistas:** permite obtener conocimiento cualitativo del fenómeno, además de la obtención de puntos clave que ponen al descubierto cómo se desarrolla actualmente el proceso de revisión en el Grupo de Calidad del Centro CEGEL a partir del intercambio con el cliente para recopilar información sobre el negocio, las necesidades existentes y los requisitos

que debe cumplirse en el desarrollo del software, ayudando a adquirir experiencias personales del entrevistado.

- ✓ **Medición:** permite obtener a partir del uso de las métricas para validar los requisitos y el diseño la información numérica que se necesita para comprobar la correcta elaboración de los mismos. Así como obtener resultados estadísticos de la realización de las pruebas al sistema.

EL presente trabajo de diploma queda estructurado por 3 capítulos.

**Capítulo 1: Fundamentación teórica**

En el capítulo se exponen los principales conceptos asociados al dominio del problema, así como la metodología y las principales herramientas que serán usadas en el desarrollo de la aplicación.

**Capítulo 2: Propuesta de solución**

El capítulo se centra en definir los temas de negocio, requisitos, análisis y diseño de la solución, donde se plantean las características del sistema para su posterior implementación, se realiza la modelación de la herramienta, la cual incluye la realización de los diagramas, así como la generación de los diferentes artefactos y su documentación correspondiente. Además se realiza la validación de los requisitos y del diseño mediante las métricas seleccionadas.

**Capítulo 3: Implementación y prueba**

EL capítulo aborda acerca de la implementación de la herramienta y los estándares de codificación a utilizar para obtener un código legible, así como toda la documentación asociada al flujo de trabajo. Además se plasma la validación de la solución propuesta mediante las pruebas y diseño de casos de prueba para asegurar el correcto funcionamiento de la aplicación, así como la validación de las variables de la investigación que demuestran la solución a la problemática planteada.

# Capítulo 1: Fundamentación Teórica

## 1.1. Introducción

En el presente capítulo se exponen los conceptos fundamentales utilizados a lo largo de la investigación. Además se realiza la selección de la metodología, entorno de desarrollo, gestor de base de datos, técnicas y herramientas a utilizar en la implementación.

## 1.2. Conceptos Fundamentales

**Calidad de software**

La calidad del software es una preocupación a la que se dedican muchos esfuerzos. Sin embargo, el software casi nunca es perfecto. Todo proyecto tiene como objetivo producir software de la mejor calidad posible de manera que cumpla con las expectativas de los usuarios. La ISO 8402 (UNE 66-001-92) define la calidad del software como:
*"El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas".* (5)

Según Roger S. Pressman, la calidad del software es la: *"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente"*. (1)

Es importante destacar que en estos conceptos falta agregar el factor tiempo que es muy importante a la hora de desarrollar un producto de software porque hay que tener siempre presente los compromisos de entrega. Por lo tanto el primer compromiso que se debe de trazar un equipo de desarrollo de software es producir un software con la calidad requerida en el tiempo establecido.

En el proceso de desarrollo de software, para muchos desarrolladores se torna una tarea engorrosa evaluar la calidad del producto en cada subproceso, por lo que esta actividad es llevada a cabo una vez terminado el producto, detectando un sin número de errores que podrían haber sido resueltos con anterioridad. Por esta razón se han definido un conjunto de actividades para garantizar la calidad del producto durante todo el proceso de desarrollo, conocido como el Aseguramiento de la calidad de software.

**Aseguramiento de la calidad de software**

Según la Norma ISO 9000:2000, el aseguramiento de la calidad es la parte de la gestión de la calidad orientada a proporcionar confianza en que se cumplirán los requisitos de calidad. (6)

Específicamente para el desarrollo de software el autor también expone que el Aseguramiento de calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza de que el software satisfaga los requisitos dados de calidad.

Hay tres aspectos muy importantes con relación al Aseguramiento de la calidad del software: (7)

1. La calidad no se puede probar, se construye.
2. El aseguramiento de la calidad del software no es una tarea que se realiza en una fase particular del ciclo de vida de desarrollo.
3. Las actividades asociadas con el Aseguramiento de la calidad del software deben ser realizadas por personas que no estén directamente involucradas en el esfuerzo de desarrollo.

Estas acciones no son suficientes para un correcto aseguramiento de la calidad, debe tener como base la capacitación del personal técnico en procesos de desarrollo, mantenimiento y herramientas, y el compromiso de todos los trabajadores para desarrollar un proceso de aseguramiento con un alto nivel de calidad. Los elementos anteriores no garantizan el desarrollo o mantenimiento de una aplicación libre de defectos; sin embargo, reducen notablemente el riesgo de que ocurran.

**Revisiones**

Son un filtro para el proceso de ingeniería del software, se aplican en varios momentos del desarrollo y sirven para detectar defectos que puedan así ser eliminados. También para purificar las actividades de ingeniería del software. (8) Una revisión se puede definir como una reunión formal, en la que se presentan el estado actual de los resultados de un proyecto. Esto puede ser a un usuario, cliente u otro tipo de persona interesada.

Con las revisiones se consigue que el peso de la evaluación técnica no recaiga sobre las mismas personas involucradas en la producción del software, que por la posición que ocupan no pueden ser totalmente objetivas, sino en otras personas técnicamente competentes y objetivas.

Uno de los objetivos fundamentales de las revisiones técnicas es ofrecer información fiable acerca de los aspectos técnicos del proceso de desarrollo de software, los costos y la programación del trabajo. Mediante el uso de la misma se pueden tomar decisiones adecuadas y dirigir con éxito el proyecto.(9)

Dentro del proceso de revisión se aplica una lista de chequeo con lo cual el grupo de trabajo identifica el avance que se ha logrado en el desarrollo.

**Lista de Chequeo**

Se entiende por lista de chequeo o de verificación a un listado de preguntas, en forma de cuestionario que sirve para verificar el grado de cumplimiento de determinadas reglas establecidas a priori con un fin determinado. Estas son un conjunto de directrices que deben tenerse presente desde el inicio de un proyecto. (10)

Constituye un instrumento de muy fácil utilización para las revisiones y debe estar disponible para la ejecución de sus actividades. Las mismas permiten mejorar la planificación y da la oportunidad de hacer una recopilación de los datos que surgen durante este proceso.

Mediante el uso de estas se detectan errores en el software, más conocidos como no conformidades, las cuales se tratan desde su hallazgo hasta su solución.

**Gestión de No conformidades**

Según la norma ISO 9000:2005 una No Conformidad es un incumplimiento de un requisito del sistema, sea este especificado o no. Se conoce como requisito una necesidad o expectativa establecida, generalmente explícita u obligatoria. Dicha norma define un Producto como el resultado de un proceso. (11)

A partir de estos dos conceptos dados por la ISO se deduce que una no conformidad para el producto es un resultado previsto de los procesos que no cumplen los requisitos. (12)

Según norma ISO 9000:00 un proceso es el conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados.

Luego de analizar los conceptos asociados a no conformidad, proceso y producto los autores llegan a la conclusión de que una no conformidad asociada al proceso y producto (en este caso no se refiere al sistema) puede verse como las deficiencias encontradas en el proceso que se sigue, ya sea, en las actividades o tareas definidas

que tienen que ser realizadas para construir un producto de software de alta calidad; desde la concepción inicial, hasta la entrega y el retiro final del sistema, así como los productos de trabajo o artefactos que se tienen que generar.

La gestión de las no conformidades es esencial para la mejora continua de la calidad. Sin embargo no es suficiente corregir eventuales desvíos. Es preciso adoptar el enfoque de eliminar las causas de las no conformidades para evitar que vuelvan a suceder.

A partir del análisis de los diferentes conceptos referentes al negocio del proceso de revisiones que se realiza en el Grupo de Calidad del Centro CEGEL, es necesario conocer la situación actual del proceso para contribuir así a la mejora del mismo.

## 1.3. Escenario actual del proceso

En el proceso de revisión de PPQA que se lleva a cabo en el Centro de Gobierno Electrónico se encuentran involucrados los miembros del Grupo de Calidad y el equipo de proyecto al cual se le hará la revisión. El mismo comienza cuando el Asesor de la Calidad realiza la planificación anual de revisiones de adherencia a proceso y producto definiendo cuándo le corresponde a cada proyecto la revisión y para ello se le envía una notificación de revisión, la cual se hace por medio del correo electrónico o de forma personal.

Durante este proceso el Grupo de Calidad redacta todas las No Conformidades que son encontradas en la revisión a través de la aplicación de una lista de chequeo (LCH). Este proceso termina con el análisis de los resultados y los reportes de tendencias.

Para facilitar el desarrollo de esta actividad se han desarrollado sistemas que reúnen algunas características del proceso de revisiones, a continuación se realiza un estudio de las soluciones informáticas dedicadas al tema en cuestión.

## 1.4. Valoración del estado del arte

**GESPRO (Ecosistema de software para la Dirección Integrada de Proyectos)**
Combina el uso de una solución informática para la dirección integrada de proyectos y un sistema de formación especializada en gestión de proyectos.

Este sistema permite la gestión de listas de chequeo asociadas a las revisiones de los proyecto, gestiona las no conformidades y seguimiento de las mismas, permite asignar tareas, así como notificarlas, además permite realizar las planificaciones de la calidad. (13)

**Sistema para la Gestión de Revisiones (SIGRE)**

En el Grupo Calidad del CESIM se creó un sistema para la gestión de revisiones (SIGRE), el mismo permite conocer la fecha de realización de las revisiones a los distintos proyectos pertenecientes al CESIM, el estado actual de las no conformidades, los plazos de corrección de las mismas, así como las listas de chequeo utilizadas en cada caso. Además la informatización de este proceso permite realizar de forma sencilla y rápida acciones de modificación, inserción y eliminación de datos generados como resultado de las revisiones realizadas. (12)

**Herramienta para certificar la madurez de los procesos que requiere CMMI 2**

En la Facultad de Informática de la Universidad Nacional de la Plata, se creó una herramienta para certificar la madurez de los procesos que requiere CMMI nivel 2, ayudando para ello al personal que realiza el aseguramiento de la calidad. El objetivo principal de la misma es llevar el seguimiento de las auditorías y del registro y tratamiento de las no conformidades requeridos por el Área de PPQA. Entre las funcionalidades que presenta se encuentra generar y escalar no conformidades, recordar al usuario las auditorías y no conformidades registradas, crear proyectos a partir de plantillas predefinidas, agregar y registrar auditorías, configuración general para el seguimiento y control de cumplimiento de auditorías y verificar el cumplimiento de las tareas de una auditoría. (14)

Esta Universidad creó un proceso adaptable a las particularidades de su estructura, por lo que el mismo no garantiza el cumplimiento de las políticas definidas en la UCI, además pertenece al sector privado. Por estas razones este sistema no constituye una solución factible respecto a las necesidades de la Universidad de las Ciencias Informáticas. (12)

De forma general los sistemas anteriormente analizados presentan las siguientes limitaciones:

- ✓ No muestran los resultados asociados a los indicadores Adherencia a criterios de evaluación y Distribución de impacto y tipo de no conformidades.
- ✓ No evalúan el grado de implementación de los procesos institucionalizados por la universidad.
- ✓ No evalúan el servicio brindado, por lo que no se puede conocer el grado de satisfacción de los clientes en los proyectos evaluados.

- ✓ No posibilitan obtener los artefactos del Expediente de evaluación, tales como: las Minutas de reunión, Informe de evaluación y Registro de evaluaciones del proyecto.

Estos sistemas no cumplen a cabalidad con las expectativas del Grupo de Calidad del Centro CEGEL ya que no presentan todas las funcionalidades necesarias para llevar a cabo el proceso de revisiones de manera general. Para darle solución a la problemática planteada se decide implementar una herramienta para la gestión del proceso de revisión de PPQA en el Grupo de Calidad de Software del Centro de Gobierno Electrónico.

## 1.5. Metodología, Herramientas y Tecnologías

### 1.5.1. Selección de la metodología

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente, logrando así que cumpla con los requisitos planteados respondiendo a necesidades y condiciones del equipo de desarrollo y del cliente. Existe una gran variedad de metodologías para la creación del software, las cuales se clasifican en dos grandes grupos:

- ✓ Las orientadas al control de los procesos estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas son llamadas Metodologías Pesadas. (16)
- ✓ Las orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales al cliente en intervalos cortos de tiempo para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas Metodologías ágiles. (15)

Después de analizar los dos grupos, se definió el uso de una metodología pesada escogiéndose el Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés, Rational Unified Process) debido a que el negocio es complejo y es necesario generar la documentación correspondiente para su mejor entendimiento, así como para su posterior mantenimiento una vez concluida la aplicación. Otro elemento que influye en la selección de RUP es que el cliente no forma parte del equipo de desarrollo y cuenta con poco tiempo para interactuar con el mismo, además que proporciona una forma controlada de trabajar, guiando el orden de las actividades y tareas de los desarrolladores individuales y del equipo como un todo, siendo una metodología adaptable al contexto y necesidades.

**Proceso Unificado de Desarrollo**

Esta metodología se encarga de asignar tareas y responsabilidades en una empresa de desarrollo, se define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto, está basada en componentes e interfaces bien definidas. (17)

RUP es una metodología que se caracteriza por:

- ✓ Dirigido por casos de uso, los cuales constituyen un elemento integrador y una guía del trabajo para orientar al proyecto.
- ✓ Centrada en la arquitectura, relacionando la toma de decisiones que indican cómo debe ser construido el sistema y en qué orden.
- ✓ Proceso iterativo e incremental, donde se centra en el perfeccionamiento gradual de un sistema a través de múltiples ciclos de análisis, diseño y construcción agregándole funcionalidad al sistema en los ciclos productivos, una creación incremental se compone de varios ciclos de desarrollo iterativo. (16)

Además divide el proyecto en mini proyectos donde los casos de usos y la arquitectura cumplen sus objetivos de manera más depurada. Busca detectar defectos en las fases iniciales e intenta reducir el número de cambios tanto como sea posible ya que estos incrementan notablemente el costo.

El ciclo de vida de RUP se divide en 4 fases:

- ✓ **Inicio:** esta fase se enfoca hacia la comprensión del problema y la tecnología haciendo mayor énfasis en actividades de modelado del negocio y de requisitos.
- ✓ **Elaboración:** se orienta al establecimiento y desarrollo de la arquitectura óptima.
- ✓ **Construcción:** en esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- ✓ **Transición:** se pretende garantizar un producto disponible para la comunidad de usuarios**.**

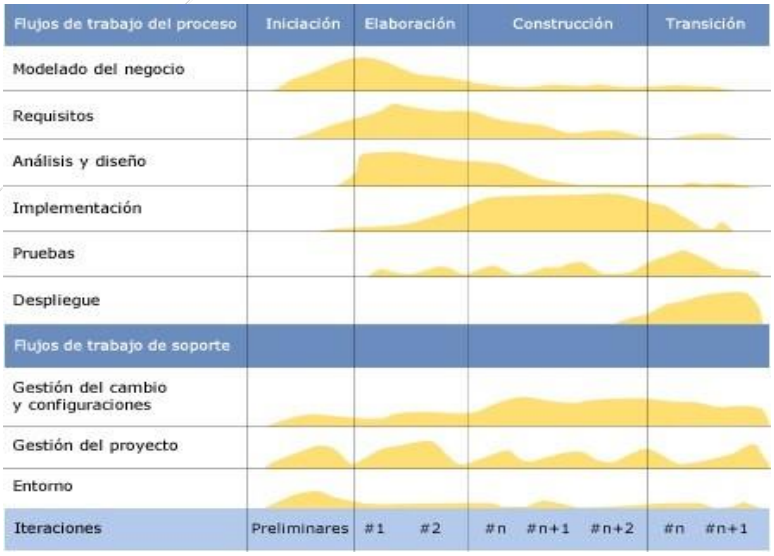



**Fig. Nº 1 Metodología de desarrollo RUP (19)**

El proceso de desarrollo de software además de su componente metodológico está sustentado en el uso de herramientas. A continuación se hace un análisis de las más convenientes y de las razones por las cuales son usadas en el propósito de esta investigación.

### 1.5.2. Selección del tipo de aplicación

**Aplicación Web**

Se decide realizar una aplicación web por su facilidad para actualizar y mantenerla sin distribuir e instalar software a varios usuarios potenciales. Además esta se puede ejecutar desde cualquier ordenador que tenga instalado un navegador, no es necesario que el ordenador sea de grandes prestaciones pues es una aplicación muy ligera y consume muy pocos recursos del equipo en el que es ejecutada. No hay problemas de incompatibilidad entre versiones porque todos los usuarios trabajan con la misma. Por parte de la seguridad puede ser tan alta como el equipo de desarrollo desee.

Para el desarrollo del sistema se hizo necesario utilizar determinadas herramientas y lenguajes para facilitar el trabajo y la modelación del mismo. A continuación se detallan las características significativas que definieron su uso.

### 1.5.3. Lenguaje y notación de modelado

**UML 2.0**

El Lenguaje de Modelado Unificado (UML, por sus siglas en inglés, Unified Modeling Language) es un lenguaje visual que permite especificar, construir y documentar los

artefactos del sistema de software respondiendo a un enfoque orientado a objetos.(21) UML 2.0 define varios tipos de diagramas para facilitar el entendimiento entre el cliente y los desarrolladores divididos en tres categorías: estructura de aplicación estática, en la que se encuentra el diagrama de clase, diagrama de componentes y diagrama de despliegue que son los usados en la aplicación; tipos generales de comportamiento, que permiten obtener el diagrama de casos de uso utilizado por la metodología seleccionada durante la recopilación de los requisitos; y aspectos de las interacciones incluyéndose en este el diagrama de secuencia.

**BPMN 2.0**

La Notación para el Modelado de Procesos de Negocio (BPMN, por sus siglas en inglés Business Process Modeling Notation) es una notación gráfica estandarizada basada en diagramas de flujo para definir procesos de negocio, ofreciendo una notación sencilla de fácil entendimiento por todos los involucrados en el modelado del negocio. Su selección está dada porque a pesar de estar emparentado con UML por el hecho de que ambos definen una notación gráfica para los procesos de negocio, usa un enfoque diferente para modelarlos, estar centrado en los procesos, mientras que UML en general ofrece un enfoque orientado a objetos para modelar aplicaciones. (22)

### 1.5.4. Herramientas utilizadas para el modelado

**Visual Paradigm for UML 8.0**

Visual Paradigm for UML es una herramienta CASE que soporta el modelado mediante UML. Facilita la reutilización, pues se dispone de una herramienta centralizada donde se encuentran los modelos utilizados en cada proyecto creado. (29).

Permite visualizar el flujo central detallado de cada proceso mediante diagramas, posibilitando la obtención de los mismos definidos por la metodología escogida, entre ellos el diagrama de clase, el modelo de datos, etc. Se selecciona además porque soporta el ciclo de vida del desarrollo de software, fundamentalmente el diseño orientado a objetos, construcción, pruebas y despliegue. Unido a las ventajas que brinda, en la UCI se cuenta con la licencia para su uso y está incluida como herramienta propuesta del Programa de mejoras. Permite visualizar el flujo central detallado de cada proceso mediante diagramas.

**AxurePro 5.5**

Axure permite generar prototipos interactivos o un diseño de interfaz de usuario sin necesidad de programar, así como exportar dichos prototipos en formato HTML. Permite

simular el comportamiento de las interfaces y cuenta con una amplia gama de componentes. Se especializa en las anotaciones permitiendo especificar el estado de cada elemento y el beneficio esperado, riesgo y la estabilidad. Proporciona además la navegación entre páginas y la generación de especificaciones en documentos. (31)

Esta herramienta es seleccionada por las facilidades que ofrece para trabajar con aplicaciones web, pues con ella se pueden generar prototipos en HTML, lo cual permite que el cliente pueda ver de manera anticipada cómo va a quedar el sistema, ya que esta herramienta simula acciones que el usuario realizará en el nuevo sistema familiarizándose así con el mismo.

### 1.5.5. Selección del entorno de desarrollo

**NetBeans 7.4**

Es un Entorno de Desarrollo Integrado (IDE) que permite desarrollar rápidamente aplicaciones web. Ofrece un excelente entorno para programar en PHP (acrónimo de Hypertext Preprocessor) y un amplio soporte para el marco de trabajo Symfony. Cuenta con las características de multiplataforma, integración con marcos de trabajo, depurar y ejecutar programas, importar y exportar proyectos. Es gratuito y de código abierto con una gran comunidad de usuarios y desarrolladores de todo el mundo. (27)

### 1.5.6. Marco de Trabajo

**Symfony 2.3.1**

Es un marco de trabajo de PHP basado en la arquitectura Modelo Vista Controlador (MVC, por sus siglas en inglés, Model View Controller). Es seleccionado teniendo en cuenta que fue diseñado para optimizar el desarrollo de aplicaciones web, proporcionando diferentes tecnologías para agilizar aplicaciones complejas y guiando al desarrollador a acostumbrarse al orden y buenas prácticas dentro del proyecto. Cuenta con un amplio soporte para la seguridad del sistema. (18)

Otra característica importante para su selección es la inclusión de Twig, un motor de plantillas que permite separar el código PHP del lenguaje de marcas de hipertexto (HTML, por sus siglas en inglés, Hypertext Markup Language); Symfony se distribuye bajo licencia Open Source MIT[1], que no impone restricciones y permite el desarrollo de

[1] Open Source MIT: Licencia de código abierto que otorga el Instituto Tecnológico de Massachusetts dentro de un proyecto que persigue una ubicación central para almacenar, mantener y dar seguimiento de software de código abierto que se desarrolla dentro de la comunidad del MIT.

aplicaciones de código abierto así como propietarias y además es el marco de trabajo definido en el Centro CEGEL para el desarrollo de aplicaciones web.

**ORM Doctrine 2.3**

Doctrine 2 es un mapeador de objeto relacional (ORM, por sus siglas en inglés, Object Relational Mapper) para PHP 5.3.0+ que proporciona persistencia transparente de objetos PHP. Se sitúa en la parte superior de una poderosa capa de abstracción de base de datos (DBAL, por sus siglas en inglés, Database Abstraction Layer). Una de sus principales características es que cuenta con la opción de escribir consultas a bases de datos en un lenguaje de consulta estructurado (SQL, por sus siglas en inglés Structured Query Language) orientado a objetos llamado lenguaje de consulta de Doctrine (DQL[2], por sus siglas en inglés, Doctrine Query Language). (20)

### 1.5.7. Lenguajes de programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. (23)

**Del lado del Servidor:**

**PHP5.3.8**

PHP (acrónimo de Hypertext Preprocessor), es un lenguaje de código abierto originalmente diseñado para el desarrollo web y puede ser incrustado en HTML. El código es ejecutado en el servidor, generando HTML y enviándolo al cliente. Es un lenguaje multiplataforma con capacidad de conexión con la mayoría de los manejadores de base de datos, entre ellos PostgreSQL, además de su soporte para extensiones de base de datos como la Capa de Abstracción de Base de Datos (DBA) y soporte para comunicarse con otros servicios usando protocolos tales como el Protocolo Ligero de Acceso a Directorios (LDAP, por sus siglas en inglés, Lightweight Directory Access Protocol), protocolo para la transferencia simple de correo electrónico (SMTP, por sus siglas en inglés, Simple Mail Transfer Protocol), entre otros. Permite la utilización de las técnicas de Programación Orientada a Objetos. (24)

[2] DQL: Es un lenguaje creado para ayudar al programador a extraer objetos de la base de datos. Es importante considerar su uso para mejorar el rendimiento.

**Motor de Plantillas Twig 2.1**

Twig es un motor de plantillas para PHP muy rápido y seguro ya que compila las plantillas en código PHP optimizado. Además es flexible debido a que el desarrollador puede definir sus propias etiquetas y filtros personalizados. (25) Implementa un novedoso mecanismo de herencia múltiple de plantillas y bloques de código, con esto se tienen todas las opciones posibles para crear plantillas; es fácil de aprender y con una amplia documentación.

**Del lado del Cliente:**

**JavaScript 1.2**

JavaScript es un lenguaje de programación interpretado multiplataforma orientado a objetos. No distingue entre tipos de objetos. La herencia se realiza a través del mecanismo de prototipado, los métodos y propiedades pueden ser añadidos a cualquier objeto dinámicamente. Los tipos de datos variables no son declarados (definición dinámica de tipos). Está diseñado para una fácil incrustación en otros productos y aplicaciones, tales como los navegadores Web. (26)

Este lenguaje se usa del lado del cliente, permite la realización de validaciones y el desarrollo de web dinámicas. Tiene la ventaja de ser incorporado en cualquier página web y de ser ejecutado sin la necesidad de instalar otros programas para ser visualizado.

**HTML 5.0**

Es un lenguaje para escritura de hipertexto, es decir, documentos de texto estructurados, incluye enlaces (links) que conducen a otros documentos o a otras fuentes de información y permite la inclusión de información por formularios, entre otros. HTML5 se caracteriza por ofrecer una mejor estructura eliminando el uso excesivo de las etiquetas <*div*>, esta se emplea para definir un bloque de contenido o sección de la página; con el objetivo de que la web sea más coherente y comprensible.

**CSS 3.0**

CSS (acrónimo de Cascading Style Sheets) es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML (del inglés eXtensible HyperText Markup Language). CSS es la mejor forma de separar los contenidos de su presentación y optimiza mucho el trabajo para la creación de páginas web complejas. Su uso mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad

de dispositivos diferentes. CSS 3 incorpora nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas.

### 1.5.8. Herramientas para base de datos

**PostgresSQL 9.1**

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[3] y con su código fuente disponible libremente. 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. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.(28)

**PgAdmin III 1.14**

PgAdmin es una herramienta rica en características para la administración de base de datos de código abierto, siendo la más avanzada en la plataforma de desarrollo PostgreSQL. La misma es designada para responder las necesidades de los clientes, desde escribir una simple consulta SQL hasta desarrollar una base de datos compleja, además puede ejecutarse en cualquier plataforma.

### 1.5.9. Servidor web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. Para la transmisión de todos estos datos generalmente se utiliza el protocolo HTTP[4] o el protocolo HTTPS[5] (la versión cifrada y autenticada), para estas comunicaciones pertenecientes a la capa de aplicación del modelo OSI[6].(35)

**Apache 2.0**

Es un servidor web de los más utilizados en el mundo por las ventajas que brinda, es de código abierto, multiplataforma y flexible debido a que es altamente configurable y de diseño modular. (36) Esta herramienta trabaja con varios lenguajes de script, entre ellos

[3] ***Berkeley Software Distribution*** o **BSD** (en español, «distribución de software Berkeley»), licencia de software libre que permite ver el código y modificarlo pero también permite cerrar el sistema o la aplicación.

[4] HyperText Transfer Protocol (en español protocolo de transferencia de hipertexto).

5 Secure HTTP.

6 Open System Interconnection (modelo de referencia de Interconexión de Sistemas Abiertos).

PHP, el cual ha sido seleccionado para la implementación. Presenta abundante documentación posibilitando así un mejor trabajo por parte de los miembros del equipo de desarrollo.

## 1.6. Patrones

**Patrón de arquitectura**

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades, relaciones y las formas en que colaboran. (37)

**Modelo Vista Controlador (MVC)**

El Modelo Vista Controlador es un patrón de arquitectura que divide una aplicación interactiva en tres componentes: 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.

El empleo de este patrón proporciona reutilización de los componentes, facilidad para la realización de pruebas unitarias a los mismos y simplicidad en el mantenimiento del sistema.

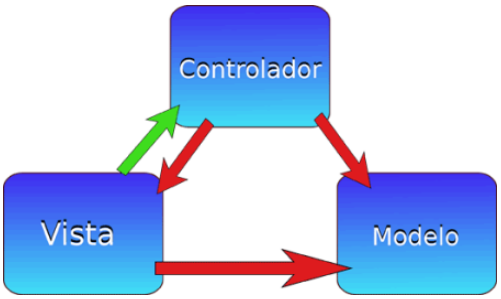



**Fig. Nº 2 Patrón Arquitectónico MVC**

## 1.7. Patrones de diseño

Los diseñadores expertos no resuelven los problemas desde el principio; reutilizan soluciones que han funcionado en el pasado. Se encuentran patrones de clases y objetos de comunicación recurrentes en muchos sistemas orientados a objetos. Estos patrones resuelven problemas de diseño específicos y hacen el diseño flexible y reusable. Un patrón de diseño es una descripción de clases y objetos comunicándose

entre sí, adaptada para resolver un problema de diseño general en un contexto particular. (40)

La utilización de patrones de diseño, permite ahorrar grandes cantidades de tiempo en la construcción de software. El producto obtenido es más fácil de comprender, mantener y extender. Existen versiones ya implementadas de estas funcionalidades comunes (marcos de trabajo) que permiten centrarse en desarrollar sólo la funcionalidad específica requerida por cada aplicación y que además, dan mejor imagen de profesionalidad y calidad. Los patrones de diseño se agrupan en dos grandes categorías: GRASP[7] y GOF[8]. Los primeros describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Dentro de este grupo de patrones se encuentran los siguientes: Experto, Creador, Bajo Acoplamiento, Controlador y Alta Cohesión. Los patrones de diseño GOF son 23 y se clasifican según su propósito en Creacionales, Estructurales, Comportamiento y según su ámbito en Objeto y de Clase.

## 1.8. Métricas

Las métricas de software tienen un papel decisivo en la obtención de un producto de alta calidad, porque determinan mediante estadísticas basadas en la experiencia, el avance del software y el cumplimiento de parámetros requeridos. (42)

### 1.8.1. Métricas para requisitos

**Estabilidad de los requisitos**

Es necesario lograr una estabilidad en los requisitos para el correcto funcionamiento de los demás flujos de trabajo. El objetivo de esta métrica es medir la estabilidad de los requisitos para asegurar su adecuación antes de pasar al próximo flujo de trabajo. Se considera que los requisitos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación. (43)

$$ETR = \left[\frac{RT - RM}{RT}\right] \times 100$$

Donde:

1. ETR: valor de la estabilidad de los requisitos.

[7] GRASP: Acrónimo de "General Responsibility Assignment Software Patterns" *en español Patrones generales de software para asignación de responsabilidades.*

[8] Gang of Four, en español Banda de los Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.

2. RT: total de requisitos definidos.
3. RM: número de requisitos modificados, que se obtienen como la sumatoria de los requisitos insertados, modificados y eliminados.

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

**Especificidad de los requisitos**

El objetivo de esta métrica es cuantificar la especificidad o falta de ambigüedad en la definición de los requisitos. Para calcular esta métrica deben contarse los requisitos que tuvieron igual interpretación por los revisores y compararlos con el total de requisitos definidos. El valor de esta métrica debe estar siempre entre 0 y 1. Mientras más cerca de 1 esté el valor de la especificidad de los requisitos mayor será la consistencia de la interpretación de los revisores para cada requisito y menor será la ambigüedad en la especificación de los requisitos. (44) La misma se calcula como:

$$ER = {}^{n_{ui}}/_{n_r}$$

Donde:

- ✓ ER: grado de especificidad de los requisitos.
- ✓ nui: número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.
- ✓ nr: cantidad de requisitos en una especificación y comprende la suma de los requisitos funcionales y no funcionales.

El valor de esta métrica debe estar siempre entre 0 y 1. Mientras más cerca de 1 esté el valor de ER mayor será la consistencia de la interpretación de los revisores para cada requisito y menor será la ambigüedad en la especificación de los requisitos.

**Grado de validación de los requisitos**

Los requisitos deben ser posibles de validar. La validación de los requisitos se realiza en consenso del equipo de desarrollo al contrastar lo que desea el cliente con la posibilidad real de implementarlo. El grado de validación de los requisitos mide la corrección en la definición de los requisitos. (42)

$$VR = {}^{n_c}/_{(n_c + n_{nv})}$$

Donde:

- ✓ VR: grado de validación de los requisitos.
- ✓ nc: número de requisitos que se han validado como correctos.
- ✓ nnv: número de requisitos no validados aún.

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de corrección en la definición de los requisitos.

### 1.8.2. Métricas para diseño

**Tamaño Operacional de Clase (TOC)**

Consiste en medir el tamaño general de una clase tomando el valor de la cantidad de operaciones que están encapsuladas dentro de dicha clase. (45)

Si el resultado obtenido indica valores grandes, significa que la clase posee un alto grado de responsabilidad, debido a esto se reducirá la reutilización, se hará mucho más difícil la implementación y la realización de pruebas de dicha clase. Por tanto mientras menor sea el valor para el TOC se hará mucho más fácil la reutilización de dicha clase dentro del sistema.

| Atributo | Categoría | Criterio |
|---|---|---|
| **Responsabilidad** | Baja | Cantidad de operaciones<=Promedio |
| | Media | Promedio<=Umbral<=2*Promedio |
| | Alta | Cantidad de operaciones >2*Promedio |
| **Complejidad de Implementación** | Baja | Cantidad de operaciones <= Promedio |
| | Media | Promedio<= Cantidad de operaciones <=2*Promedio |
| | Alta | Cantidad de operaciones >2*Promedio |
| **Reutilización** | Baja | Cantidad de operaciones >2*Promedio |
| | Media | Promedio<= Cantidad de operaciones <=2*Promedio |
| | Alta | Cantidad de operaciones <= Promedio |

**Tabla Nº 1 Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica**

**Relaciones entre Clases (RC)**

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:(46)

**Acoplamiento**: un aumento del RC implica un aumento del Acoplamiento de la clase.

**Complejidad de mantenimiento**: un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.

**Cantidad de pruebas**: un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

| Atributo | Categoría | Criterio |
|---|---|---|
| **Acoplamiento** | Ninguno | 0 |
| | Bajo | 1 |
| | Medio | 2 |
| | Alto | Cantidad de relaciones de uso > 2 |
| **Complejidad del Mantenimiento** | Baja | Cantidad de relaciones de uso <= Promedio |
| | Media | Promedio <= Cantidad de relaciones de uso <= 2* Promedio |
| | Alta | Cantidad de relaciones de uso > 2* Promedio |
| **Cantidad de Pruebas** | Baja | Cantidad de relaciones de uso <= Promedio |
| | Media | Promedio <= Cantidad de relaciones de uso <= 2* Promedio |
| | Alta | Cantidad de relaciones de uso > 2* Promedio |

**Tabla Nº 2 Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica**

## 1.9. Pruebas

Las pruebas son técnicas de comprobación dinámica, estas siempre implican la ejecución del programa permitiendo: (47)

- ✓ Evaluar la calidad de un producto.
- ✓ Mejorarlo identificando defectos y problemas.

### 1.9.1. Pruebas de verificación

Conjunto de actividades que aseguran que el software implemente correctamente una función específica. (46)

**Pruebas a nivel de unidad**

Se realizan pruebas funcionales durante la fase de construcción, específicamente en el flujo de trabajo de implementación; las cuales se basan en probar los componentes implementados como unidades individuales. Estas se realizan a través de dos métodos: pruebas de caja blanca y pruebas de caja negra.

**Método de prueba de Caja Blanca**

El método de prueba de caja blanca, denominado a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero de software puede obtener casos de prueba que: (1) garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; (2) ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; (3) ejecuten todos los bucles en sus límites y con sus límites operacionales; y (4) ejerciten las estructuras internas de datos para asegurar su validez. (44)

**Método de prueba de Caja Negra**

Las pruebas de caja negra son aquellas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del funcionamiento del sistema sin tener mucho en cuenta la estructura interna del software. (50)

Estas pruebas permiten encontrar:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a las Bases de Datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

Según Pressman para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

1. **Técnica de la Partición de Equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
2. **Técnica del Análisis de Valores Límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

3. **Técnica de Grafos de Causa-Efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

En este caso se escoge dentro del método de Caja Negra la técnica de la Partición de Equivalencia que es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

### 1.9.2. Pruebas de validación

Es el proceso de evaluación de un sistema o de uno de sus componentes durante o al final del proceso de desarrollo para determinar si satisface los requisitos marcados por el cliente. (46)

**Pruebas de Aceptación**

La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o un especialista de la aplicación y es conducida a determinar cómo el sistema satisface sus criterios de aceptación validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio. Está considerada como la fase final del proceso para crear una confianza en que el producto es el apropiado para su uso en explotación.

## 1.10. Conclusiones Parciales

El estudio de las diferentes soluciones informáticas para el proceso de revisiones permitió demostrar que las soluciones analizadas no se ajustan a las necesidades del Grupo de Calidad del Centro CEGEL. La metodología, las herramientas y lenguajes de programación que serán utilizados son los adecuados para el desarrollo del sistema y se resume en el uso de tecnologías de código abierto.

# Capítulo II Propuesta de Solución

## 2.1. Introducción

En el presente capítulo se expone el negocio, requisitos, análisis y diseño de la herramienta a desarrollar, describiéndose todos los artefactos que serán generados, además de las validaciones correspondientes, la descripción de la arquitectura del sistema y los patrones a utilizar.

## 2.2. Modelado del Negocio

### 2.2.1. Descripción del Negocio

En el proceso de revisiones del Grupo de Calidad del Centro CEGEL, inicialmente el Asesor de la Calidad realiza la planificación anual de revisiones de adherencia a procesos y productos de cada proyecto quedando estructurado el Plan de revisiones del Centro.

El proceso de revisiones comienza cuando el Asesor de la Calidad asigna la tarea de evaluación al Revisor Líder quien tiene la función de garantizar la realización de la revisión, el cual procede a organizar la misma determinando los recursos humanos necesarios para ello, y así distribuir las tareas a los revisores siendo ellos los encargados de llevar a cabo dicho proceso. Luego el Revisor Líder le envía una notificación de revisión al proyecto involucrado, definiendo en ella el alcance y la fecha planificada en la cual se realizará la evaluación.

Al comenzar la evaluación el Revisor Líder realiza una reunión de inicio con los implicados para dar a conocer el objetivo, alcance, duración de la evaluación a efectuar y conciliar las actividades a realizar, así como el equipo de revisores. Se designan las responsabilidades para evaluar cada área de proceso y elabora una minuta de reunión. Luego el equipo de revisores realiza entrevistas a los distintos miembros del proyecto acorde al rol que desempeñan y aplica una lista de chequeo a los procesos y productos de trabajo. Las no conformidades identificadas se recogen en el Registro de las mismas.

Partiendo del artefacto en cuestión el Revisor Líder registra la evaluación realizada en el Registro de Evaluaciones del Proyecto y redacta el Informe de Evaluación. Posteriormente comunica los resultados en la reunión de cierre con la participación de todos los involucrados a través del Informe de Evaluación y se elabora una minuta de esta actividad. Después crea el Expediente de la evaluación para el cual tiene como máximo tres días para enviarlo al equipo de desarrollo y evalúa el servicio realizado.

El Administrador de la Calidad junto al Jefe de Proyecto analiza los resultados y define las acciones correctivas requeridas, definiendo responsables y fechas de cierre para las mismas, seguido; se actualiza el Registro de NC. Ambos se encargan además de controlar sistemáticamente la ejecución de las acciones correctivas según lo previsto y la solución de las no conformidades.

Las no conformidades que no pueden ser resueltas en cada instancia el Administrador de Calidad las escala a instancias superiores para su solución, dándole seguimiento hasta que todas se encuentren cerradas.

El Asesor de Calidad a partir del análisis de la información obtenida del Registro de No Conformidades, Plan de Revisiones y Registro de Evaluaciones elabora los reportes de tendencias y los analiza para tomar decisiones al respecto con el objetivo de mejorar la calidad de los productos y servicios entregados. Así concluye el proceso de revisiones del Centro CEGEL.

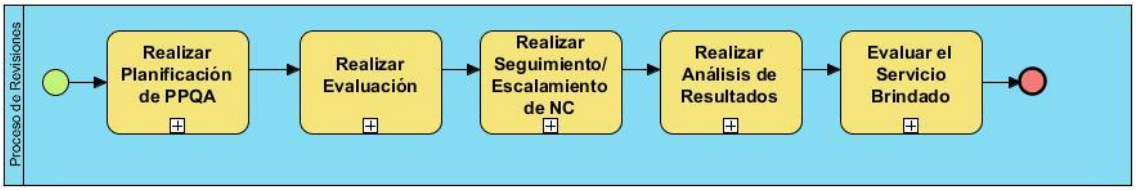



**Fig. Nº 3 Subprocesos del Modelo del Negocio**

### 2.2.2. Objeto de Automatización

Teniendo en cuenta el negocio descrito anteriormente se puede decir que se desean automatizar varios servicios que ayudarán a mejorar las dificultades planteadas. Por tanto lo primero a automatizar es la planificación anual de revisiones puesto que es importante tener registrado todas las revisiones que se realizarán en el año. Seguidamente, la gestión de las tareas donde el usuario puede visualizar las que le fueron asignadas. Además, se informatizará el servicio de envío de Notificación de revisión para que no se realice por medio del correo electrónico como se realiza actualmente. El sistema contará además con la Gestión de los elementos de la lista de chequeo y la Evaluación del servicio realizado. También se podrá generar desde el sistema los diferentes documentos obtenidos en el proceso de revisión como lo son el Registro de Evaluación del Proyecto, Plan de Anual de Revisiones, Registro de No Conformidades, Informe de Evaluación, Minuta de Reunión de Apertura, Minuta de Reunión de Cierre y los reportes de tendencias para conformar el Expediente de Evaluación.

### 2.2.3. Descripción de los actores del negocio

Actor no es más que el rol que alguien o algo toma cuando interactúa con el negocio. Los actores representan un rol y no una persona específica.

**Miembro de Equipo de Desarrollo:** responde las entrevistas realizadas por los revisores y resuelve las no conformidades detectadas.

**Proyecto:** entidad a la que se le realizará la revisión.

### 2.2.4. Descripción de los trabajadores del negocio

Los trabajadores del negocio son aquellos que trabajan realizan las actividades que se llevan a cabo en el proceso. Muestran las responsabilidades que puede tener una persona, por lo que representa una abstracción de una persona que actúa dentro del negocio. (51)

**Asesor de la Calidad:** asigna tareas a los revisores líderes, se encarga del análisis de tendencias. Realiza la planificación anual de revisiones. Puede realizar todas las actividades del Revisor líder y del Revisor.

**Revisor líder:** revisor que está al frente de un grupo de revisores al cual organiza y distribuye tareas. Es responsable de la aplicación del proceso total de revisión de calidad. Efectúa y dirige la reunión de inicio y cierre de la evaluación. Supervisa, planifica, controla y cierra las actividades propias de la revisión. Controla además, la generación de los artefactos del proceso de revisión y terminado el proceso redacta el informe final. Puede realizar todas las actividades del Revisor.

**Revisor:** es el responsable de trabajar con la lista de chequeo y realizar las entrevistas a los miembros del equipo de desarrollo para generar las no conformidades en caso que existan. Participa en la reunión de inicio y cierre de la evaluación.

**Jefe de Proyecto:** participa en la reunión de inicio y cierre de las evaluaciones y analiza los resultados. Elabora las acciones correctivas para resolver las no conformidades detectadas durante las evaluaciones. Evalúa el servicio prestado por el Grupo de Calidad. Da solución a las no conformidades encontradas que corresponden a sus respectivas áreas de trabajo dentro del proyecto. Además es entrevistado por los revisores.

**Administrador de la Calidad:** mantiene actualizado el Registro de evaluaciones del proyecto. Monitorea el cumplimiento de las acciones correctivas, el estado de las no conformidades y el escalamiento de las mismas en caso que se necesite.

### 2.2.5. Patrones para modelación del negocio

Para modelar el negocio utilizando BPMN se hizo uso de determinados patrones, que se encuentran agrupados de la siguiente manera:

**Patrones básicos de control de flujo**

- ✓ **Secuencia**

El patrón secuencia expone que en un mismo proceso para que una actividad pueda iniciarse tiene que haberse ejecutado la anterior. En el caso siguiente para determinar los recursos humanos necesarios para la revisión tiene que haberse aceptado la tarea de revisión.


**Fig. Nº 4 Patrón Secuencia**

- ✓ **Selección exclusiva**

Este patrón permite realizar decisiones a la hora de ejecutar actividades decidiendo en este caso qué hacer luego de aplicar la lista de chequeo.

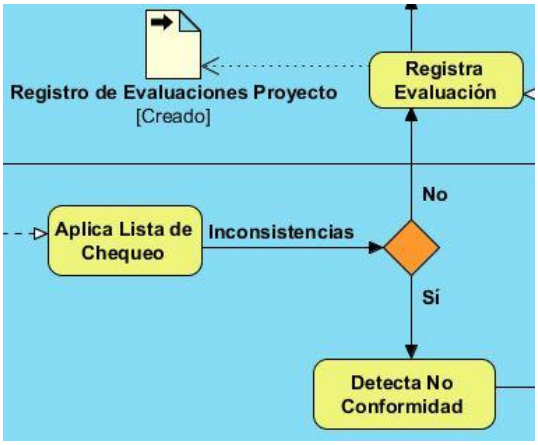



**Fig. Nº 5 Patrón Selección Exclusiva**

Los diagramas correspondientes al modelo de negocio se pueden visualizar en los anexos del 1 al 5.

Luego de haber modelado el negocio para el proceso de revisiones del Grupo de Calidad del Centro CEGEL se realiza la descripción de los actores del sistema, donde se asocia cada actor con sus respectivas actividades en el sistema.

## 2.3. Descripción de los actores del sistema

**Asesor de la Calidad:** realiza la planificación anual de las revisiones, asigna tareas a los revisores líderes y se encarga del análisis de tendencias. Puede realizar todas las actividades del Revisor líder y del Revisor. Además consulta los resultados de la evaluación del servicio brindado, así como gestionar las preguntas para e mismo. Gestionar proyectos, módulos, usuarios, elementos de la lista de chequeo y los datos de nomenclador.

**Revisor Líder:** Responsable de la aplicación del proceso total de revisión de calidad. Controla además, la generación de los artefactos del proceso de revisión y terminado el proceso redacta el informe final. Le da seguimiento a la revisión realizada. Puede realizar todas las actividades del Revisor. Además envía notificación de revisión y gestiona la misma.

**Revisor:** responde la lista de chequeo para generar las no conformidades en caso que existan. Participa en la reunión de inicio y cierre de la evaluación.

**Jefe de Proyecto:** actualiza el registro de evaluaciones del proyecto con el monitoreo del estado de las no conformidades y sus acciones correctivas. Evalúa el servicio entregado por el Grupo de Calidad.

**Administrador de la Calidad:** actualiza el registro de evaluaciones del proyecto con el monitoreo del estado de las no conformidades, sus acciones correctivas y el escalamiento.

## 2.4. Requisitos

### 2.4.1. Técnicas para la captura de requisitos.

El levantamiento de requisitos es una etapa esencial en el arranque de todo proyecto de desarrollo de software y debe de realizarse efectivamente para elevar las probabilidades de éxito de los proyectos. Soluciona las posibles disparidades entre las personas involucradas en el mismo, con el propósito de definir y refinar los requisitos para cumplir las restricciones acordadas por ambas partes.

El proceso de levantamiento de requisitos soporta el desarrollo de la especificación de los requisitos, de tal forma que tengan los siguientes atributos: ser completos, consistentes y han de estar dentro del alcance del proyecto, tener un único identificador, cumplir con los objetivos de los clientes, ser viables y apropiados para el desarrollo, además de tener capacidad de prueba. (52) Para ello fue utilizada la siguiente técnica:

**Entrevista:** se emplea para la obtención de requisitos por ser la mejor fuente de información cualitativa. Tiene como ventaja ser una oportunidad para que el analista conozca el grado de aceptación o resistencia que existe entre los usuarios hacia el sistema que se desea diseñar.

A partir del modelado del negocio, la descripción de los actores del sistema y el resultado de aplicar las técnicas de obtención de requisitos se obtuvieron los siguientes requisitos funcionales:

RF_1 Adicionar planificación anual de revisiones.
RF_2 Editar planificación anual de revisiones.
RF_3 Listar planificación anual de revisiones.
RF_4 Mostrar planificación anual de revisiones.
RF_5 Eliminar planificación anual de revisiones.
RF_6 Adicionar tarea.
RF_7 Editar tarea.
RF_8 Listar tarea.
RF_9 Mostrar tarea.
RF_10 Eliminar tarea.
RF_11 Adicionar notificación de revisión.
RF_12 Editar notificación de revisión.
RF_13 Listar notificación de revisión
RF_14 Mostrar notificación de revisión
RF_15 Eliminar notificación de revisión
RF_16 Enviar notificación de revisión
RF_17 Adicionar elementos de la lista de chequeo.
RF_18 Editar elementos de la lista de chequeo.
RF_19 Listar elementos de la lista de chequeo.
RF_20 Mostrar elementos de la lista de chequeo.
RF_21 Eliminar elementos de la lista de chequeo.
RF_22 Adicionar no conformidad.
RF_23 Editar no conformidad.

RF_24 Listar no conformidad.

RF_25 Mostrar no conformidad.

RF_26 Eliminar no conformidad.

RF_27 Adicionar revisión de proyecto.

RF_28 Editar revisión de proyecto.

RF_29 Listar revisión de proyecto.

RF_30 Mostrar revisión de proyecto.

RF_31 Eliminar revisión de proyecto.

RF_32 Adicionar evaluación de proyecto.

RF_33 Editar evaluación de proyecto.

RF_34 Listar evaluación de proyecto.

RF_35 Mostrar evaluación de proyecto.

RF_36 Eliminar evaluación de proyecto.

RF_37 Adicionar usuario.

RF_38 Editar usuario.

RF_39 Listar usuario.

RF_40 Mostrar usuario.

RF_41 Eliminar usuario.

RF_42 Adicionar datos de nomenclador.

RF_43 Editar datos de nomenclador.

RF_44 Listar datos de nomenclador.

RF_45 Mostrar datos de nomenclador.

RF_46 Eliminar datos de nomenclador.

RF_47 Adicionar proyecto.

RF_48 Editar proyecto.

RF_49 Listar proyecto.

RF_50 Mostrar proyecto.

RF_51 Eliminar proyecto.

RF_52 Adicionar módulo.

RF_53 Editar módulo.

RF_54 Listar módulo.

RF_55 Mostrar módulo.

RF_56 Eliminar módulo.

RF_57 Adicionar preguntas de evaluación del servicio brindado.

RF_58 Editar preguntas de evaluación del servicio brindado.

RF_59 Listar preguntas de evaluación del servicio brindado.

RF_60 Mostrar preguntas de evaluación del servicio brindado.
RF_61 Eliminar preguntas de evaluación del servicio brindado.
RF_62 Evaluar el servicio brindado
RF_63 Visualizar resultado de evaluación del servicio brindado
RF_64 Evaluar el grado de implementación de los procesos
RF_65 Visualizar reporte de indicador Adherencia a criterios de evaluación.
RF_66 Visualizar reporte de indicador Distribución de impacto y tipo de no conformidades.
RF_67 Guardar reporte de indicador Adherencia a criterios de evaluación.
RF_68 Guardar reporte de indicador Distribución de impacto y tipo de no conformidades.
RF_69 Autenticar usuarios.

### 2.4.2. Requisitos No Funcionales

Las técnicas de obtención de requisitos también ayudaron a determinar los requisitos no funcionales donde se especifican las propiedades del sistema que tienen que ver con las características no funcionales.(54)

**Disponibilidad**

RNF_1 El sistema debe estar disponible todo el tiempo para sus usuarios, descontando el tiempo en que se encuentre en mantenimiento.

RNF_2 El período entre fallos recuperables, como por ejemplo fallos en el servidor principal no debe exceder las 24 horas.

**Usabilidad**

RNF_3 La aplicación informática a desarrollar será una aplicación web.

RNF_4 El sistema debe ser intuitivo y tener un alto nivel de usabilidad permitiendo que usuarios sin mucho conocimiento informático pueda utilizarlo sin problemas. Además el sistema debe utilizar en el diseño del mismo un lenguaje que resulte familiar y asequible al usuario que interactuará con él.

**Eficiencia**

RNF_5 Tiempo de respuesta: la mayoría de los procesos que se implementan con transacciones donde se modifica la base de datos deben tener tiempos de respuesta no mayores a los 3 segundos. En el caso de información que involucre consultas a las base de datos los tiempos de respuestas no deben exceder los 10 segundos.

**Soporte**

RNF_6 La aplicación debe estar documentada y proveer el código fuente previendo futuras modificaciones en el mismo para potenciar su alcance o eficiencia.

RNF_7 Consta con la documentación necesaria para el aprendizaje sobre el uso de la herramienta informática.

**Interfaz**

RNF_8 Fácil de usar para el usuario.

RNF_9 Se usarán colores que den una combinación agradable a la aplicación.

**Seguridad**

RNF_10 La seguridad está a nivel de gestión de roles con el fin de mantener la integridad de los datos, por el cual el acceso a la herramienta será a través de estos roles, trayendo consigo además la protección de la información.

**Servidor para instalar el Gestor de Base de Datos**

RNF_11 PostgreSQL v9.1.

RNF_12 Administrador de PostgreSQL: PgAdmin III.

**Requisitos de hardware requerido para desplegar y utilizar la aplicación web**

RNF_13 La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo TCP/IP.

RNF_14 La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo HTTP.

- ✓ **PC Cliente**

Las PC clientes deben cumplir con los siguientes requisitos de hardware:

RNF_15 Ordenador Pentium IV con 1.7 GHz de velocidad de microprocesador o superior.

RNF_16 Memoria RAM mínimo 256MB.

- ✓ **PC Servidor**

La PC servidor debe cumplir con los siguientes requisitos de hardware:

RNF_17 Ordenador Pentium IV con 1.7 GHz de velocidad de microprocesador o superior.

RNF_18 Memoria RAM mínimo 512MB.

**Requisitos de software**

RNF_19 Servidor Web Apache 2.2 o superior, con módulo PHP 5.3 o superior.

RNF_20 Navegador web Mozilla Firefox 13.0 o superior (para la pc cliente).

RNF_21 El sistema debe ser multiplataforma con enfoque en tecnologías basadas en software no propietario.

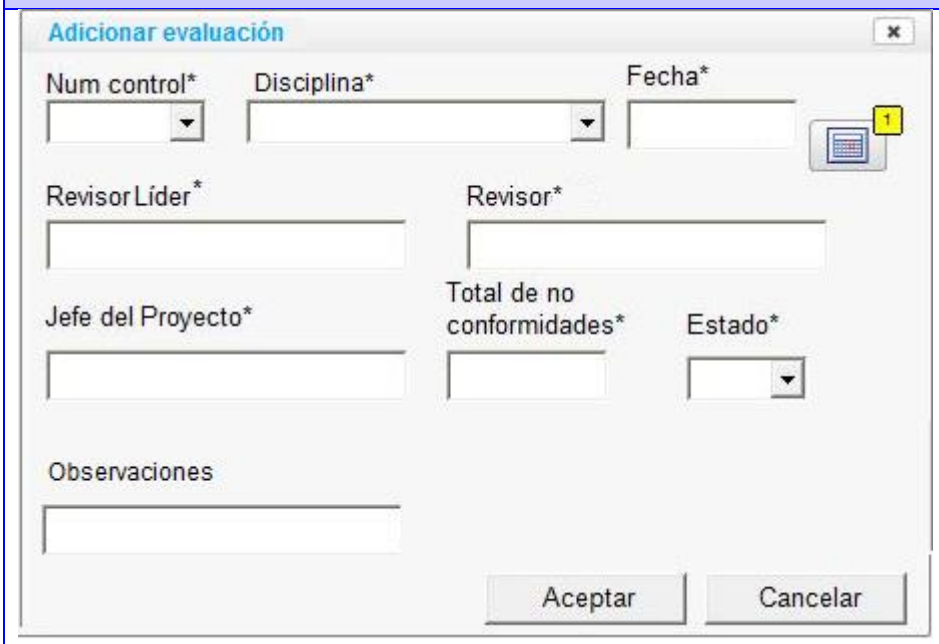
### 2.4.3. Especificación de requisitos

A continuación se muestra la especificación del requisito Adicionar evaluación de proyecto, donde se muestra su prototipo así como la descripción de sus campos y las restricciones de los mismos. El documento de especificación de requisitos puede consultarse en el expediente de proyecto.

Especificación del requisito Adicionar evaluación de proyecto.

| | | | |
|---|---|---|---|
| *[RF_32]* | Adicionar evaluación de proyecto. | El sistema permitirá adicionar una evaluación a partir de los criterios introducidos. | Media |

***Prototipo***




| ***Campos*** | ***Tipos de Datos*** | ***Reglas o Restricciones*** |
|---|---|---|
| Número de control | Cadena | Obligatorio<br>Formato: letras, números y carácter especial “-”<br>Longitud: máximo 30 caracteres |
| Disciplina | Cadena | Obligatorio<br>Formato: solo letras |

| | | |
|---|---|---|
| | | Longitud: máximo 50 caracteres |
| Fecha | Fecha | Obligatorio<br>Longitud: máximo 10 caracteres.<br>Formato: DD/MM/AAAA |
| Revisor líder | Cadena | Obligatorio<br>Formato: letras y espacio<br>Longitud: máximo 50 caracteres<br>Formato: empieza con mayúscula |
| Revisor | Cadena | Obligatorio<br>Longitud: máximo 500 caracteres<br>Formato: letras, espacio y carácter ";" |
| Jefe del proyecto | Cadena | Obligatorio<br>Longitud: máximo 50 caracteres<br>Formato: letras y espacio |
| Total de no conformidades | Entero | Obligatorio<br>Longitud: máximo 4 caracteres<br>Formato: números |
| Estado | Cadena | Obligatorio<br>Longitud: máximo 20 caracteres<br>Formato: solo letras |
| **Observaciones** | [opcional] | |

**Tabla Nº 3 Especificación del requisito Adicionar Evaluación**

### 2.4.4. Métricas para requisitos

Los requisitos del software son la base de las medidas de la calidad. En la disciplina Requisitos se tuvo en cuenta la métrica para medir su estabilidad, especificidad y grado de validación.

**Aplicación de la métrica Estabilidad de requisitos:**

Teniendo en cuenta que se identificaron un total de 69 requisitos funcionales, de los cuales 7 resultaron modificados (2 por inserción, 3 por actualización y 2 por eliminación), se calcula:

ETR = [(69– 7) / 69 * 100]= 89,85

Como resultado se obtuvo un valor de 89.85. Dicha cifra demuestra que no se han realizado cambios significativos sobre los requisitos, son estables y, por tanto, es confiable el análisis y diseño sobre ellos.

**Aplicación de la métrica Especificidad de los requisitos:**

La especificidad de los requisitos se calcula como:

Q**1** = n**ui** /n**r** = 69 /69 = 1

Como resultado se obtuvo que la especificación de los requisitos no presenta ambigüedad, asegurando un alto nivel de calidad en el proceso de especificación.

**Aplicación de la métrica Grado de validación:**

El grado de validación de los requisitos se calcula:

Q**3** = n**c** / (n**c** + n**nv**) = 69 /(69+ 0) = 1

La aplicación de esta métrica dio como resultado 1, por lo tanto se concluye que la definición de los requisitos es correcta.

### 2.4.5. Validación de los requisitos

El proceso tiene por finalidad comprobar que los requisitos del software poseen todos los atributos de calidad: consistentes, completos, precisos, realistas, verificables y que definan lo que el usuario desea del producto final con el objetivo de evitar cambios en los mismos una vez que se haya avanzado en el proceso de desarrollo. Se aplican dos técnicas para la validación de los requisitos las cuales son:

- ✓ *Revisión de requisitos.* Se realizaron reuniones para localizar errores en el documento. Donde se agregaron requisitos y se modificaron otros.
- ✓ *Prototipado.* Se construyó una maqueta del sistema a desarrollar a partir de los requisitos recogidos en la especificación. La misma se evaluó por el cliente para comprobar su corrección y completitud realizándose los cambios necesarios para cumplir con las expectativas del cliente.

## 2.5. Modelo de casos de uso del sistema

El modelo de casos de uso describe las funcionalidades de la propuesta de solución. Este se utiliza para la comunicación clara y el levantamiento de requisitos, representándolos mediante diagramas de casos de uso. (47)

### 2.5.1. Patrones de casos de uso

**CRUD Completo**

**Descripción:** El patrón CRUD Completo consiste en un caso de uso para administrar la información (CRUD Información), permite modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar.

**Aplicabilidad:** Este patrón deberá ser usado cuando todas las operaciones contribuyen al mismo valor de negocio y todas son cortas y simples. (41)

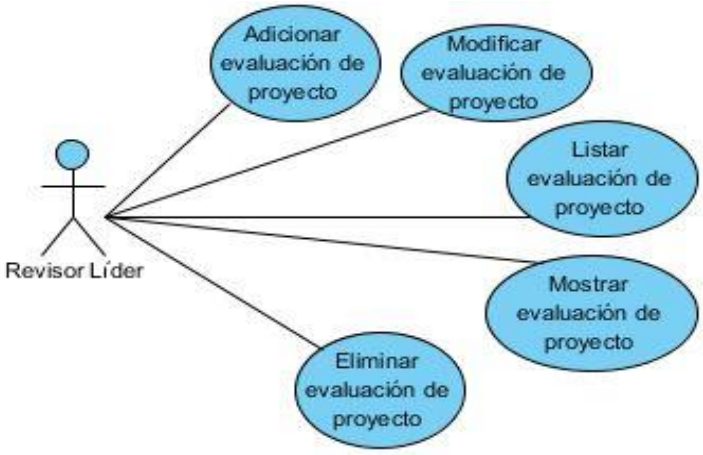



**Fig. Nº 6 Patrón CRUD Completo**

**Múltiples Actores**

A un Caso de Uso ingresan más de dos actores y estos tienen un rol común. Ejemplo: Se tiene el siguiente caso donde en un sistema web dos actores acceden a un mismo caso de uso.

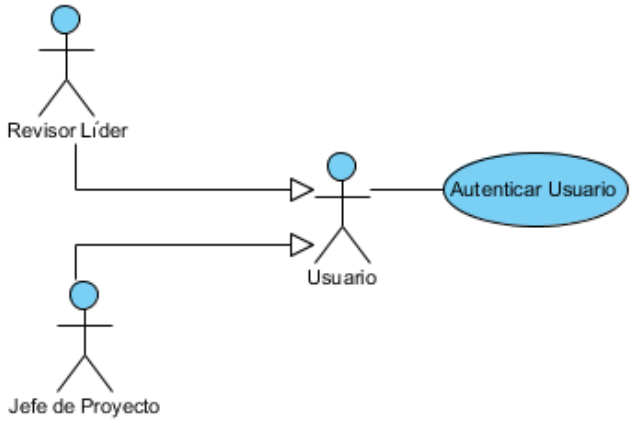



**Fig. Nº 7 Patrón Múltiple Actores**

**Inclusión**

Relación de dependencia entre dos casos de uso que denota la inclusión del comportamiento de un escenario en otro. Como ejemplo de su aplicación se puede ver la figura que a continuación se muestra:

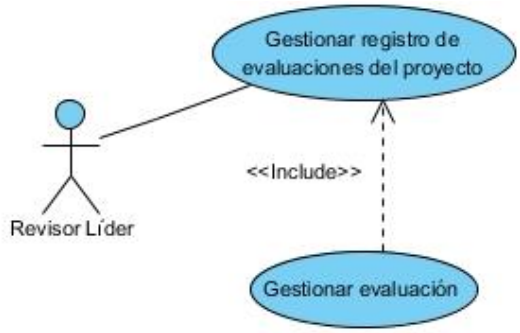



**Fig. Nº 8 Patrón inclusión**

### 2.5.2. Diagrama de casos de uso del sistema

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de casos de uso y los actores participantes en el proceso. Los mismos sirven para facilitar la comunicación con los futuros usuarios del sistema, resultando especialmente útiles para determinar las características necesarias que tendrá el sistema. En el anexo 7 se puede visualizar el diagrama de casos de uso del sistema.

### 2.5.3. Descripción de casos de usos del sistema

Se realiza con el objetivo de describir cada una de las acciones que realiza el sistema tras las opciones que selecciona el usuario. A continuación se muestra la especificación del CU Gestionar evaluación. Las demás especificaciones se encuentran en el documento de especificación de casos de uso perteneciente al expediente del proyecto.

| | |
|---|---|
| **Objetivo** | Gestionar evaluación |
| **Actores** | Revisor líder |
| **Resumen** | El caso de uso se inicia cuando el Revisor Líder necesita gestionar una evaluación. Consiste en que el Revisor Líder selecciona la opción de adicionar una nueva evaluación y llena los campos requeridos para su creación. También tiene la posibilidad de seleccionar una evaluación ya sea para modificar sus parámetros o eliminarlo. El caso de uso termina con la creación, modificación o eliminación de una evaluación. |
| **Complejidad** | Media |

| | | |
|---|---|---|
| **Prioridad** | Media | |
| **Precondiciones** | ✓ Debe ser inicializado por el Revisor Líder<br>✓ El usuario y la contraseña deben ser correctos. | |
| **Postcondiciones** | ✓ El sistema queda con una nueva evaluación creada.<br>✓ El sistema queda con una evaluación actualizada.<br>✓ El sistema queda con una evaluación eliminada. | |
| ***Flujo de eventos*** | | |
| ***Flujo básico Gestionar Evaluación*** | | |
| | ***Actor*** | ***Sistema*** |
| 1. | Accede a la opción Registrar evaluación | |
| 2. | | Muestra la interfaz Adicionar Evaluación con los siguientes campos:<br>Número de Control, Disciplina, Fecha de evaluación, Revisor, Jefe de proyecto, Estado de la evaluación, Total de no conformidades y Observaciones. Además se visualizan las opciones:<br>✓ Adicionar<br>✓ Listar |
| 3. | Selecciona la opción Adicionar evaluación | Muestra la interfaz Mostrar Evaluación con los siguientes campos:<br>Número de Control, Disciplina, Fecha de evaluación, Revisor, Jefe de proyecto, Estado de la evaluación, Total de no conformidades y Observaciones. Además se visualizan las opciones:<br>✓ Editar<br>✓ Listar<br>✓ Eliminar |
| 4. | Selecciona la opción Listar. | |
| 5. | | Muestra la interfaz Listar Evaluación de proyectos con los siguientes campos:<br>Número de Control, Disciplina, Fecha de evaluación, Revisor, Jefe de proyecto, Estado de la evaluación, Total de no conformidades y Observaciones. Además se visualizan las opciones:<br>✓ Adicionar<br>✓ Eliminar<br>✓ Mostrar<br>✓ Editar |

| | | |
|---|---|---|
| 6. | Selecciona la opción eliminar<br><br>Mostrar (sección 5a)<br><br>Editar (sección 3a) | |
| 7. | | Muestra un mensaje preguntando si está seguro que desea eliminar de forma permanente, con las opciones:<br>✓ Aceptar<br>✓ Cancelar |
| 8. | Selecciona la opción aceptar.<br><br>Cancelar (flujo alterno 7b) | |
| 9. | | Elimina la evaluación y muestra la interfaz Lista de evaluación de proyectos con los siguientes campos:<br><br>Número de Control, Disciplina, Fecha de evaluación, Revisor, Jefe de proyecto, Estado de la evaluación, Total de no conformidades y Observaciones. Además se visualizan las opciones:<br>✓ Adicionar<br>✓ Eliminar<br>✓ Mostrar<br>✓ Editar |
| **Secciones** | | |
| **5a. Mostrar** | | |
| | **Actor** | **Sistema** |
| 1. | Selecciona la evaluación deseada, marcando en el botón correspondiente a la misma. | |
| 2. | | Muestra la evaluación seleccionada con los siguientes campos:<br><br>Número de Control, Disciplina, Fecha de evaluación, Revisor, Jefe de proyecto, Estado de la evaluación, Total de no conformidades y Observaciones. Además se visualizan las opciones:<br>✓ Aceptar<br>✓ Actualizar<br>✓ Eliminar |
| **3a. Editar** | | |

| | **Actor** | **Sistema** |
|---|---|---|
| 1. | Selecciona la opción Editar. | |
| 2. | | Muestra la interfaz de Editar evaluación con los siguientes campos:<br>Número de Control, Disciplina, Fecha de evaluación, Revisor, Jefe de proyecto, Estado de la evaluación, Total de no conformidades y Observaciones. Además se visualizan las opciones:<br>✓ Actualizar<br>✓ Listar<br>✓ Eliminar |
| 3. | Selecciona la opción actualizar. Flujo alterno de campos incorrectos e incompletos (flujo alterno 5b). | |
| 4. | | Actualiza la evaluación y se mantiene en la página actual.<br>Además se visualizan las opciones:<br>✓ Aceptar<br>✓ Actualizar<br>✓ Eliminar |
| **Flujos Alternos** | | |
| **5b. Flujo alterno de campos incorrectos e incompletos** | | |
| | **Actor** | **Sistema** |
| 1. | Introduce datos incorrectos o inconclusos. | |
| 2. | | Muestra un mensaje de error y señala los campos que contienen datos incorrectos o incompletos. Se mantiene visualizada la interfaz actual. |
| **7b. Flujo alterno Cancelar** | | |
| | **Actor** | **Sistema** |
| 1. | Selecciona la opción cancelar. | |
| 2. | | Cancela la acción a realizar y muestra la interfaz anterior. |

**Tabla Nº 4 Especificación de casos de uso Gestionar Evaluación**

Una vez concluido el proceso de análisis se realiza la definición de la arquitectura, la cual puede considerarse como el puente entre los requisitos del sistema y la implementación. Independientemente de la metodología seleccionada, es importante obtener una arquitectura con la documentación necesaria, y asegurar que el sistema cumple con los servicios y la funcionalidad que espera el usuario.

La arquitectura de software hace alusión a la especificación de la estructura del sistema, entendida como la organización de componentes y relaciones entre ellos, además de ser la base del diseño del sistema a desarrollar.

## 2.6. Arquitectura del sistema

El desarrollo de la presente aplicación responde a una arquitectura basada en capas compuesta por: la capa de presentación, la capa de acceso a datos y la capa de datos. Es importante destacar el uso del patrón Modelo Vista Controlador (MVC) en las capas de presentación y acceso a datos. En la figura se puede apreciar la estructura arquitectónica en capas y la ubicación de los componentes en cada una de ellas, destacando los elementos del patrón utilizado. La arquitectura propuesta posee la característica de tener varios complementos transversales a las capas para garantizar seguridad, tratamiento de excepciones, entre otros aspectos. Cada uno de los módulos o bundles[9] se estructuran arquitectónicamente igual. Las entidades del dominio son accesibles en la sub-capa servidor de la capa de presentación y la capa de acceso a datos. (55)

[9] Un bundle es un conjunto estructurado de archivos que implementan una característica única y que puede ser fácilmente compartido con otros desarrolladores.

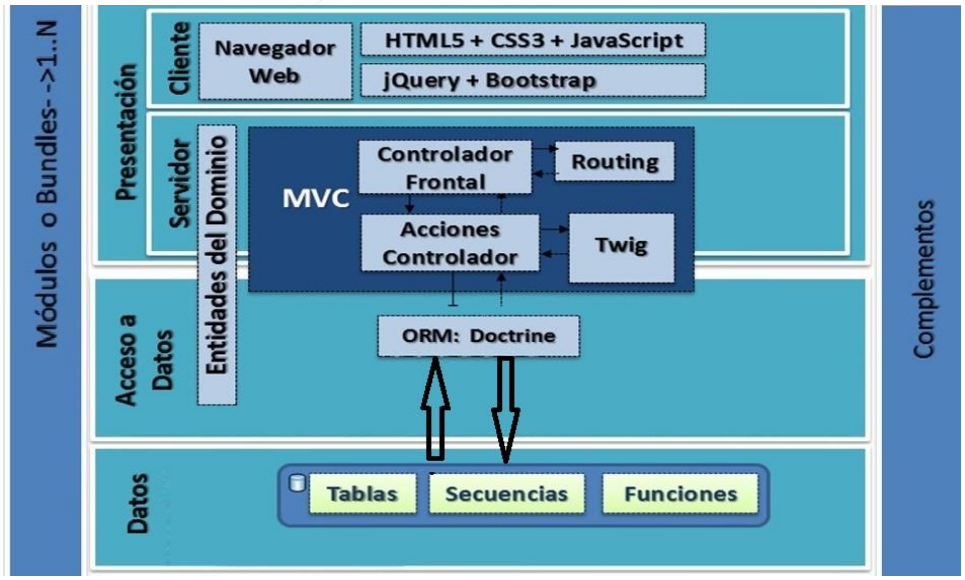



**Fig. Nº 9 Esquema de la arquitectura**

**Capa de Presentación**

Es la capa que contiene los componentes con los que va a interactuar el usuario (páginas). Permite alcanzar las funcionalidades que brinda el controlador y mostrar o capturar la información a través de los diferentes elementos que comprende: twigs y formularios. Se divide en dos sub-capas: cliente y servidor. En la primera se visualiza mediante el navegador web (utilizando tecnologías como HTML5, CSS3, JavaScript, Bootstrap y JQuery) datos procesados. En la segunda se maneja la lógica de control así como la construcción de páginas y formularios.

**Capa de Acceso a datos**

Contiene las entidades y repositorios que mapea Doctrine como marco de trabajo para la comunicación con el servidor de datos. Gestiona las peticiones de la capa de negocio consultando la base de datos y retorna los datos correspondientes.

**Capa de datos**

En esta capa se encuentra el gestor de base de datos PostgreSQL y en él un conjunto de esquemas y tablas que permiten persistir la información con la que trabaja la aplicación y manejar su almacenamiento.

**Complementos**

Los Complementos son un grupo de facilidades y mejoras que permiten lograr una arquitectura más flexible y adaptable tales como gestión de seguridad, de validaciones, de mensajería y de configuración así como tratamiento de excepciones y otras.

**Módulos (Bundles)**

Cada módulo constituye una estructura de carpetas que se organizan según la arquitectura que se describe en la Ilustración 9. Permiten utilizar funcionalidades construidas por terceros o empaquetar sus propias funcionalidades para distribuirlas y reutilizarlas.

## 2.7. Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Para la construcción de la solución se utilizan los siguientes patrones:

**Patrones GoF**

**Decorador**: permite añadir responsabilidades extra a objetos concretos de manera dinámica y transparente. Mantiene una referencia al objeto componente y define una interfaz conforme a la de dicho componente. Evita que las clases más altas en la jerarquía estén demasiado cargadas de funcionalidad y sean complejas. (56) En la aplicación el archivo layout.html.twig o plantilla global almacena el código HTML que es común a todas las páginas de la herramienta.

**Patrones GRASP**

**Experto:** asigna la responsabilidad de la creación de un objeto o la implementación de un método a la clase que conoce toda la información necesaria para crearlo. Así se obtiene un diseño con mayor cohesión. Se evidencia en las clases modelos, ejemplo: la clase Usuario, la cual contiene toda la información referente a la manipulación de los datos de los usuarios. (57)

**Alta cohesión:** la cohesión es la medida en la que un componente se dedica a realizar solo la tarea para la cual fue creado, delegando las tareas complementarias a otros componentes, (51) ejemplo: la entidad *NoConformidad* contiene varias funcionalidades estrechamente relacionadas con los datos que maneja.

**Bajo acoplamiento:** se utilizó con el propósito de disminuir la dependencia entre las clases. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases. En la herramienta la clase *PlanRevisionesController* al acceder a las entidades a través del método *getRepository* esto asegura que independientemente de la implementación de la entidad exista un grado moderado de acoplamiento entre las clases pues no existen dependencias directas entre ellas. (57)

**Creador:** el patrón creador permite identificar quién debe ser el responsable de la instanciación de nuevos objetos o clases. Da soporte al bajo acoplamiento. En Symfony2 en la clase Controller se definen y ejecutan un conjunto de acciones en las que se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Controller es "creador" de dichas entidades. (57)

**Controlador:** sirve como intermediario entre las capas presentación y acceso a datos. Ellas son las encargadas de ejecutar las funcionalidades para dar respuesta a la petición del cliente. (58) En Symfony2 todas las peticiones web son manejadas por el controlador frontal, que es el único punto de entrada de toda la aplicación en un entorno determinado.

## 2.8. Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación tienen impacto en el sistema a desarrollar. (60)

### 2.8.1. Diagramas de Clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clase son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará y los componentes que se encargarán del funcionamiento y la relación entre uno y otro. (44)

A continuación se muestra un fragmento del diagrama de clases para el caso de uso Gestionar Evaluación donde se puede observar la sección de las clases de la vista identificadas con estereotipos twig y la clase controladora. Dicho diagrama se puede consultar en el anexo 6.

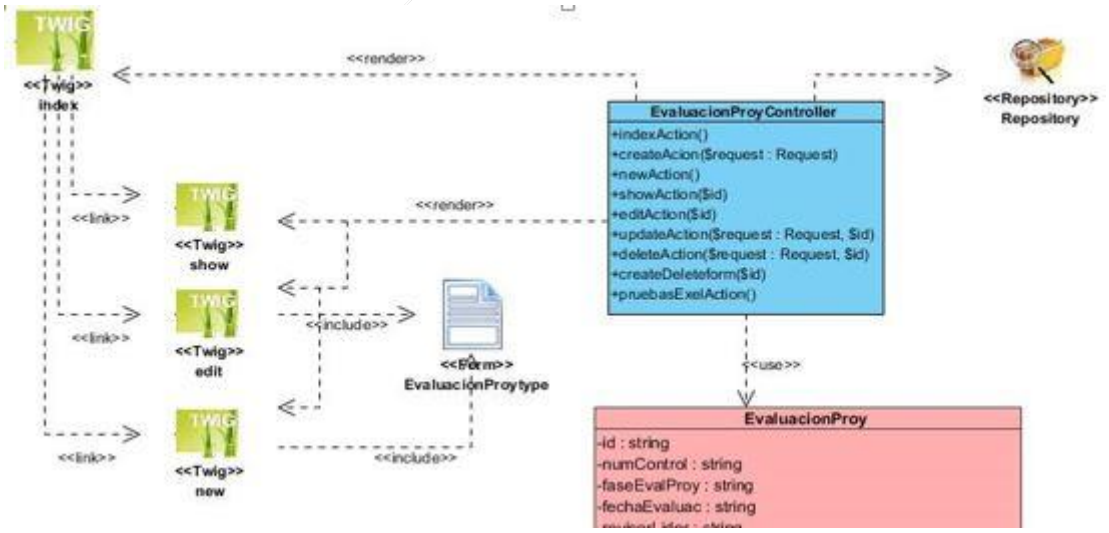



**Fig. Nº 10 Diagrama de clases del caso de uso Gestionar evaluación**

### 2.8.2. Diagramas de Interacción

Los diagramas de interacción se pueden expresar en diagramas de secuencia y en diagramas de colaboración. Muestran las interacciones entre objetos, ordenadas en secuencia temporal durante un escenario concreto, destacan la organización de las clases que participan en una interacción. En los diagramas de secuencia se modelan las interacciones entre las clases de diseño mediante la transferencia de mensajes entre objetos o subsistemas. (44)

Para la realización del caso de uso Gestionar Evaluación se determinaron varios diagramas de secuencia construidos por escenarios. En el anexo 8 se muestra el diagrama del caso de uso gestionar evaluación para el escenario adicionar evaluación, los demás pueden ser visualizados en el expediente de proyecto.

## 2.9. Modelo de datos

El modelo de datos se utiliza para describir la estructura lógica y física de la información persistente gestionada por el sistema así como la correlación entre las clases de diseño y las estructuras de datos persistentes. En otras palabras, permite describir las estructuras de datos de la base, su tipo, descripción y la forma en que se relacionan. A continuación se presenta el modelo de datos del Sistema. En el anexo 9 se puede visualizar el modelo de datos.

## 2.10. Patrones de diseño de bases de datos

Los patrones de diseño de bases de datos son plantillas que ya han sido evaluadas como las responsables de resolver un problema, son la guía para apoyarse en realizar

el trabajo. En esencia, los patrones de diseño de bases de datos constituyen la base para la búsqueda de soluciones a problemas comunes en el proceso de diseño de las mismas. (61)

**Patrón de llaves subrogadas**

Este patrón consiste en asignar una llave o identificador único para cada entidad cuyo único requisito es almacenar un valor numérico único para cada fila de la tabla, actuando como una clave sustituta, de forma totalmente independiente a los datos de negocio. (62) En la siguiente figura se puede visualizar un ejemplo del mismo.

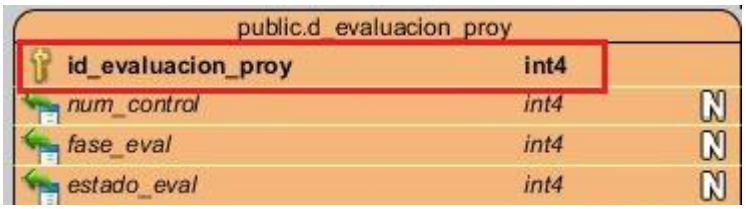



**Fig. Nº 11 Patrón de llaves subrogadas**

## 2.11. Validación del diseño

**Métricas para diseño**

**Relaciones entre clases (RC)**

La métrica RC está dada por el número de relaciones de uso de una clase con otra. Para medir el diseño según RC, se evalúan un conjunto de atributos de calidad entre los que se encuentra la Reutilización, definida en la métrica RC, además los que se relacionan a continuación:

- ✓ **Acoplamiento**: consiste en el grado de dependencia o interconexión de una clase o estructura de clases con otras, está muy ligada a la característica de Reutilización.
- ✓ **Reutilización:** consiste en el grado de reutilización presente en una clase o estructura de clases, dentro de un diseño de software.
- ✓ **Complejidad del mantenimiento**: consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ✓ **Cantidad de pruebas**: un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para determinar el grado de afectación de los atributos de calidad que mide la métrica RC es necesario determinar la cantidad de relaciones de uso (CRU) que posee cada una de las clases a medir. Una vez determinada la CRU, se procede a calcular el promedio de las mismas y teniendo ambos valores según los criterios expuestos en el capítulo 1 se determina la incidencia de los atributos de calidad en cada una de las clases.

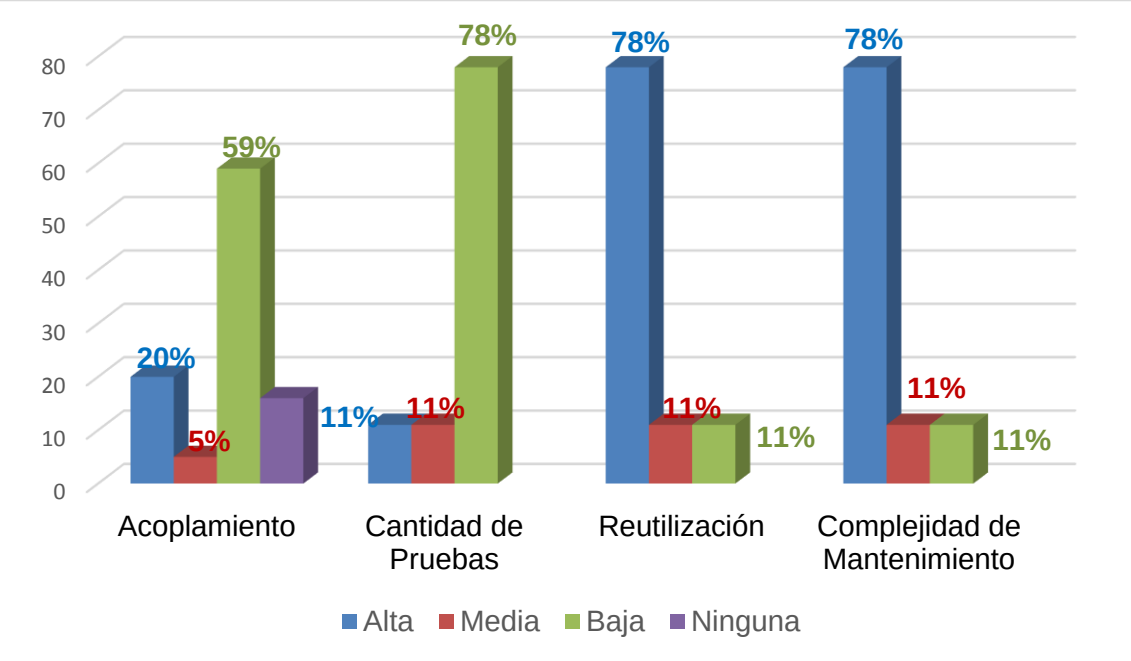



**Fig. Nº 12 Resultados de las métricas RC**

**Acoplamiento:** según los resultados que se muestran en la figura un poco más de la mitad (58%) de las clases tienen bajos valores de acoplamiento validando una realización correcta del diseño.

**Complejidad de mantenimiento:** según los resultados que se muestran en la figura se comportan de forma satisfactoria pues cerca del 70% de las clases son de fácil soporte.

**Cantidad de pruebas:** luego de aplicar la métrica se obtuvo que sólo un 11% requiere de un alto esfuerzo a la hora de realizar pruebas de unidad.

**Reutilización:** la métrica generó resultados positivos al informar que el diseño de la solución posee casi un 77% de clases altamente reutilizables.

Según lo analizado anteriormente los valores de RC se comportan de forma satisfactoria siendo discretos en la mayoría de las clases lo cual implica una disminución del

acoplamiento, así como mayor facilidad de mantenimiento de las mismas. La reutilización de las clases es alta y es factible el diseño realizado.

**Tamaño operacional de la clase (TOC)**

Al aplicar la métrica TOC se tendrá en cuenta un conjunto de atributos de calidad que se relacionan a continuación:

- ✓ **Responsabilidad**: consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio.
- ✓ **Complejidad de implementación**: consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ✓ **Reutilización**: consiste en el grado de reutilización presente en una clase o estructura de clases, dentro de un diseño de software.

La aplicación del instrumento de evaluación de la métrica TOC para el número total de operaciones arrojó los resultados que se plasman en el gráfico de la figura.

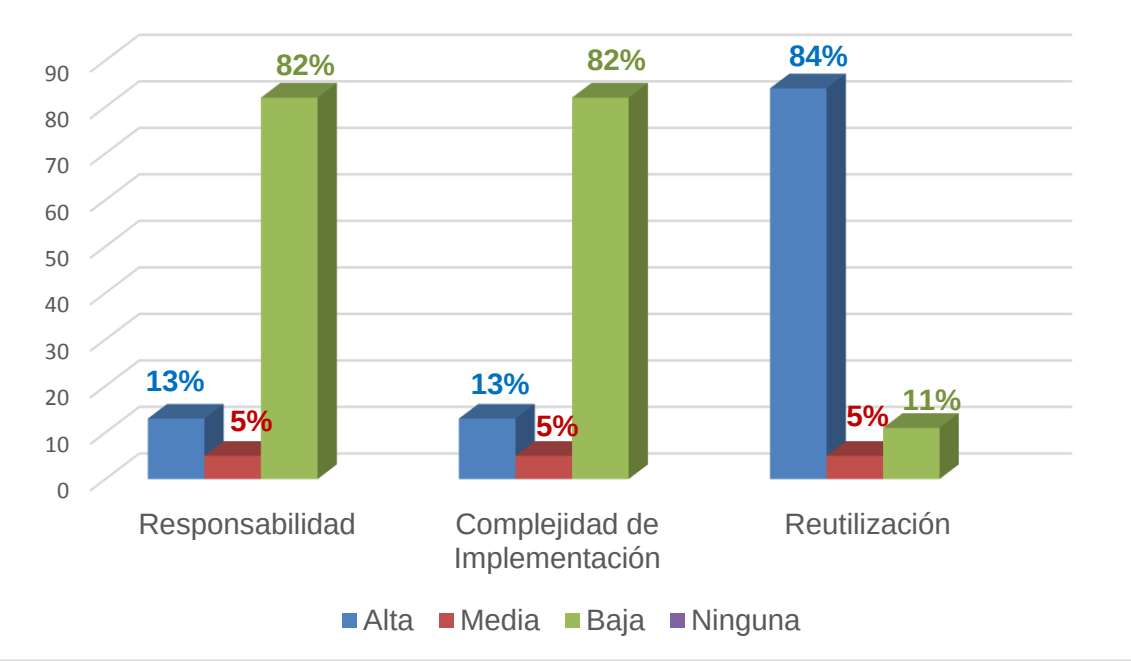



**Fig. Nº 13 Resultados de la métrica TOC**

**Responsabilidad: luego de aplicar la métrica se obtuvieron resultados satisfactorios que reflejan una responsabilidad baja con valor del 82%.**

**Complejidad de la implementación:** después de haberse realizado la medición de la métrica, arrojó también resultados positivos ya que la complejidad de las clases es baja en un 82%.

**Reutilización:** se obtuvieron valores que según muestra la gráfica de la figura anterior se comporta en un nivel alto con un 84%.

Haciendo un análisis de los resultados obtenidos para los atributos de la métrica TOC se puede observar que el atributo reutilización cuenta con un porciento alto demostrando así que el componente cuenta con una elevada reutilización, baja responsabilidad y complejidad en el diseño propuesto. Por lo que se concluye que los resultados obtenidos en esta métrica son positivos.

## 2.12. Conclusiones parciales

Los artefactos descritos en este capítulo son necesarios y de vital importancia para la construcción del sistema ya que posibilitan definirlo con suficiente detalle para permitir su interpretación y realización física. Además el proceso de ingeniería de requisitos posibilita que los requisitos de los clientes sean identificados, especificados y validados, lo cual garantizó estabilidad, adecuados niveles de corrección y carencia de ambigüedades.

El empleo de patrones de diseño garantizó una solución que tiene como premisa la reutilización de código y la solución a problemas que tienen contextos similares en el desarrollo de software. También los resultados obtenidos en la aplicación de las métricas validaron una alta reutilización de clases, el acoplamiento y la complejidad de la implementación tienen niveles bajos.

# Capítulo III Implementación y Prueba

## 3.1. Introducción

En el presente capítulo se abordan los aspectos más importantes relacionados con la implementación y pruebas del sistema, así como la validación de la investigación.

## 3.2. Diagrama de Componentes

El diagrama de componentes muestra las organizaciones y dependencias lógicas entre los componentes del software, sean estos componentes fuentes, binarios o ejecutables. Prevalecen en el campo de la arquitectura de software pero pueden ser usados para describir la vista de implementación estática de un sistema. (63)

Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. (63)

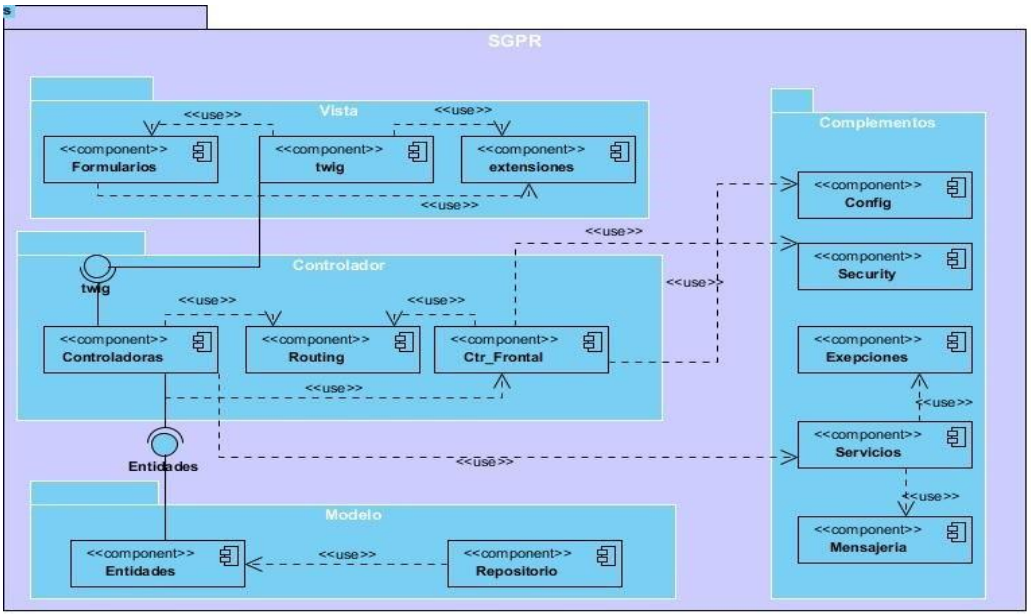



**Fig. Nº 14 Diagrama de componentes**

En el paquete SGPRC se encuentran archivos de configuración como son: Security, encargado de la seguridad del sistema; Routing, gestiona las rutas del sistema y Config posee las principales configuraciones de SGPRC. El componente Ctr_Frontal es el controlador frontal que recibe cada petición realizada al sistema. Mensajería para la

gestión de los mensajes que deben ser mostrados y Excepciones para la captura y tratamiento de las excepciones generadas.

El paquete Controlador posee el componente Controladora compuesto por las clases controladoras encargadas de dar respuesta a las peticiones de los usuarios, además se relaciona con el componente Entidades que pertenece al paquete Modelo, este contiene las clases entidades del sistema. Entidades es usado por el componente Repositorio que contiene las clases encargadas de realizar las consultas a la base de datos. El paquete Vista, contiene el componente Form encargado del trabajo con los formularios, Twig tiene como objetivo el trabajo con las interfaces de usuario y el Extensiones posee los javascript y los css. Services ofrece los servicios públicos a los que se accede desde cualquier parte del sistema SGPRC.

## 3.3. Diagrama de despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema, mostrando la configuración de los elementos de hardware (nodos) y cómo los elementos y artefactos del software se trazan en esos nodos.

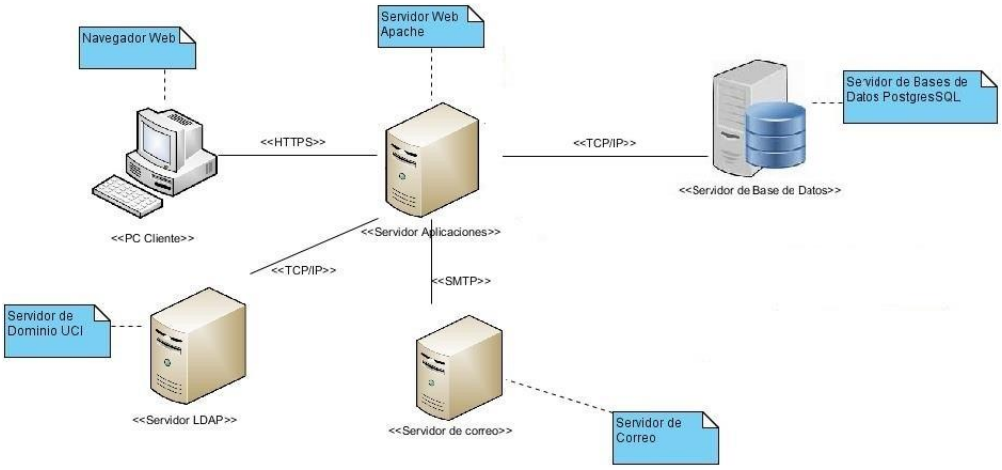



**Fig. Nº 15 Diagrama de despliegue**

A continuación se muestra la descripción de cada nodo del diagrama de despliegue:

- ✓ **Nodo PC Cliente:** ordenador desde donde los usuarios acceden a la herramienta. Debe tener el sistema operativo Linux o Windows y el navegador web Mozilla Firefox.
- ✓ **Nodo Servidor de Base de Datos:** se encontrará la base de datos que tendrá la información del sistema. El servidor debe contar con el Sistema Gestor de Bases de Datos PostgreSQL 9.1.

- ✓ **Nodo Servidor de Aplicaciones:** se encontrará todo lo concerniente a la aplicación, donde estarán agrupados los archivos a través de los cuales el usuario logra acceder al sistema, también se encuentra contenida toda la información específica de cada registro, sus clases; además almacena la configuración general del sistema, las clases y librerías externas y los plugins de instalación de la aplicación.
- ✓ **Nodo Servidor LDAP o Directorio Activo:** el directorio activo almacena información referente a los usuarios de una infraestructura y el acceso o permisos que estos tienen para interactuar o acceder al resto de los recursos de la red.
- ✓ **Nodo Servidor de Correo:** es el encargado de enviar las notificaciones generadas por el sistema.
- ✓ **HTTPS:** protocolo utilizado para la conexión entre las pc clientes y el servidor web.
- ✓ **SMTP:** protocolo utilizado para la conexión entre el servidor de aplicaciones y el servidor de correo.
- ✓ **TCP/IP:** protocolo utilizado para la conexión entre el servidor de aplicaciones y los servidores ldap, repositorio y base de datos.

## 3.4. Estándares de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código, el cual debe reflejar un estilo armonioso. Para asegurar que el desarrollo se realice de forma coordinada, mantener un código de alta calidad y legibilidad es necesario establecer un estándar de codificación, permitiendo además obtener un buen rendimiento.

**Indentación en el código (tabs o espacios):** La unidad de indentación de bloques de sentencias deben ser 4 espacios logrando así una estructura organizada y legible del sistema.

```
$entities = $em->getRepository('SGPRCBundle:Revision')->obtenerRevisionesXProyecto($idProyecto);
switch ($id) {
    case 1:
        $twig = $this->render('SGPRCBundle:Graficos:AdherenciaCriterio.html.twig', array(
            'revisiones' => $entities,
        ));
        break;
    case 2:
        $twig = $this->render('SGPRCBundle:Graficos:DistribucionImpacto.html.twig', array(
            'revisiones' => $entities,
```

**Fig. Nº 16 Estándar de Codificación Indentación**

**Nomenclatura de Clases:** en este caso se utiliza la notación PascalCase donde el identificador de la clase que esté compuesto por varias palabras juntas, inicie cada palabra con letra mayúscula.

```
class GradoImplementacionController extends Controller {
```

**Fig. Nº 17 Estándar de Codificación Nomenclatura de Clases**

**Nomenclatura según el tipo de Clase:** consiste en agregar al final del nombre de cada clase una palabra que identifique el tipo de clase. En el ejemplo, “Controller” define que esta es una clase controladora.

```
class GradoImplementacionController extends Controller {
```

**Fig. Nº 18 Estándar de Codificación Nomenclatura según el tipo de Clases**

**Nomenclatura de Variables:** el nombre de las variables se escribe en minúscula anteponiendo el símbolo “$” el cual es definido por el lenguaje PHP, en el caso que el nombre sea compuesto las palabras se escriben juntas iniciando la palabra en mayúscula excepto la primera palabra.

```
$entity = new NoConformidad();
$form = $this->createForm(new NoConformidadType(), $entity);
$form->bind($request);
```

**Fig. Nº 19 Estándar de Codificación Nomenclatura de Variables**

```
$idProyecto = $_SESSION['id_proyecto'];
$em = $this->getDoctrine()->getManager();
```

**Fig. Nº 20 Estándar de Codificación Nomenclatura de Variables**

## 3.5. Verificación del sistema

Es el proceso de comprobación donde se verifica que las funcionalidades del producto desarrollado responden datos de forma correcta. En la investigación se hace uso de las pruebas de funcionalidad realizadas a nivel de unidad, utilizando los métodos de prueba de Caja Blanca y Caja Negra para comprobar que tanto el código como la interfaz no contengan errores y se ejecutan adecuadamente.

### 3.5.1. Método de prueba de Caja Blanca

La técnica del camino básico permite obtener una medida de la complejidad de un diseño procedimental y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez. En el anexo 10 se puede visualizar el código del método datos_graficos_distribucion_impacto perteneciente al siguiente grafo de flujo.

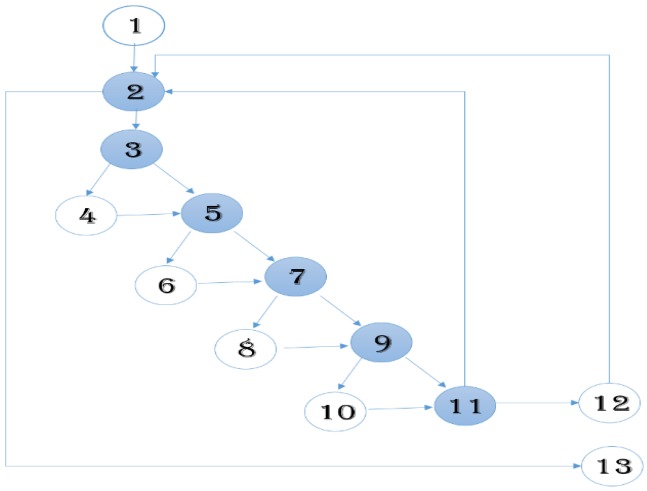



**Fig. Nº 21 Grafo de flujo del método datos_graficos_distribucion_impacto**

La complejidad ciclomática indica el número de caminos independientes que dan un valor límite para el número de pruebas que se deben diseñar y se puede determinar de la siguiente forma *V (G) = Aristas - Nodos + 2.* El grafo construido tiene un total de 18 aristas y 13 nodos, entonces:

V (G) = 18-13+2 = 7.

Una vez calculada la complejidad ciclomática se determinan un total de 7 caminos independientes. A partir de ellos y su respectivo grafo de flujo se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino. Se muestra en la tabla el caso de prueba del camino de ejecución.

| | |
|---|---|
| **Entrada** | Para que se ejecute la función, son necesarios los datos impacto y tipo de nc de la Entidad NoConformidad. |

| | |
|---|---|
| **Resultados esperados** | Obtener el valor asociado al dato 'impacto_ncValor' y a 'tipo_ncValor'. |
| **Condiciones** | if ($entity['impacto_ncValor'] == 'valor')<br>if ($entity['tipo_ncValor'] == 'valor') |

**Tabla Nº 5 Caso de prueba camino básico # 3**

Una vez ejecutados los casos de prueba diseñados se demostró que todos los caminos básicos identificados fueron probados satisfactoriamente arrojando que no existe código innecesario.

### 3.5.2. Método de prueba de Caja Negra

Para la realización de las pruebas de Caja Negra se aplica la técnica partición de equivalencia. Para proveer una guía de cómo realizar la prueba, mostrar los pasos a seguir y los datos a introducir con la finalidad de obtener el resultado esperado se realizan diseños de casos de prueba por cada requisito. Estas pruebas se aplican a todas las funcionalidades comprobando cada uno de sus campos con el objetivo de detectar errores a través de la respuesta del sistema.

Se realizan pruebas a nivel de equipo de desarrollo y luego a nivel de calidad del Centro donde se muestran las 3 iteraciones en las que se detectaron no conformidades, las que fueron resueltas en el transcurso de dichas iteraciones. Con esto se garantiza que el sistema cumple con las especificaciones definidas. En el siguiente gráfico se visualizan las no conformidades detectadas por cada iteración realizada.

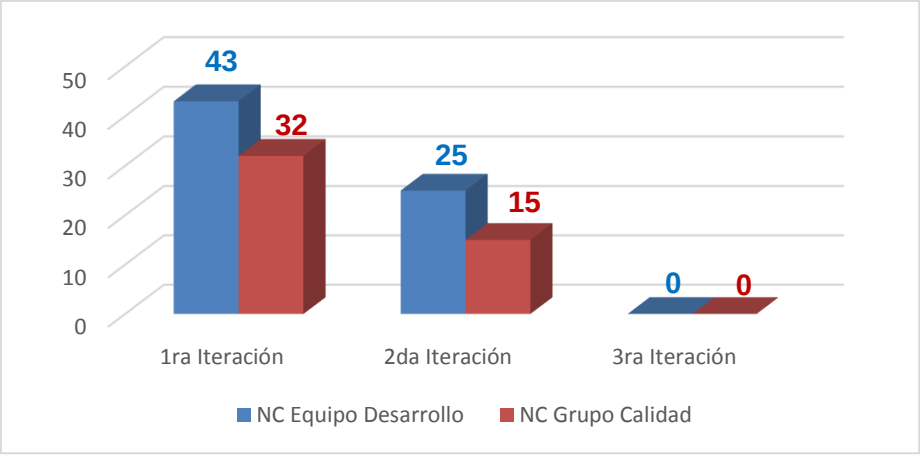



**Fig. Nº 22 Total de NC detectadas en cada iteración de pruebas a la interfaz**

## 3.6. Validación del sistema

Para la validación del sistema se realizan pruebas de aceptación, con diferentes datos de entradas para la ejecución de las mismas, identificándose resultados satisfactorios e insatisfactorios, evaluando las funcionalidades en cada una de las iteraciones. Los resultados insatisfactorios son corregidos al final de la iteración correspondiente, para de esta manera poder pasar a la próxima iteración y que el sistema evolucionara de manera estable. Son efectuadas por el cliente con el apoyo del equipo de desarrollo para confirmar que el sistema cumple con los requerimientos planteados. Las pruebas de aceptación se realizan en el entorno en que se va a utilizar el sistema pero de manera controlada, finalmente es aceptado y se puede ver el acta de aceptación en el anexo…

## 3.7. Vistas del sistema

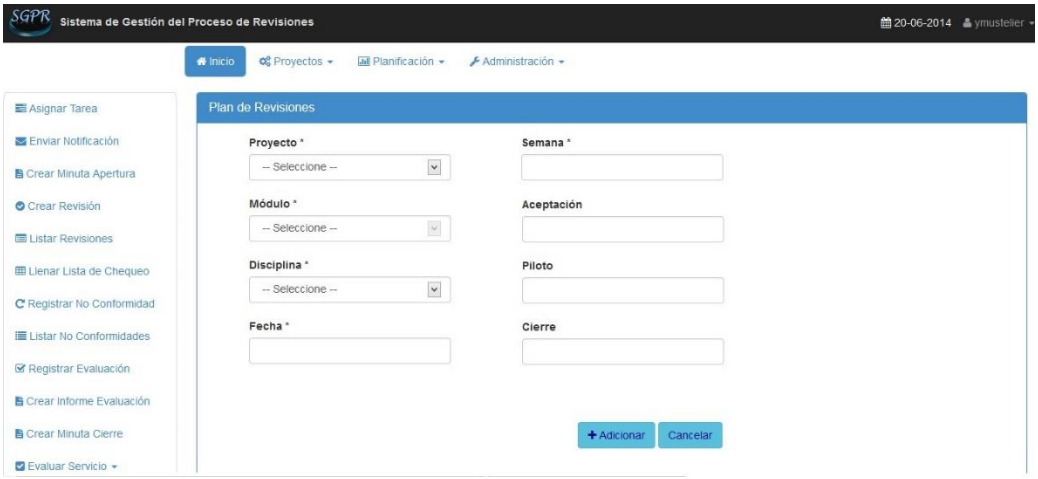



**Fig. Nº 23 Vista Realizar Plan de revisiones**

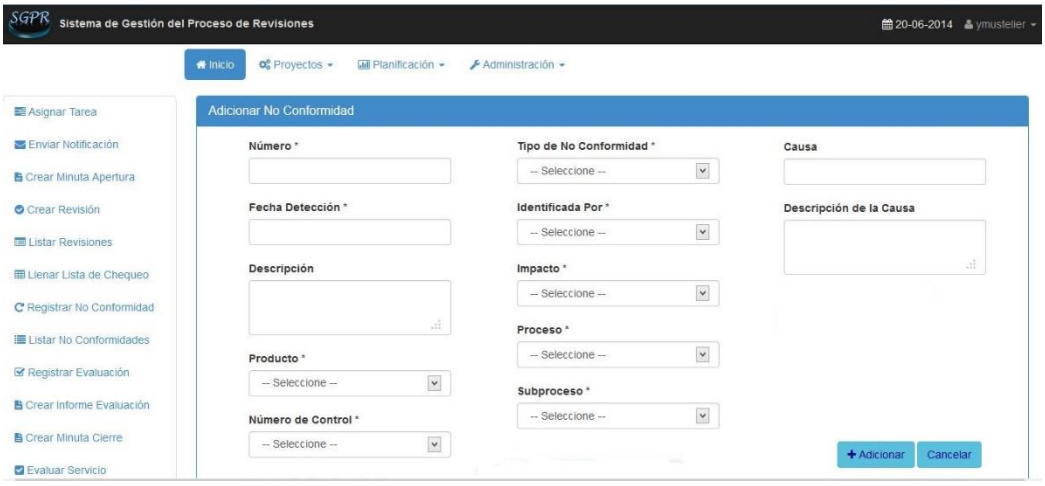



**Fig. Nº 24 Vista Adicionar no conformidad**

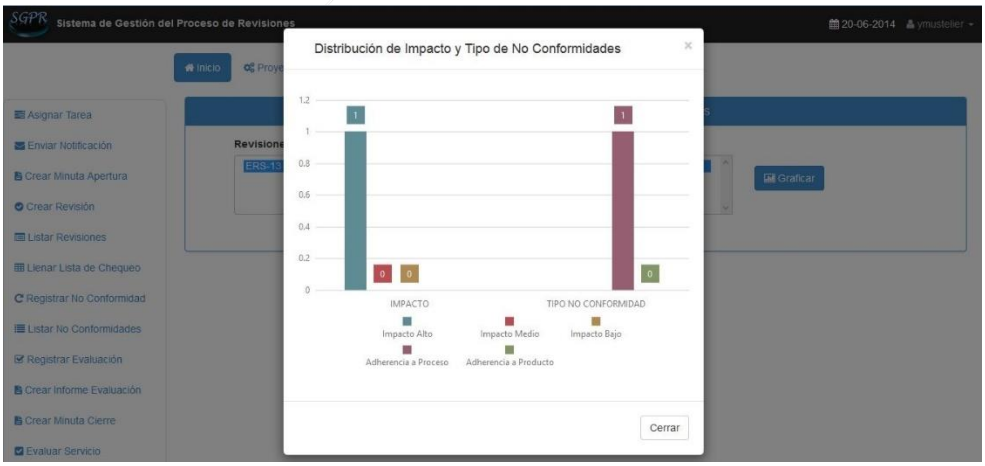



**Fig. Nº 25 Vista Generar gráfico**

## 3.8. Validación de las variables

Con el desarrollo del sistema, se mejora el proceso de Revisiones de Software en el Grupo de Calidad del Centro de Gobierno Electrónico, esto se evidencia con los siguientes elementos:

| Antes | Después |
|---|---|
| La notificación de revisión se realiza por medio del correo electrónico y de manera personal. | La notificación de revisión se realiza mediante el uso del sistema donde son registradas y enviadas permitiendo mantener el control sobre ellas. |
| El Grupo de Calidad no evalúa el servicio que brinda, por lo que carece de la posibilidad de conocer el grado de satisfacción de los clientes. | Se evalúa el servicio brindado permitiéndole al Grupo de Calidad obtener una retroalimentación de la calidad percibida por el proyecto en la realización del servicio. |
| Descentralización y falta de control en los resultados de la lista de chequeo. | A partir del uso del sistema los revisores acceden a una única lista de chequeo la cual se va actualizando a medida que van surgiendo no conformidades. Además queda registrada en el sistema permitiendo un control sobre ella. |
| Demora en el tiempo de entrega y respuesta de las no conformidades. | Las No conformidades se encuentran registradas en la aplicación y pueden ser |

| | |
|---|---|
| | vistas por los proyectos en tiempo real al ser detectadas, así como darle respuesta. |
| La generación de los gráficos para representar los resultados de los indicadores Adherencia a criterios de evaluación y Distribución de impacto y tipo de no conformidades se realiza a través de un excel que se llena de forma manual, con tendencias a pérdida de información. | A partir de los resultados de la lista de chequeo y registro de las no conformidades el sistema genera los gráficos asociados a los indicadores Adherencia a criterios de evaluación, Distribución de impacto y tipo de no conformidades, controlando en todo momento los resultados de las revisiones. |
| El Plan Anual de Revisiones se realiza haciendo uso de un excel siendo guardado en la computadora del asesor, lo que puede traer consigo pérdida de información afectando el control sobre la planificación de las revisiones. | El Plan Anual de Revisiones se gestiona mediante la aplicación y además se puede generar el excel correspondiente, permitiendo así un mayor control sobre la planificación de las revisiones. |

**Fig. Nº 26 Validación de las variables**

Por los elementos antes expuestos se demuestra el cumplimiento del problema de investigación planteado, constatando que con el desarrollo del sistema realizado, se facilita el control de las actividades en la gestión del proceso de revisiones en el Grupo de Calidad de Software del Centro de Gobierno Electrónico.

## 3.7. Conclusiones Parciales

El diagrama de componentes proporcionó una vista arquitectónica del sistema permitiendo obtener una visión lógica y estructurada del mismo. Además posibilitó lograr un proceso de implementación más organizado y productivo. El modelo de despliegue permitió obtener una visión general de la estructura que se requiere para la posterior fase de transición cuando el producto sea implantado en el entorno real de la organización. El uso de los estándares de codificación definidos permitió la obtención de un código fuente legible y fácil de entender. Además a través de las pruebas de Caja Blanca y Caja Negra se verificó la funcionalidad del sistema, se efectuaron los diseños de casos de pruebas, los cuales permitieron obtener resultados satisfactorios, teniendo buena aceptación el sistema por parte del cliente.

## Conclusiones Generales

Luego de realizado el presente trabajo de diploma se concluye que:

- ✓ El análisis de sistemas enfocados en el proceso de revisiones para el Aseguramiento de la Calidad a Proceso y Producto, demostró que ninguna se adapta a las condiciones del Grupo de Calidad y por tanto es necesario implementar una solución propia que responda a las particularidades del centro y que cumpla con el paradigma de independencia tecnológica por la que aboga el país.
- ✓ Se realizó la modelación de los procesos del negocio con lo que se logró una completa familiarización con los procesos a informatizar por parte del equipo de desarrollo logrando una profunda comprensión de los principales conceptos manejados con vista a la construcción de la solución.
- ✓ Se obtuvieron los requisitos de software a partir de los procesos modelados, los que fueron validados con el cliente, a partir de los que se definieron las principales funcionalidades a desarrollar.
- ✓ La realización del análisis y diseño garantizó el entendimiento de las funcionalidades del sistema por parte de los miembros del equipo de desarrollo. Los artefactos generados propuestos por la metodología seleccionada facilitaron la realización de la implementación.
- ✓ Se implementó la solución propuesta con lo que se le dio solución a los requisitos funcionales y no funcionales identificados. La aplicación de patrones de diseño y el uso del marco de trabajo determinado por el equipo de desarrollo, permitieron el desarrollo rápido y eficiente de la solución.
- ✓ La utilización de las pruebas de Caja Negra y Caja Blanca permitió comprobar el éxito del cumplimiento de las funcionalidades del sistema para la gestión del proceso de revisiones en el Grupo de Calidad del Centro de Gobierno Electrónico.
- ✓ Como resultado general del trabajo efectuado se le dio cumplimiento al objetivo general propuesto logrando así controlar las actividades del proceso de revisiones del Centro.

## Recomendaciones

Se considera que de forma general se cumplieron los objetivos planteados y se proponen las siguientes recomendaciones:

- ✓ Agregar funcionalidades que permitan el registro y control de trazas.
- ✓ Permitir la asignación de varios roles a un determinado usuario.
- ✓ Los centros que apliquen el mismo proceso de revisiones pueden hacer uso del sistema y comprobar su utilidad en las actividades que se realizan.

## Referencias Bibliográficas

1. Pressman, Roger S. *Ingenieria de Software. Un enfoque Práctico.* 2002.

2. Qué significa CMMI: Resumen de Aseguramiento de la calidad del proceso y producto. [En línea] [Citado el: Diciembre 03, 2013.] http://asprotech.blogspot.com/2010/08/resumen-de-aseguramiento-de-la-calidad.html.

3. *Implantación de CMMI nivel de madurez 2 en la Universidad de las Ciencias Informaticas.* 10, La Habana : Grupo Editorial Ediciones Futuro, 2010, Serie Científica de la Universidad de las Ciencias Informáticas, Vol. 3.

4. Portal de la Universidad de Las Ciencias Informáticas. [En línea] 2012. [Citado el: Junio 1, 2014.] http://www.uci.cu/?q=entorno-productivo.

5. Cueva Lovelle, Juan Manuel. *Calidad de Software.* Universidad Nacional de la Pampa : s.n., 1999.

6. Mendoza, L.E. Sistemas de Información III. [En línea] 2001. [Citado el: Marzo 12, 2014.] http://prof.usb.ve/lmendoza/Documentos/PS6117%20(Teor%EDa)/PS6117%20Calidad%20del%20Software.pdf.

7. Mendoza, L E. Sistemas de Información III. [En línea] 2001.

9. Antonio, A.D. Gestión, Control y Garantía de la Calidad de Software. [En línea] 2001. [Citado el: Marzo 12, 2014.] http://www.inf.uach.cl/rvega/asignaturas/info265/apuntes.htm.

10. Bichachi, Dra. Diana Susana. *El uso de las Listas de Chequeo (Chesk- list) como herramienta para controlar la calidad de la ley.* Humahuaca 3927 PB B-C.P.1192 - Capital Federal : s.n., 2001.

11. Asociación Española para la Calidad (AEC). [En línea] 2013. [Citado el: Junio 02, 2014.] http://www.aec.es/web/guest/centro-conocimiento/no-conformidad.

12. Portal Calidad. [En línea] [Citado el: Junio 07, 2014.] http://www.portalcalidad.com/etiquetas/361-Producto_no_conforme.

13. Xedro gespro. *Suite de Gestión de Proyectos.* [En línea] Laboratorio de Investigaciones en Gestión de Proyectos, UCI, 2006-2013. [Citado el: Junio 02, 2014.] http://gespro.cegel.prod.uci.cu/gespro/about.

14. Anido González, Yaima y Castrillo González, Yanelis. Informática Salud 2013. [En línea] 2013. [Citado el: Junio 02, 2014.] http://www.informatica2013.sld.cu/index.php/informaticasalud/2013/paper/viewFile/263/205.

15. Gabriel Vargas, César y Biagioli, Germán. Sistema para auditar el cumplimiento de CMMI-SW nivel 2. [En línea] 2009. [Citado el: Junio 02, 2014.] http://sedici.unlp.edu.ar/bitstream/handle/10915/3956/Documento_completo.pdf?sequence=15.

16. *Comercial (DC) para el Sistema de Gestión Integral de la Aduana.* Habana: Universidad de las Ciencias Informáticas : s.n., Junio 2012.

17. IBM. [En línea] [Citado el: Junio 07, 2014.] http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf.

18. Larman, Craig. UML y Patrones. 1ra Edición. México : Ed. Prentice Hall, 1999, Vol. VIII "Temas especiales", 37 "Problemas del proceso de desarrollo" p.435.

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

20. IBM. [En línea] [Citado el: Marzo 10, 2014.] http://www-01.ibm.com/software/rational/uml/.

21. PMQuality Artículos – IT Management. [En línea] Enero 31, 2014. [Citado el: Junio 07, 2014.] http://articulosit.files.wordpress.com/2014/01/bpm-vs-uml.pdf.

22. Guión Visual Paradigm for UML. [En línea] [Citado el: Febrero 19, 2014.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

23. AXURE. [En línea] [Citado el: Marzo 18, 2014.] http://www.axure.com.

24. NetBeans IDE - Overview. [En línea] [Citado el: Febrero 19, 2014.] https://netbeans.org/features/index.html.

25. Introducción a Symfony 2 | Maestros del WebMaestros del Web. [En línea] [Citado el: Febrero 18, 2014.] http://www.maestrosdelweb.com/editorial/curso-symfony2-introduccion-instalacion/.

26. Doctrine 2 ORM Documentation. [En línea] 2011. [Citado el: Febrero 18, 2014.] http://sf2-es.net16.net/_downloads/Doctrine2ORM.pdf.

27. Lenguaje de Programación | Programa.us - 1-866-TECNOLOGIA | Lenguaje, Programación, Programa, Máquina, Programas, Lenguajes. [En línea] [Citado el: Febrero 19, 2014.] http://www.programa.us/asesorias/educacion/musica/lenguaje_de_programacion.

28. PHP: ¿Qué es PHP? - Manual. [En línea] [Citado el: Marzo 18, 2014.] http://www.php.net/manual/es/intro-whatis.php.

29. The flexible, fast, and secure PHP template engine. [En línea] [Citado el: Febrero 19, 2014.] http://twig.sensiolabs.org.

30. ¿Qué es Javascript? | Maestros del Web. [En línea] [Citado el: Febrero 19, 2014.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

31. Sobre PostgreSQL. [En línea] [Citado el: Marzo 18, 2014.] http://www.postgresql.org.es/sobre_postgresql.

32. Soporte a Servidores. [En línea] [Citado el: Marzo 18, 2014.] http://www.ustamed.edu.co/sistemas/index.php/frentes/sistemas/soporte-a-servidores.

33. Welcome! - The Apache HTTP Server Project. [En línea] [Citado el: Marzo 18, 2014.] http://httpd.apache.org/.

34. Larman, Craig. UML y Patrones. Mexico : Prentice Hall, 1999.

35. Gamma, E., y otros. *Desing Patterns: Addison Wesley.* 1995.

36. Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica. [En línea] [Citado el: Marzo 30, 2014.] http://vinculando.org/articulos/sociedad_america_latina/propuesta_guia_de_medidas_para_evaluacion_sistemas_informacion.html.

37. Sánchez Fornaris. Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica. [En línea] 2010. [Citado el: Junio 1, 2014.] http://vinculando.org/articulos/sociedad_america_latina.

39. Lincke, R., Lundberg, J. y Löwe, W. *Comparing software metrics tools. International Symposium on Software Testing and Analysis. .* Berlin : s.n., 2008.

40. Desarrollo Web. [En línea] [Citado el: Abril 09, 2014.] http://www.desarrolloweb.com/articulos-copyleft/articulo-metricas-de-software.html.

41. Blanco Bueno, Carlos. *Ingeniería del Software II.*

42. Pressman, Roger S. *Ingeniería de Software: un enfoque práctico.* 6ta Edición. s.l. : McGraw-Hill, Nueva York, E.U.A. Capitulo 9, 2007.

43. Fernández Peña, J.M. Pruebas de software. [En línea] 2006. www.uv.mx/personal/jfernandez/files/2010/07/Cap3-Caminos.pdf.

44. Pressman, Roger S. *Ingenieria de Software. Un enfoque Práctico.* 2001.

45. Memorias dentro del desarrollo de software. [En línea] 01 07, 2012. [Citado el: Diciembre 05, 2013.] http://phigux.blogspot.com/2012/02/que-es-el-levantamiento-de.html.

47. Enterprise Architect - Modelo de Caso de Uso. [En línea] [Citado el: junio 18, 2014.] http://www.sparxsystems.com.ar/resources/tutorial/use_case_model.html.

48. Larman, Craig. *UML y Patrones.* s.l. : Prentice Hall, 1ra Edición, México. Parte IV Fase de Diseño, Capitulo 18 "GRASP: Patrones de asignación de responsabilidades" p.185, 1999.

49. Maza Oval, Doris y Arencibia Cruz, Humberto. *Diseño e implementación del procedimiento Ejecutivo Económico del Proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos.* La Habana : s.n., 2013.

50. Guerra Sánchez, Esther. Patrones de Diseño. *Patrón Estructural Decorator.* [En línea] 2008-09. [Citado el: Junio 01, 2014.] http://arantxa.ii.uam.es/~eguerra/docencia/0809/10%20Decorator.pdf.

51. Universidad Técnica Federico Santa María. [En línea] [Citado el: Junio 02, 2014.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08.

52. Turruelles Tejeda, Yosvani. Serie Científica de la Universidad de las Ciencias Informáticas. [En línea] 2008. [Citado el: junio 18, 2014.] publicaciones.uci.cu/index.php/SC/article/download/23/24.

53. *UML y Patrones .* 2003.

54. José García, Dr. Francisco , Conde González, Miguel Ángel y Bravo Martín, Sergio . OpenCourseWare de la Universidad de Salamanca. [En línea] Octubre 16, 2008. http://ocw.usal.es/ensenanzas-tecnicas/ingenieria-del-software/contenidos/Tema6-DOO-1pp.pdf.

55. Blaha, M. *Patterns of Data Modeling.* 2010.

56. GuilleSQL. [En línea] 2007. [Citado el: Junio 02, 2014.] http://www.guillesql.es/Articulos/Claves_Subrogadas_Slowly_Changing_Dimension_SCD_Tipo_2.aspx.

57. Arizaca Ramírez, Elisa. [En línea] [Citado el: Junio 02, 2014.] http://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=14&ved=0CFU

QFjAN&url=http%3A%2F%2Fvirtual.usalesiana.edu.bo%2Fweb%2Fpractica%2Farchiv%2Fcompon.doc&ei=7FaNU7eyJZPMsQSKhoH4CQ&usg=AFQjCNEIhax0o3YyJUQDILhXy0_IjTAQBg&bvm=bv.68191837,d.cWc.

58. Developer Network. [En línea] 2014. [Citado el: Junio 02, 2014.] http://msdn.microsoft.com/es-es/library/dd409390.aspx.

59. Estándares de codificación y buenas prácticas. [En línea] [Citado el: Junio 02, 2014.] http://blog.optimyth.com/es/2013/11/estandares-de-codificacion-y-buenas-practicas.

60. TWIG – Freecode. [En línea] [Citado el: Febrero 19, 2014.] http://freecode.com/projects/twig.

61. *TortoiseMerge - Una herramienta de diferencias/fusión para Windows.* [En línea] [Citado el: Febrero 19, 2014.] ftp://ucistore.uci.cu/software/Desarrollo/Control%20de%20versiones/Tortoise/TortoiseMerge-1.7-es.pdf.

62. Márquez Gómez, José Jorge. [En línea] [Citado el: Marzo 03, 2014.] http://jorge.queideas.com/wp-content/uploads/2011/11/Arquitectura-MVC.pdf.

63. UML y Patrones. [aut. libro] Craig Larman. [ed.] Pablo Eduardo Roig Vázquez. 1ra Edición. México : Ed. Prentice Hall, 1999, Vol. I Introducción, Capítulo 1 "Análisis y Diseño Orientados a Objetos" pag. 15.

64. Object Management Group . *Business Process Model and Notation.* [En línea] [Citado el: Marzo 10, 2014.] http://www.bpmn.org.

65. *Business Process Model and Notation .* [En línea] Enero 2011. [Citado el: Marzo 10, 2014.] http://www.omg.org/spec/BPMN/2.0.

66. Sommerville, Ian. 7ma Edición. Madrid : Pearson Educación,S.A, 2005, Vol. II Requerimientos.

67. Pressman, Roger S. 6ta Edición. 2005, Vol. II, 09 Ingeniería del Diseño.

68. Qué significa CMMI: Qué es CMMI. [En línea] [Citado el: Marzo 17, 2014.] http://asprotech.blogspot.com/2013/10/que-es-cmmi.html.

69. SISTEMA DE GESTIÓN DE INFORMACIÓN PARA LA PRESTACIÓN DE SERVICIOS DE LA EMPRESA CENEX DE CIENFUEGOS. [En línea] [Citado el: Marzo 17, 2014.] http://www.eumed.net/libros-gratis/2012a/1150/fundamentacion_teorica.html.

70. Symfony para Todos. [En línea] [Citado el: Marzo 18, 2014.] http://symfony.cubava.cu/introduccion/symfony-2/.

71. Desde Linux. [En línea] [Citado el: Marzo 18, 2014.] http://blog.desdelinux.net/hablemos-de-la-licencia-bsd/.

72. Rivera Cumbicus, Luis Miguel. Integrando las tecnologías .NET (MVC - EF) y jQuery UI para la generación automática de aplicaciones Web. [En línea] [Citado el: Marzo 18, 2014.] http://eciencia.urjc.es/bitstream/10115/7807/3/1112-MIIM-TFM-LuisMiguelRiveraCumbicus.pdf.txt.

75. Pressman, Roger S. *Ingeniería del Software. Un enfoque práctico.* 1998.

76. Research. [En línea] [Citado el: Abril 24, 2014.] http://research.microsoft.com/en-us/projects/pex/.

78. UML Quick Star. *VISUAL PARADIGM INTERNATIONAL LTD.* [En línea] 2012. [Citado el: Junio 1, 2014.] http://www.visual-paradigm.com/product/vpuml/.