

Universidad de las Ciencias Informáticas
Facultad 6



Título: Herramienta para informatizar la gestión de los subprocesos pruebas y revisiones del aseguramiento de la calidad en el Centro GEySED.

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

Autores: Yeilen Morales Aliaga.

Lisbet Daniel Rodríguez.

Tutor: Ing. Yesleny Becerra Torreira.

Co-tutor: Ing. Odiel Estrada Molina.

Asesor: Lic. Dieter Reynaldo Fuentes Cancell.

Consultante: Ing. Adrián Gracia Águila

21 de junio del 2013

Año 55 del triunfo de la Revolución

DECLARACIÓN DE AUTORÍA

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 firmo la presente a los ____ días del mes de _____ del año _____.

Yeilen Morales Aliaga

Lisbet Daniel Rodríguez

Firma del Autor

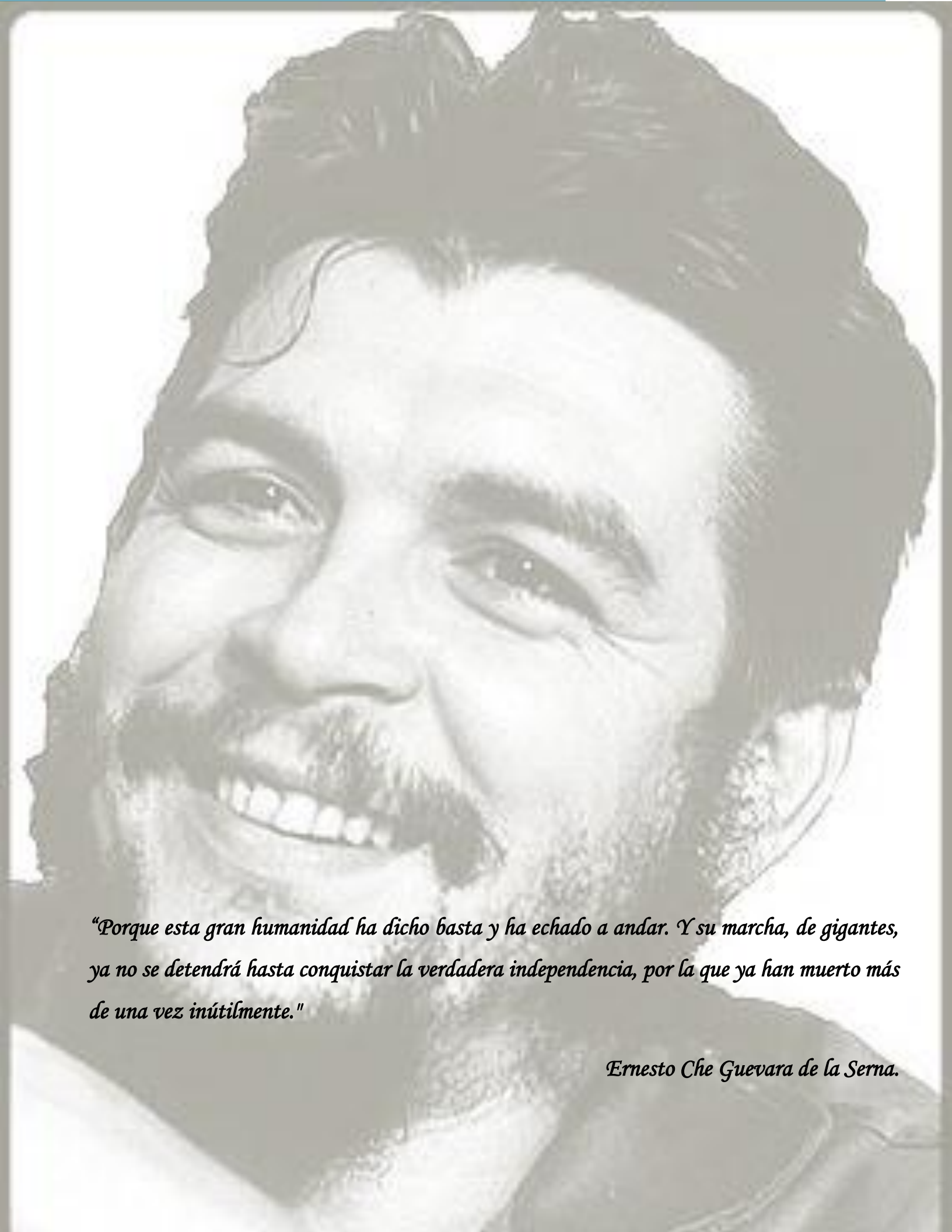
Firma del Autor

Yesleny Becerra Torreira

Odiel Estrada Molina

Firma del Autor

Firma del Cotutor



"Porque esta gran humanidad ha dicho basta y ha echado a andar. Y su marcha, de gigantes, ya no se detendrá hasta conquistar la verdadera independencia, por la que ya han muerto más de una vez inútilmente."

Ernesto Che Guevara de la Serna.

DATOS DE LOS CONTACTOS

“Tutora”

Nombre y Apellidos: Yesleny Becerra Torreira.

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniero en Ciencias Informáticas.

E-mail: ybecerra@uci.cu

Cargo: Asesora de calidad del Centro de Geoinformática y Señales Digitales (GEySED).

“Asesor”

Nombre y Apellidos: Dieter Fuentes Cancell.

Institución: Universidad de las Ciencias Informáticas.

Título: Licenciado en sociología.

E-mail: dieter@uci.cu

Cargo: Jefe de Departamento de Ciencias Sociales y Humanidades.

“Cotutor”

Nombre y Apellidos: Odiel Estrada Molina.

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniero en Ciencias Informáticas.

E-mail: ostrada@uci.cu

Cargo: Profesor.

AGRADECIMIENTOS

A todos los que han contribuido, de una forma u otra, a mi formación como profesional, especialmente:

A toda mi familia por el apoyo incondicional y la confianza que han depositado en mí, especialmente a mi madre, por el amor que me ha dado en todo momento y por ser el motor impulsor siempre. Por crear en mi ese espíritu de lucha y sacrificio, por brindarme ese hombro en el cual apoyarme cuando lo necesitaba. Estoy muy orgullosa de ti. Gracias mami.

A mi novio y más cercano colaborador Yerandy por ser mi brazo derecho durante este tiempo, por estar siempre ahí cuando lo he necesitado y por tanto cariño que me ha entregado, por aguantar en todo este tiempo mis malcriadeces y tenerme tanta paciencia.

A mis abuelos Pepe y Ofelia y a mi tía Odalis por impulsarme a seguir hacia adelante y estar siempre al tanto de mis estudios y por su apoyo incondicional.

A mis primas Yailenys y Yailén que son las hermanas que nunca tuve y siempre me han querido.

A Maydelín, Katia, por esa frase de “lo que te haga falta”, por estar siempre pendientes de mi trabajo y por su ayuda infinita.

A todos mis buenos amigos, por pasar tan agradables y a veces no tan agradables momentos, pero siempre juntos, y por hacerme saber que en cualquier momento puedo contar con ellos, al igual que ellos conmigo.

A mi asesor Dieter y cotutor Odiel que tanto que me ayudaron con sus revisiones, por aguantarme cuando me ponía malcriada o estaba muy estresada y no los dejaba trabajar; a ellos les debo mi trabajo.

A mi compañera de tesis Yeilen por formar tan buen equipo para la elaboración de este trabajo.

A mi tutora por la ayuda brindada en este tiempo.

A Carlos Luis, Adrián e Iris por aclararme todas las dudas que se me presentaban.

A todas estas personas especiales para mí, quiero darle las gracias por brindarme toda su ayuda, apoyo y comprensión para poder lograr hacer mi sueño realidad.

A todos aquellos que me ayudaron y no mencioné sin querer.

A aquellos que alguna vez me preguntaron ¿y la tesis?

AGRADECIMIENTOS

A mi Mamá:

Por confiar en mí, por apoyarme y guiarme en todo momento, por ser ese tipo de madre que todos deseamos tener que a pesar de cualquier golpe de la vida sigue a mi lado como una guía firme y sabia, gracias por todo ese cariño y apoyo que he recibido.

A mi Padre:

Por apoyarme siempre en todo, por confiar en que yo alcanzaría esta meta que hace 5 años al verme partir parecía un sueño lejano, a pesar de que hoy no está físicamente le agradezco todo su esfuerzo en ayudarme y mi mayor deseo es que esté orgulloso de mí.

A mi hermanito Yunior:

Por la confianza depositada, por todo el apoyo, por sentirse orgulloso de ser mi hermano y ser un ejemplo a seguir para él.

A mi tía Viñe, Idalmis, a mi abuela ,mi padrino, mis primas, a mi tío Noguero, a mi madrina, que aunque no estén presentes de una forma u otra contribuyeron al logro de este sueño.

A toda mi familia en general, a mi novio Osvaldo por apoyarme en este día tan especial y aconsejarme en todo momento, por todo el amor y cariño que me ha dado durante todo este tiempo “gracias mi amor”, a mis compañeros de grupo y de apartamento, profesores del proyecto, al tribunal presente, que de una forma u otra me han ayudado a lo largo de mi carrera y en la realización de este trabajo.

A todos mis amigos, en especial a Nataly, Kirenia, Darcy, a mi hada madrina de los “ppt” Yarisel, por todos los buenos y malos momentos que hemos pasado juntas.

A mi cotutor Odiel y el asesor Dieter por la ayuda incondicional que me brindaron cuando más lo necesitaba, a ellos les debo los resultados de este trabajo.

A mi compañera de tesis Lisbet por el apoyo y dedicación para el desarrollo de este trabajo.

A mi tutora por la ayuda brindada en todo momento.

A todos gracias, por ayudarme a hacer realidad mi sueño.

DEDICATORIA

DEDICATORIA

Principalmente, con mucho cariño, a mi madre Dianabel, por ser la persona que además de darme la vida, me ha guiado y ayudado a llegar hasta aquí y que sé que está orgullosa de mí y eso me hace la persona más feliz del mundo.

A aquellos que actuando desde las sombras contribuyeron a sostenerme en todo lo que he logrado.

A mis abuelos, en especial a mi abuelo Pepe que me ha malcriado muchísimo y ha sido como un padre.

A mi tía Iliana que por ser la última no es la menos importante a ella le debo este trabajo y todo mi esfuerzo durante estos cinco años, que aunque no esté físicamente conmigo siempre ha estado presente en todo lo que he hecho en cada paso y tropiezo que he dado; eres la estrella que me ha guiado para llegar a ser hoy quien soy y cumplir mi sueño.

A mi novio Yerandy que es una de las personas más importantes de mi vida y a mis amistades en especial a Alain y Eduardo por ser como mis hermanos y estar siempre presentes en todo momento cuando los he necesitado, Lincoln, Yaismel, Alleyne, Osmel, Jimm, Reina, Yeilen, Luisbel, Aindamais, Yari en fin, a todas las personas que me han aguantado y apoyado durante estos años.

A todos los presentes.

Con ustedes comparto mi triunfo, ya que a ustedes se los debo.

DEDICATORIA

A mis padres, lo más grande que tengo en este mundo, por darme las alas para perseguir mi sueño.

A mi tía Viñe, compañera y consejera de toda la vida. A mi abuela Mimi, quien hubiese sido muy feliz viéndome realizar este sueño.

A mi hermanito, mi más grande tesoro. A mi novio Osvaldo por ser una persona especial y hacerme tan feliz.

RESUMEN

El aseguramiento de la calidad del *software* (SQA) hace que todo lo creado tenga sentido, pero no se puede asegurar la calidad sino existen procesos que guíen el desarrollo de los productos de *software*. Entre los principales subprocesos que conforman el SQA se encuentran las pruebas y revisiones. En la Universidad de las Ciencias Informáticas (UCI) se encuentra el Centro de Geoinformática y Señales Digitales (GEySED), al que pertenece el Grupo Calidad que es el encargado de asesorar a los ingenieros en el cumplimiento de las normas de calidad establecidas para el desarrollo de soluciones informáticas.

La investigación surge producto de las insuficiencias del Centro GEySED pues estos subprocesos se llevan a cabo de forma manual y por tanto surge la necesidad de desarrollar una herramienta de *software* para informatizar estos subprocesos. La herramienta permite al responsable del SQA del centro las siguientes opciones: planificar estos subprocesos por cada proyecto, emitir una evaluación a nivel de proyecto y por cada subproceso, mostrar una alerta con las planificaciones realizadas, entre otras funcionalidades con el objetivo de agilizar el trabajo lucrativo del Asesor de Calidad. Para el desarrollo de la herramienta se utilizó la metodología robusta RUP, el lenguaje de programación PHP y JavaScript para el trabajo con los frameworks Symfony2 y JQuery. La herramienta fue probada utilizando las pruebas funcionales mediante los métodos de pruebas de caja negra y las pruebas de aceptación final del cliente para validar el funcionamiento de la herramienta.

Palabras claves

Aseguramiento de la calidad, pruebas, revisión.

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS

INTRODUCCIÓN	1
FUNDAMENTACIÓN TEÓRICA	6
1.1 Introducción.....	6
1.2 Conceptos asociados al SQA	6
1.3 Modelo de madurez CMMI	10
1.4 El Aseguramiento de la calidad según el modelo CMMI.....	12
1.5 Subprocesos de pruebas y revisiones del SQA.....	13
1.6 Descripción de los subprocesos de pruebas y revisiones del SQA que se desarrollan en el Centro GEySED.	14
1.7 Solución existente asociada al SQA.....	15
1.8 Conclusiones parciales	17
CARACTERÍSTICAS DEL SISTEMA	18
2.1 Introducción	18
2.2 El Proceso Unificado de Desarrollo de <i>Software</i> (RUP) como base en el desarrollo de la solución..	18
2.3 Lenguaje de modelado: UML	19
2.4 Herramientas	20
2.5 Lenguajes de programación.....	21
2.6 Framework	22
2.7 Sistema de gestión de base de datos.....	25
2.8 Herramienta para administración de datos: PgAdmin III 1.10.0.....	25
2.9 Entorno de desarrollo integrado (IDE) NetBeans 7.3.....	26
2.10 Servidor Web	27
2.11 Conclusiones parciales	27
ANÁLISIS Y DISEÑO DEL SISTEMA	28
3.1 Introducción	28
3.2 Descripción del Modelo de Negocio	28
3.3 Modelado del sistema	33

3.4 Descripción de la arquitectura de <i>software</i>	41
3.5 Patrones de diseño	44
3.6 Modelo de análisis	46
3.7 Modelo de diseño.....	47
3.8 Descripción de la herramienta propuesta.	47
3.9 Conclusiones parciales	48
IMPLEMENTACIÓN Y PRUEBA	49
4.1 Introducción	49
4.2 Diseño de la base de datos	49
4.3 Modelo de Implementación	50
4.4 Validación del sistema.....	53
4.5 Conclusiones parciales	60
CONCLUSIONES GENERALES	61
REFERENCIAS BIBLIOGRÁFICAS.....	63
ANEXOS	66

ÍNDICE DE FIGURAS

Fig. 1: Niveles de Pruebas	8
Fig. 2: Tipos de revisión.....	9
Fig. 3 Visual Paradigm 8.0.....	20
Fig. 4: Funcionamiento interno de Symfony2.....	23
Fig. 5: Arquitectura Cliente –Servidor.	42
Fig. 6: Esquema del funcionamiento interno de Symfony2 con el patrón MVC.....	43
Fig. 7: Diagrama de Despliegue.....	52
Fig. 8: Descripción del flujo de trabajo.....	54
Fig. 9: Representación de técnica de prueba de Caja negra.	56

ÍNDICE DE DIAGRAMAS

Diagrama 1: Diagrama de CU del Negocio	31
Diagrama 2: Diagrama de actividades del CU Elaborar Planificación	32
Diagrama 3: Diagrama general de objeto.....	33
Diagrama 4: Diagrama del CU del Sistema.....	37
Diagrama 5: Diagrama de clase del diseño del CU Realizar Prueba.....	47
Diagrama 6: Diagrama de entidad-relación.....	49
Diagrama 7: Diagrama de componente	51

ÍNDICE DE TABLAS

Tabla 1: Actores del negocio y su descripción.....	29
Tabla 2: Trabajadores del negocio y su descripción.....	30
Tabla 3: Descripción del caso de uso Elaborar Planificación.....	32
Tabla 4: Actores del sistema y su descripción.....	37

ÍNDICE DE FIGURAS, TABLAS Y DIAGRAMAS

Tabla 5: Descripción del CU Realizar Revisión	41
Tabla 6: Descripción de los nodos y enlaces de comunicación.....	53

Introducción

En los últimos 18 años ha aumentado considerablemente el desarrollo de productos de *software* y soluciones informáticas debido al impacto en el uso de las tecnologías de la información y las comunicaciones en la esfera económica, política y social; por consiguiente para la implementación de dichos productos se requiere de la aplicación y puesta en práctica del aseguramiento de la calidad en las producciones de *software*.

Entre los principales subprocesos que conforman el aseguramiento de la calidad, se evidencian las pruebas y las revisiones, en el caso de los proyectos de *software* de la Universidad de las Ciencias Informática (UCI); en las pruebas se evalúa la calidad de los productos entregados a la empresa responsabilizada con la gestión de la calidad (Calisoft), la cual se encuentra ubicada en dicha institución; y en las revisiones, se verifica el desarrollo y funcionamiento idóneo de cada producto informático teniendo en cuenta los parámetros e indicadores de calidad asociados a los mismos.

Existen procesos asociados a la gestión de la calidad, tales como el aseguramiento, mejora, control y planificación; factores determinantes que influyen en el desarrollo y calidad de cada producto. Una prueba fehaciente de cómo se desarrolla estos procesos en la UCI son las pruebas y revisiones que se le hacen a los productos para verificar la calidad del *software*. En la Facultad 6 se encuentra el Centro GEySED, a este pertenece el Grupo de SQA encargado del cumplimiento de la calidad requerida en el desarrollo de las soluciones informáticas que se despliegan en dicho centro.

Entre las principales funciones que lleva a cabo el grupo de SQA, se destacan las siguientes:

- Preparación del Plan SQA.
- Desarrollo de la descripción del proceso de *software*.
- Revisión de *software*.
- Actividades de verificación.
- Documentación.
- Reporte de no conformidades.

Con el propósito de identificar las deficiencias que presenta el Centro GEySED relacionadas con el SQA se llevó a cabo una investigación donde se constató un cúmulo de **insuficiencias** en el desarrollo de los subprocesos de pruebas y revisiones que conforman dicho aseguramiento.

- Se evidencian atrasos durante la entrega de los resultados en las revisiones realizadas a un conjunto de productos de *software*, propiciando inconformidad al grupo de desarrollo que esperan por dichos resultados para continuar su proceso de trabajo.

INTRODUCCIÓN

- Las evaluaciones se realizan a través de métricas e indicadores, las cuales se calculan por fórmulas matemáticas de forma manual.
- La emisión de los partes y los resultados estadísticos efectuados una vez que finaliza cada revisión, resulta muy engorroso llevarlas a cabo, ya que estas operaciones se elaboran de forma manual; propiciando la ocurrencia de algún fallo humano.
- En la actualidad cuando se desarrollan las pruebas y revisiones a un proyecto determinado, es perceptible un conjunto de no conformidades que son almacenadas en un Registro de Evaluaciones, por lo que a ese registro no se le aplica el seguimiento requerido al no existir una forma informatizada de actualizar su estado en cada una de las revisiones o pruebas que se efectúan. Este registro es único por cada proyecto y contiene toda la información de las revisiones y pruebas realizadas al mismo. Este proceso resulta engorroso para los revisores y probadores ya que la información de cada subproceso se encuentra en los repositorios de cada departamento. Igualmente dificulta la realización de reportes de interés para el centro debido a que la información no está centralizada y es necesario hacer una búsqueda por cada repositorio.

Después de haber realizado una investigación apriorística relacionada con el trabajo que desarrolla el Grupo de Calidad en el Centro GEySED; se logró determinar que existen deficiencias en la puesta en práctica de los subprocesos de pruebas y revisiones; propiciando de manera notable y como consecuencia de ello, morosidad en la gestión de los mismos; por eso se hace necesario desarrollar una herramienta que agilice la gestión de dichos subprocesos.

Una vez identificada esta problemática se realizó un análisis documental asociado a las soluciones informáticas existentes a nivel nacional e internacional relacionada con el SQA. La solución existente que se analizó fue la plataforma **GreenEvolution**, que a su vez no es aplicable en su totalidad en la gestión de los subprocesos del SQA de los proyectos de desarrollo de *software* del Centro GEySED, debido a que sus componentes son privativos.

En la UCI existe un Sistema de Gestión de Proyectos (GESPRO) para cada centro de desarrollo donde su misión es desarrollar productos, servicios y soluciones informáticas en el campo del procesamiento de Señales Digitales y la Geoinformática, contribuyendo a la formación integral de profesionales que respondan a las necesidades del progreso científico técnico y socioeconómico, permitiendo un posicionamiento en el mercado nacional e internacional.

Según Pressman identifica cuatro subprocesos que conforman el SQA: planificación, prueba, revisión y auditoría, esta última es llevado a cabo por la empresa Calisoft; el GESPRO solo implementa la

INTRODUCCIÓN

fase de planificación, además ésta no realiza las siguientes funcionalidades: emitir una evaluación a nivel de proyecto y por cada subproceso de prueba y revisión que se efectúe a cada producto de *software*, mostrar una alerta de las planificaciones de estos subprocesos de un proyecto determinado una vez que se acceda a la herramienta y llevar a cabo un seguimiento de las no conformidades encontradas en cada uno de estos subprocesos que se le efectúe a las diferentes soluciones informáticas desarrolladas en cada centro de la institución.

Analizando lo planteado anteriormente surge como **problema a resolver**: ¿Cómo agilizar la gestión de los subprocesos de pruebas y revisiones del SQA en los proyectos de desarrollo de *software* del Centro GEySED?

Para el desarrollo de la investigación y a partir de lo antes expuesto se define como **objeto de estudio**: los procesos del SQA y como **campo de acción**: los subprocesos de pruebas y revisiones en el Centro GEySED.

Se propone como **objetivo general**: desarrollar una herramienta para la informatización de los subprocesos de pruebas y revisiones del SQA en el Centro GEySED, que permita agilizar la gestión de dichos subprocesos.

Por lo que surge la siguiente **idea a defender**: con la implementación de una herramienta informática que permita la gestión de los subprocesos de pruebas y revisiones de los productos informáticos desarrollados en el Centro GEySED se contribuirá a agilizar los subprocesos del SQA de dicho centro.

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

1. Caracterización de los subprocesos y procedimientos del aseguramiento de la calidad en los proyectos del Centro GEYSED.
2. Identificación de aquellas soluciones informáticas existentes que tienen implementado los subprocesos de pruebas y revisiones asociados al SQA.
3. Selección de la metodología de desarrollo de *software*, tecnologías y herramientas necesarias para el desarrollo de la herramienta que se propone.
4. Realización de la documentación de los artefactos que intervienen en la construcción del sistema propuesto según la metodología escogida.
5. Implementación de la herramienta propuesta.
6. Validación de la herramienta informática aplicando las pruebas de validación de requisitos.

Para el desarrollo de estas tareas se decidió utilizar los siguientes métodos científicos de investigación: teóricos y empíricos.

INTRODUCCIÓN

Métodos Científicos de Investigación

Métodos teóricos

- **Método analítico-sintético:** para analizar la teoría relacionada con los subprocesos de pruebas y revisiones del SQA, permitiendo así, extraer los elementos más coherentes e importantes para la trayectoria de esta investigación.
- **Modelación:** se utiliza para el diseño de la herramienta informática a través de la elaboración de diagramas que se define según la metodología escogida, permitiendo así un mejor entendimiento de las funcionalidades definidas para dicha herramienta.

Métodos empíricos

- **Observación:** permitirá constatar el grado de cumplimiento de las funcionalidades definidas en la herramienta a partir de la aplicación de técnicas asociadas a las pruebas de validación de *software*.

Por lo que se considera como posibles resultados del presente trabajo:

Resultados esperados

1. Los documentos y artefactos asociados a la metodología de *software* durante el desarrollo de la herramienta.
2. La implementación de una herramienta para la gestión de los subprocesos de pruebas y revisiones del SQA en el Centro GEySED que permita agilizar la gestión de dichos subprocesos.

A continuación se muestra la estructura de la investigación que consta de 4 capítulos en los cuales se abordará todo el contenido necesario para realizar la investigación:

Capítulo 1. “Fundamentación Teórica”

Introducción al estudio de los subprocesos del SQA. En este capítulo se determinó los fundamentos teóricos del tema y se especificaron algunos conceptos asociados a los subprocesos del SQA.

Capítulo 2. “Herramientas y Tecnologías”

Caracterización de las diferentes tecnologías que pueden ser utilizadas en la construcción de la herramienta, a partir del estudio de las mismas. Se realiza una breve justificación con respecto a la elección de las herramientas que se utilizarán para dar cumplimiento al objetivo propuesto, así como los lenguajes de programación y la metodología de desarrollo que se utilizan para llevar a cabo los subprocesos del SQA en el Centro GEySED.

INTRODUCCIÓN

Capítulo 3. “Análisis y diseño del sistema”

Análisis de los casos de uso del sistema para diseñar las clases que serán implementadas, se representan los diagramas del diseño y los patrones de diseño.

Capítulo 4. “Implementación y prueba”

Este capítulo refleja todo lo relacionado con la implementación y prueba del sistema. Se describen los diseños de casos de pruebas y las pruebas funcionales para verificar los requisitos del *software* asociados a los métodos de pruebas de caja negra y la prueba de aceptación final del cliente.

Capítulo

1

FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El SQA actualmente constituye el eslabón principal para llevar a cabo el desarrollo de un producto de *software*, garantizando así la calidad de los mismos. En este capítulo se abordarán los principales conceptos para la comprensión del SQA, así como un análisis de las soluciones existentes que servirán como base para llevar a cabo el desarrollo de la herramienta que se propone según las necesidades del Centro GEySED.

1.2 Conceptos asociados al SQA

Los principales conceptos asociados al SQA son los siguientes: calidad, calidad de *software* y aseguramiento de la calidad de *software*.

Para obtener productos y servicios de calidad se debe asegurar su calidad desde el momento de su diseño. Un producto o servicio de calidad es el que satisface las necesidades del cliente, la **calidad** según la Organización Internacional para la Estandarización (ISO) significa: “Conjunto de propiedades y de características de un producto o servicio, que le confieren aptitud para satisfacer necesidades explícitas o implícitas”. (Minguet, 2008)

La calidad del *software* es una preocupación a la que se dedican muchos esfuerzos. Todo proyecto tiene como objetivo principal producir *software* con la mejor calidad posible, que cumpla, y si puede supere las expectativas de los usuarios, según el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) define como **calidad de software**: “Es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario” (Informática D.C, 2000).

El SQA tiene como propósito entregar a la administración una visibilidad adecuada del proceso utilizado y los productos de *software* construidos mediante acciones planificadas y sistemáticas que aseguren la calidad de dichos procesos y productos, según Lovelle define el **Aseguramiento de la Calidad de software (SQA)** como:

“Conjunto de actividades que se planifican de forma sistemática, las cuales son imprescindibles para tener confianza en que el *software* cumplirá con los requisitos de calidad. Este proceso de aseguramiento de la calidad se crea antes de comenzar a desarrollar la aplicación” (Lovellette, 1999).

En algunos casos se puede encontrar como “garantía de calidad” y no “aseguramiento” pero eso puede traer confusión con lo que también se le llama garantía del producto. Lovellette indica que mientras el *software* que se está desarrollado reúne los requerimientos y su desempeño es el esperado, es preciso que se supervisen las actividades de desarrollo del *software* y su rendimiento, en distintas oportunidades durante cada fase del ciclo de vida. Este es el papel del aseguramiento de la calidad del *software*.

Hay tres aspectos muy importantes con relación al aseguramiento de la calidad del *software*:

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

El aseguramiento de la calidad es aquel conjunto de actividades que se planifican de forma sistemática, mientras el *software* cumpla con los requisitos esperados se deben supervisar las actividades de desarrollo del *software* y su rendimiento, ya que el SQA está basado en la prevención porque resulta más rentable prevenir los fallos de calidad que corregirlos.

1.2.1 Conceptos asociados a los subprocesos de pruebas y revisiones del SQA.

Con el paso de los años se han ido desarrollando nuevas formas de mejorar las soluciones informáticas. Estas mejoras son tomadas en cuenta dada la necesidad de obtener productos (*software*) de mejor calidad. Entre las principales formas de perfeccionar la calidad de dichos productos es recomendable realizarles pruebas y revisiones de forma periódica.

Las Pruebas de *Software* son una fase importante dentro de todos los modelos conocidos como CMMI (Modelo de Madurez de Capacidades o CMM), SA-CMM (Modelo de Madurez de Capacidades para la Adquisición de *Software*), S3M (Modelo de Madurez de Capacidades para el mantenimiento del *software*) entre otros durante el ciclo de vida del *software*, pues permite verificar si los requisitos funcionales y algunos no funcionales cumplen con las expectativas del cliente. Según Minguet las **Pruebas de *software* (software testing)**: son un conjunto de actividades dentro del desarrollo de

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo (Minguet, 2008).

En la Fig.1 se muestran los niveles de pruebas.

Niveles de pruebas				
Test	Objetivo	Participantes	Ambiente	Método
Unitario	Detectar errores en los datos, lógica, algoritmos	Programadores	Desarrollo	Caja Blanca
Integración	Detectar errores de interfaces y relaciones entre componentes	Programadores	Desarrollo	Caja Blanca, Top Down, Bottom Up
Funcional	Detectar errores en la implementación de requerimientos	Testers, Analistas	Desarrollo	Funcional
Sistema	Detectar fallas en el cubrimiento de los requerimientos	Testers, Analistas	Desarrollo	Funcional
Aceptación	Detectar fallas en la implementación del sistema	Testers, Analistas, Cliente	Productivo	Funcional

Fig. 1: Niveles de Pruebas según (Minguet, 2008)

Según Becerra, la **revisión de software**: es el grado de cumplimiento de los requisitos definidos por el proyecto de desarrollo de *software* en los artefactos generados en el ciclo de vida de una solución informática, y según las especificaciones determinadas en el SQA asociado a las métricas e indicadores de calidad.

Las Revisiones son actividades de control de calidad, que permiten detectar defectos en los proyectos de *software*. Las Revisiones pueden ser de dos tipos, tal y como se muestra en la Fig.2, dinámicas y estáticas. Las primeras son las que detectan los defectos ejecutando el *software*, fundamentalmente son las ejecutadas en la fase de prueba del proyecto. Las segundas son visuales y se realizan sin necesidad de que el *software* esté ejecutándose.



Fig. 2: Tipos de revisión

Ambas son de suma importancia y una combinación adecuada puede detectar gran cantidad de defectos y por tanto mejorar la calidad del producto final.

1.2.2 Estructura de división del trabajo (EDT) del Aseguramiento de la Calidad (QA)

A continuación se especifican los entregables que conformaran el EDT del QA. En la Fig. 3 se muestra el primer nivel de la EDT, el detalle de la misma se puede encontrar en el Anexo 1.

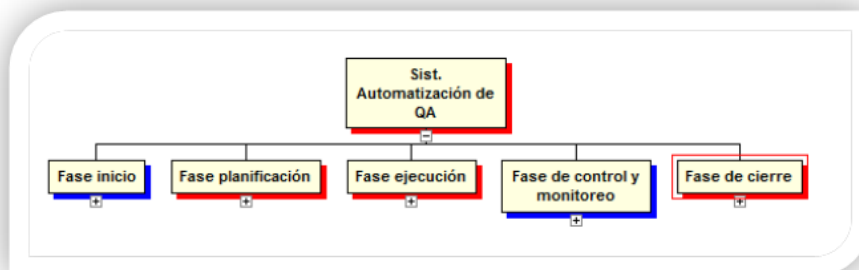


Fig. 3: Estructura de división del trabajo (EDT) del SQA (Aseguramiento de la Calidad) (Hambling, 2011).

1.2.2.1 Actividades de forma general que componen las cinco fases expuestas en el EDT

anterior:

Fase de Inicio

- Se establece la reunión inicial del proyecto.
- Se formaliza el inicio del proyecto.

Fase de Planificación

- Se identifican los requisitos funcionales detallados en el documento de requerimientos técnicos y funcionales de la herramienta a desarrollar.
- Se identifican los posibles riesgos que puedan presentarse en el proyecto.
- Se construye el Plan de Implementación y Dispersión del aplicativo.
- Se elabora el documento de logística de pruebas que se aplicará en el proyecto.
- Se elaboran los diferentes scripts de pruebas que se aplicarán en la herramienta según lo definido en el documento de logística de pruebas.

Fase Ejecución

- Coordinación para la adecuada ejecución de cada una de las actividades del proyecto.
- Ejecución de los planes establecidos en la fase de planificación.

Fase de Monitoreo y control

- Monitoreo de las actividades planeadas contra las actividades ejecutadas.
- Control sobre la ejecución de cada una de las actividades en las diferentes fases.
- Reuniones de seguimiento a lo largo del ciclo de vida del proyecto.
- Monitoreo de los recursos y de los tiempos según lo establecido en el cronograma.

Fase de Cierre

- Levantamiento de informes sobre el porcentaje de avance del proyecto.
- Recopilación y entrega de los documentos finales.
- Firma de documentos finales (Hambling, 2011).

Existen varios enfoques, o modelos para constatar el aseguramiento de la calidad de un producto. Un modelo para un Sistema de Aseguramiento de la Calidad no coloca requisitos a los procesos y actividades que se realizan en la empresa, sino al propio Sistema de Calidad. Por el hecho de proporcionar confianza, el tratamiento de un cliente a sus proveedores puede ser distinto en función del Sistema de la Calidad del cliente.

1.3 Modelo de madurez CMMI

Los factores que afectan a la calidad del *software* no cambian, por lo que resulta útil el estudio de los modelos de calidad que han sido propuestos en este sentido desde los años 70. Dado que los factores de calidad presentados para ese entonces siguen siendo válidos, se estudiarán los modelos más importantes propuestos hasta ahora: el Modelo Integrado de Capacidad y Madurez (CMMI) (Manso, 2005).

1.3.1 Características de CMMI

El CMMI es un modelo de madurez de mejora de los procesos para el desarrollo de productos y de servicios. Consiste en las mejores prácticas que tratan las actividades de desarrollo y de mantenimiento que cubren el ciclo de vida del producto, desde la concepción a la entrega y el mantenimiento. (Bañares, 2000)

Este modelo mide la madurez del desarrollo del *software* mediante 5 niveles:

- Nivel 1 (Inicial): el proceso es impredecible, es reactivo y pobremente controlado.
- Nivel 2 (Administrado): el proceso es reactivo y se caracteriza por su aplicación a proyectos.
- Nivel 3 (Definido): el proceso es proactivo y se ve a nivel de la organización.
- Nivel 4 (Administrado Cuantitativamente): el proceso es medido y controlado.
- Nivel 5 (Optimizado): el proceso se enfoca en la mejora continua.

CMMI identifica 18 áreas de procesos en sus 5 niveles. En cada área CMMI identifica objetivos generales y prácticas específicas, así como subprácticas que serían las actividades necesarias para lograr los objetivos de dicha área (Bañares, 2000).

En la Fig.4 se observan las áreas que componen los 5 niveles de CMMI:

Niveles de madurez	Áreas claves
Nivel 1 <i>Inicial</i>	Ninguna
Nivel 2 <i>Repetible</i>	Gestión de configuraciones Garantía de calidad Gestión de subcontratación del software Seguimiento y supervisión del proyecto Planificación del proyecto Gestión de requisitos
Nivel 3 <i>Definido</i>	Revisiones periódicas Coordinación entre grupos Ingeniería de productos de software Gestión de integración del software Programa de formación Definición del proceso de la organización Enfoque del proceso de la organización
Nivel 4 <i>Gestionado</i>	Gestión de calidad del software Gestión cuantitativa del proceso
Nivel 5 <i>Optimizado</i>	Gestión de cambios del proceso Gestión de cambios de tecnología Prevención de defectos

Fig. 4: Áreas que componen los 5 niveles de CMMI

La UCI está llevando a cabo un proceso de mejora encaminado a alcanzar en todos los proyectos de la universidad el nivel 2 del modelo CMMI. Mediante la gestión ágil de proyectos, que tiene como objetivos dar garantía a las cuatro demandas principales de la industria en la que se ha generado: valor, reducción del tiempo de desarrollo, agilidad y fiabilidad; y CMMI que se enfoca tanto en procesos de Administración como de Ingeniería de Sistemas y *Software*.

Las fases que constituyen el proceso de mejora son las siguientes:

- **Estudio preliminar:** en esta fase se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel.
- **Modelado del negocio:** es la fase destinada a comprender los procesos de negocio de una organización.
- **Requisitos:** el esfuerzo principal en la fase de Requisito es desarrollar un modelo del sistema que se va a construir.
- **Análisis y diseño:** durante esta fase debe considerarse necesario, a través de los modelos de análisis, los requisitos descritos durante la fase de Requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de los mismos y una descripción que sea fácil de mantener y ayude a estructurar el sistema.
- **Implementación:** en la implementación a partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares.
- **Pruebas internas:** durante esta fase el proyecto verifica el resultado de la implementación probando según sea necesaria cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas (Gracia, 2005).

1.4 El Aseguramiento de la calidad según el modelo CMMI.

CMMI contiene 22 áreas, entre ellas se encuentran: Monitorización y Control de Proyecto (PMC), Gestión de la configuración (CM), Planificación de proyecto (PP) y Aseguramiento de calidad de Procesos y Productos (PPQA), esta última tiene como objetivo que los procesos y los elementos de trabajo cumplan los procesos establecidos.

Para lo cual se deben de tener en cuenta los siguientes elementos metodológicos, según expresa PPQA (Procesos y Productos Garantía de Calidad):

- La evaluación de la ejecución de los procesos, los elementos de trabajo y servicios contra las descripciones de procesos, estándares y procedimientos.
- Identificar y documentar los elementos no conformes.
- Proporcionar información a las personas que están usando los procesos y a los gestores, de los resultados de las actividades del SQA.
- Asegurar que los elementos no conformes sean arreglados (Gracia, 2005).

1.5 Subprocesos de pruebas y revisiones del SQA

1. Pruebas de *Software*

Este proceso tiene como objetivo asegurar que el producto cumpla con lo esperado y con los requisitos definidos. Es un proceso que es transversal a todo el ciclo de vida del proyecto y presenta las siguientes políticas:

- La planificación, coordinación y ejecución de pruebas es responsabilidad de la Unidad de Control de Calidad.
- Las pruebas son ejecutadas de acuerdo al contenido del Plan de Pruebas de cada proyecto.
- Se generan reportes con observaciones y las acciones correctivas son planificadas y controladas.

La Unidad de Control de Calidad representa una función de apoyo a los proyectos de desarrollo y administra sus actividades como un proyecto independiente, aplicando las prácticas de gestión de proyectos, a saber:

- Administración de Requerimientos.
- Planificación de Proyectos.
- Control y Seguimiento.
- Aseguramiento de calidad.
- Administración de la Configuración (Gracia, 2005).

2. Revisión de *software*

La Revisión de *software* es una técnica de testing estático definido como un proceso metódico, formal y estructurado en el cual se examina un producto intermedio o final, con el propósito de encontrar y corregir defectos en forma temprana, e incluso identificar eventualmente cambios requeridos para optimizar el funcionamiento de un sistema. Las revisiones se aplican a varios momentos del desarrollo del *software* y sirven para detectar errores y defectos que pueden ser eliminados.

Los objetivos de la Revisión de *software* son:

- Descubrir errores en la función, la lógica o la implementación de cualquier representación del *software*.
- Verificar que el *software* bajo revisión alcance sus requisitos.
- Garantizar que el *software* ha sido representado de acuerdo con ciertos estándares predefinidos (Gracia, 2005).

1.6 Descripción de los subprocesos de pruebas y revisiones del SQA que se desarrollan en el Centro GEySED.

El SQA pretende dar confianza en que el producto tiene calidad a través de un conjunto de acciones planificadas y sistemáticas implantadas dentro del sistema de calidad y demostrables si es necesario, para proporcionar la confianza adecuada de que una entidad cumplirá los requisitos para la calidad (AENOR, 1995).

Según lo identifica Pressman, los subprocesos que conforman el SQA son las siguientes:

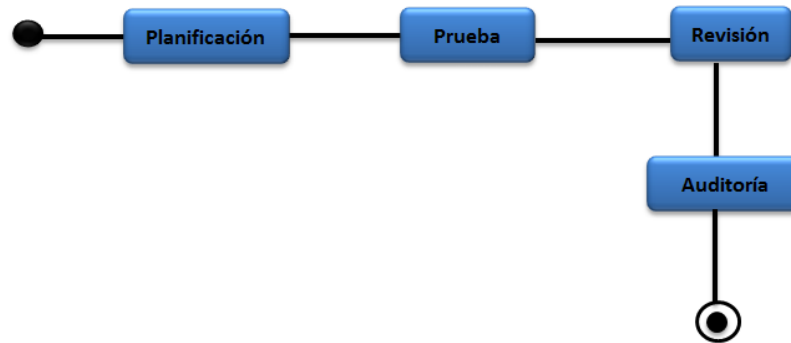


Fig. 5: Procesos que se llevan a cabo en el SQA

El grupo Calidad del Centro GEySED de la Facultad 6 de la UCI, es el encargado de realizar los subprocesos de: planificación, prueba y revisión, asociadas estas al SQA. La encargada de efectuar la auditoría como último subproceso es la empresa responsabilizada con la gestión de la calidad (Calisoft) ubicada en dicha institución, emitiendo una evaluación al final de cada auditoría que realiza. El aseguramiento de calidad en el Centro GEySED se enfoca en identificar y evaluar los defectos que puedan afectar al *software*.

Los procedimientos que se llevan a cabo en el Centro GEySED en cada fase son los siguientes:

- Planificación: Se planifica una prueba o una revisión antes de ejecutar el trabajo del proyecto, y se realiza en períodos predefinidos.
- Prueba: Consiste en comprobar que el *software* cumpla con todos los requisitos funcionales
- Revisión: Consiste en validar que el *software* cumpla con todas las funcionalidades especificadas en los casos de pruebas; de dichas validaciones surgen las no conformidades, permaneciendo plasmada en el registro de evaluaciones.

1.6.1 Especificación detallada de cómo se llevan a cabo los subprocesos de pruebas y revisiones en el centro GEySED.

Con el propósito de identificar como se realiza el SQA en el Centro de GEySED, se llevó a cabo un estudio, en la que se pudo constatar cómo se llevan a cabo los subprocesos de pruebas y revisiones del SQA; dichos subprocesos se realizan de la siguiente forma:

Planificación de la revisión y prueba:

- Se confecciona un cronograma asociado a las revisiones y pruebas de cada proyecto.
- Se realiza una reunión de inicio de establecimiento de acuerdo entre las partes, donde se le entrega al jefe de proyecto la minuta de reunión de inicio y el cronograma de la revisión.
- Luego el proyecto le entrega al especialista la documentación existente en dependencia de que sea la línea base del producto (abarca todo lo relacionado con la metodología que se lleva a cabo, que en este caso es Proceso Unificado de *Software* (RUP) o el expediente de proyecto (contiene todos los procesos que se están realizando).
- En el caso de que se vaya a realizar una prueba:
- El proyecto le instala el sistema en la computadora del especialista o procede a darle permiso en el sistema a dicho especialista.
- En caso de que se realice una revisión:
- Se copia el expediente o la línea base del producto, se hace una copia en el repositorio para revisarlo de acuerdo a lista de chequeo; luego de esa revisión se envía una lista de no conformidades al proyecto, éste tiene una cantidad de días para corregirlas y le envía la respuesta a esas no conformidades al especialista, donde éste realiza una regresión de las respuestas de estas no conformidades

Una vez identificado cómo se elabora los subprocesos de pruebas y revisiones del SQA en el Centro GEySED se verificó que estos subprocesos se realizan de forma manual, influyendo esto en el tiempo de desarrollo del *software* durante el período en que se trabaja. Además, dicho centro no cuenta con una herramienta que responda satisfactoriamente a las necesidades de los trabajadores del mismo y esto incide negativamente en el alcance de su trabajo.

1.7 Solución existente asociada al SQA

En la actualidad existe una gran variedad de *software* que se encargan de controlar el cumplimiento de los procesos del SQA, garantizando de esta forma que en cada uno de los proyectos se logre que al evaluar los procesos, planes y estándares utilizados en los mismos, cumplan con los estándares

organizacionales, siendo este el objetivo principal del SQA. Se tendrán en cuenta de acuerdo a los componentes de la plataforma GreenVolution las principales características para realizar un análisis, identificando debilidades y fortalezas que servirán de base para la propuesta a desarrollar.

1.7.1 GreenVolution. Productividad para el desarrollo de *software*

La plataforma **GreenVolution** provee herramientas para la gestión de los entregables y registros asociados a las diferentes etapas del proceso de desarrollo de *software*, tales como los requerimientos, los planes y las no conformidades. Además facilita y acelera los tiempos de implementación de modelos internacionales de calidad tales como CMMI al automatizar prácticas de verificación, validación y QA (GreenSQA, 2006).

La plataforma cuenta con 5 componentes:

- **Green Way:** Herramienta que permite personalizar el proceso particular de cada empresa para solucionar los defectos y atender los nuevos requerimientos, con base en lo cual se puede hacer seguimiento desde el registro hasta el cierre de los mismos.
- **Green Test:** Herramienta que controla y administra todas las actividades de pruebas de los productos de *software*. Brinda a los miembros del equipo de desarrollo la capacidad de validar el progreso del producto con respecto al plan de prueba.
- **Green Notes:** Herramienta que permite optimizar el esfuerzo de documentación de las funcionalidades que se incorporan en las entregas de productos de *software*, a medida que se avanza en la elaboración de su plan de desarrollo.
- **Green FAQ:** Herramienta que permite almacenar las preguntas más frecuentes, (Frequently Asked Questions) por sus siglas en inglés.
- **Green Metrics:** Herramienta que reúne la información derivada de las otras aplicaciones y con base en ella calcula y muestra el conjunto de métricas estadísticas, informes e indicadores de gestión, permitiendo a los equipos de trabajo tener retroalimentación.

La implementación de las cinco herramientas en forma conjunta, permite a las organizaciones alcanzar niveles de madurez compatibles con normas y/o modelos internacionales de calidad como ISO9000 y CMMI (Gómez, 2006).

1.7.2 Análisis de la solución existente GreenVolution

Según el análisis realizado con respecto a la herramienta mencionada anteriormente, se evidencia que en el caso de los componentes de la plataforma **GreenVolution** no están accesibles a la comunidad científica y su uso es privado, aunque se identificaron algunas funcionalidades, las cuales se presentan a continuación, que servirían de base para la implementación de la herramienta propuesta.

- Del componente **Green Test**, la funcionalidad que se va a tener en cuenta es: administrar todas las actividades de pruebas de los productos de *software*.
- Del componente **Green Notes**, la funcionalidad que se va a tener en cuenta es: optimizar el esfuerzo de documentación de las funcionalidades que se incorporan en las entregas de productos de *software* a medida que se avanza en la elaboración de su plan de desarrollo.

En consecuencia a lo expuesto anteriormente se puede decir que esta herramienta no será utilizada ya que sus componentes no son de acceso a la comunidad. En la búsqueda bibliográfica realizada no se encontró una solución que resuelva totalmente el problema planteado en este trabajo.

1.8 Conclusiones parciales

Luego del análisis realizado en el presente capítulo se puede concluir que los conceptos ayudaron a entender la importancia de la investigación. Contribuyó a un estudio de la plataforma GreenVolution relacionado con el SQA, definiéndose cuáles de sus componentes se adecuaban más al problema en cuestión; la misma no se seleccionó pues todos sus componentes son privativos. Una vez definidas las deficiencias que presenta el Centro GEySED en cuanto a la forma de ejecutar los subprocesos de pruebas y revisiones del SQA, permitió constatar que dichos subprocesos se efectúan de forma manual propiciando atraso en el período en que se trabaja.

Capítulo

2

CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se explica el uso de las herramientas y tecnologías a utilizar para desarrollar una herramienta para la gestión de los subprocesos de pruebas y revisiones del SQA en el Centro GEySED. Además se define la metodología para guiar el proceso de desarrollo de la herramienta, el lenguaje de modelado para facilitar el entendimiento del proceso a través de diagramas, el lenguaje de programación, el gestor de base de datos a utilizar, las principales librerías así como el entorno de desarrollo donde se desarrollará dicha herramienta.

2.2 El Proceso Unificado de Desarrollo de *Software* (RUP) como base en el desarrollo de la solución.

RUP además de ser una de las metodologías tradicionales más usadas posee numerosas ventajas pues se centra en la definición detallada de los procesos y tareas a realizar, en las herramientas a utilizar y requiere una extensa documentación, ya que pretende prever cualquier tipo de problema de antemano (Introduction to OMG's, 2010).

Se define a partir de tres características esenciales:

- Dirigido por los Casos de Uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

Divide además el proceso de desarrollo en ciclos, teniendo un producto al final de cada ciclo.

Este proceso de desarrollo está definido por 4 fases fundamentales:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.
- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios entregables del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.
- **Transición:** El entregable ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Cada fase del RUP concluye con un hito bien definido, en el cual se deben tomar determinadas decisiones y alcanzar las metas clave antes de pasar a la siguiente fase.

Los hitos para cada una de las fases son:

- **Inicio:** Visión de los objetivos.
- **Elaboración:** Prototipo de la arquitectura.
- **Construcción:** Capacidad operacional inicial.
- **Transición:** Liberación del producto.

Después de haber identificado las principales características de la metodología RUP se decidió utilizarla para la implementación de la herramienta, que aun siendo un proyecto a corto plazo se necesita de una completa documentación del mismo. Además tiene la particularidad de que en cada ciclo de iteración, se hace exigente el uso de artefactos, de esta forma, al finalizar el proyecto se contará con una exhaustiva información del mismo.

2.3 Lenguaje de modelado: UML

2.3.1 UML 2.0

El Lenguaje Unificado de Modelado (Unified Modeling Language, UML) se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de *software*. Es el lenguaje de modelado más conocido y utilizado en la actualidad. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Se puede aplicar en el desarrollo de *software* entregando gran variedad de formas para dar soporte a una metodología de desarrollo, pero no especifica en sí mismo qué metodología o proceso usar (Joseph, 2012.).

El nuevo estándar 2.0 del UML fue desarrollado con el objetivo de hacer el lenguaje de modelado mucho más extensible y de permitir la validación y ejecución de modelos creados mediante el UML. Ayuda a mantener la consistencia entre los artefactos del sistema: requisitos, diseños e implementaciones. Se ha convertido en un estándar para representar y modelar la información con la que se trabaja en las fases de análisis y especialmente en la fase de diseño.

Dada la selección de RUP como metodología de desarrollo de *software*, el lenguaje de modelado más eficiente y conveniente para utilizar es UML. Esta combinación de RUP con UML es la más utilizada para realizar el análisis, implementación y documentación de los sistemas orientados a objetos. Además el uso de UML como lenguaje de modelado del sistema permitirá visualizar las funciones del mismo, los procesos de negocio y los esquemas de bases de datos con los cuales el sistema interactúa.

2.4 Herramientas

2.4.1 Herramienta CASE (Ingeniería de *Software* Asistida por Computadora) Visual Paradigm 8.0

Visual Paradigm es una herramienta de diseño que hace uso del UML, soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Brinda además, la posibilidad de crear, modificar y diseñar estos diagramas con rapidez y calidad, lo que ayuda a aumentar la eficiencia del sistema de análisis y diseño de manera significativa (Manager.ORG, 2012).

Características principales:

- Tiene disponibilidad en múltiples plataformas y en múltiples versiones. Esta característica es muy importante pues Visual Paradigm está disponible para varios sistemas operativos como Windows, Linux, Unix.
- Proporciona una plataforma de modelado colaborativo para el trabajo en equipo, los miembros pueden ver y editar el mismo proyecto, o el mismo esquema, incluso de forma simultánea.
- Todos los cambios se almacenan en el servidor de Visual Paradigm en función de revisión.



Fig. 3 Visual Paradigm 8.0 (Soto, 2010)

Además se considera la herramienta de modelado más adecuada para trabajar en *software* libre. Fundamentalmente, se tiene en cuenta el proceso de migración hacia *software* libre en el que se encuentra el país.

2.5 Lenguajes de programación

2.5.1 Lenguaje del lado del servidor PHP 5

PHP es una mezcla entre interpretación y compilación para intentar ofrecer a los desarrolladores un mejor rendimiento y flexibilidad. PHP compila para el código una serie de instrucciones siempre que estas son accedidas. La concepción de lenguaje interpretado es que las instrucciones son ejecutadas una por una hasta que termina el código. Es usado principalmente en interpretación del lado del servidor para la generación de páginas web dinámicas. Las principales características de PHP son: rapidez, facilidad de aprendizaje, soporte multiplataforma tanto de diversos Sistemas Operativos como de servidores HTTP y de bases de datos destacando su conectividad con PostgreSQL. PHP se distribuye de forma gratuita bajo licencia GPL (Rosalba, 2010).

Este lenguaje se decidió utilizarlo por las siguientes ventajas:

- Puede interactuar con la mayoría de las bases de datos tales como MySQL, PostgreSQL y Oracle.
- PHP es de código abierto, le permite a los desarrolladores no depender de una compañía específica para arreglar funciones que no funcionen y no es necesario pagar dinero por actualizaciones.
- PHP es completamente expandible. Está compuesto de un sistema principal, un conjunto de módulos y una variedad de extensiones de código.
- PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz (Rosalba, 2010).

2.5.2 Características del lenguaje del lado del cliente JavaScript:

JavaScript es un lenguaje de programación interpretado, creado para que se ejecute en el navegador de cada usuario. Es utilizado en la creación de páginas web dinámicas y en la manipulación y personalización de aplicaciones. El significado de que es un lenguaje interpretado es que no es necesario compilarlo para poder ejecutarlo. Es un lenguaje de programación interpretado, se utiliza

principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámica (DOM) (Eguíluz, 2012).

Este lenguaje se decidió utilizarlo por las siguientes ventajas:

- Reducción de la carga del servidor: para realizar funciones simples como validaciones, era necesario una respuesta del servidor y con el uso de JavaScript es posible desarrollar funciones del lado del cliente para realizar estas tareas.
- Gran usabilidad: al ser compatibles con la mayoría de los navegadores es muy utilizado para desarrollar funciones y ayudar en la visualización de las páginas dinámicas.
- Fácil de aprender, rápido y potente: no es necesario poseer grandes conocimientos de programación para comenzar a utilizar este lenguaje debido a que se puede visualizar directamente en el navegador. Además, integra las propiedades de los exploradores para crear funciones muy potentes.
- No es necesario un entorno de desarrollo para programarlo, con una hoja de texto se logra la implementación de una función en JavaScript
- Basado en programación orientada a objetos aunque no incorpora la creación de clases ni la herencia (Eguíluz, 2012).

Teniendo en cuenta las características expuestas anteriormente del lenguaje PHP5 y JavaScript, se decidió utilizar ambos lenguajes para la implementación de la herramienta propuesta; el lenguaje PHP para el framework Symfony2 y el lenguaje JavaScript para el framework de presentación JQuery.

2.6 Framework

2.6.1 Symfony 2

Es un framework PHP basado en la arquitectura Model-View-Controller (MVC). Fue escrito desde un origen para ser utilizado sobre la versión 5 de PHP pues hace uso de la orientación a objetos que caracteriza a esta versión y desde la versión 2 de Symfony. Por más que Symfony puede ser utilizado para otros tipos de desarrollos no orientados a la Web, fue diseñado para optimizar el desarrollo de aplicaciones Web, proporcionando herramientas para agilizar aplicaciones complejas y guiando al desarrollador a acostumbrarse al orden y buenas prácticas dentro del proyecto (Roller, 2006).

El concepto de Symfony es no reinventar la rueda, por lo que reutiliza conceptos y desarrollos exitosos de terceros y los integra como librerías para ser utilizados por nosotros. Entre ellos encontramos que integra plenamente uno de los frameworks Object Relational Mapping (ORM) más importantes dentro de los existentes para PHP llamado Doctrine (proporciona una capa de persistencia para objetos PHP), el cual es el encargado de la comunicación con la base de datos, permitiendo un control casi total de los datos sin importar si estamos hablando de MySQL (sistema de gestión de bases de datos relacional), PostgreSQL(sistema de gestor de base de datos (SGBD) relacional orientado a objetos), SQL server (sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional), entre otros motores ya que la mayoría de las sentencias SQL no son generadas por el programador sino por el mismo Doctrine.

En cualquier caso, resulta esencial conocer cómo se aplican los principios fundamentales de la arquitectura MVC a las aplicaciones Symfony2. Observa el siguiente esquema simplificado del funcionamiento interno de Symfony2 (Desarrollo web ágil con Symfony2, 2011)

En la siguiente figura se puede observar el funcionamiento interno de Symfony2:

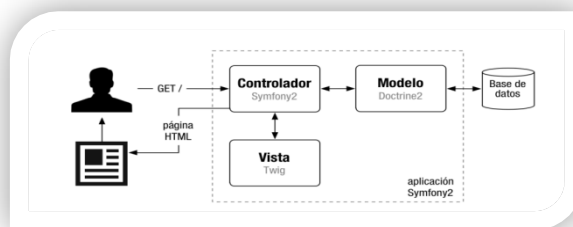


Fig. 4: Funcionamiento interno de Symfony2 (Desarrollo web ágil con Symfony2,2011)

2.6.2 Framework de presentación jQuery

jQuery es un framework JavaScript que ha tenido éxito debido a que está orientado para programadores con poca experiencia. Además posee licenciamiento bajo la Licencia MIT (Licencia que permite reutilizar el *software* así licenciado tanto para hacer *software* libre como privativo), la cual posibilita utilizarlo y liberarlo bajo BSD (Licencia que mantiene la protección de los derechos de autor, pero permite la libre redistribución y modificación, permitiéndose el uso en *software* comercial) (Manual de jQuery, 2010).

Este framework JavaScript, ofrece una infraestructura con la que se tendrá mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Con jQuery se obtendrá ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax (Asynchronous

JavaScript And XML, técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Por último posee ventajas gráficas, es multiplataforma y brinda una gran cantidad de efectos (Manual de JQuery, 2010).

2.6.2.1 JQuery UI

Es una biblioteca de componentes para el framework JQuery que le añaden un conjunto de plug-ins, widgets y efectos visuales para la creación de aplicaciones web.

La biblioteca se divide en cuatro módulos:

- Núcleo: Contiene las funciones básicas para el resto de módulos.
- Interacciones: Añade comportamientos complejos a los elementos:
- Efectos: Una API para añadir transiciones animadas y facilidades para interacciones.
- Widgets: Es un conjunto completo de controles UI. Cada control tiene un conjunto de opciones configurables y se les pueden aplicar estilos CSS. Los principales widgets que se utilizaron para la implementación de la herramienta fueron los siguientes:
 - Dialog: Ventanas con contenido.
Se utilizó en la herramienta de la siguiente forma: en las alertas de las planificaciones realizadas (las cuales son mostradas una vez que se accede a la herramienta) y en la parte de crear una minuta de reunión (en las ventanas que se muestran una vez que se realice la opción de adicionar “Acuerdos”, “Tareas”, y “Actividades”).
 - Datepicker: Calendario gráfico.
Se utilizó en la herramienta de la siguiente forma: en la parte de crear una minuta de reunión (en la ventana que se muestran una vez que se realice la opción de adicionar “Tareas” en el campo “Fecha de inicio”, el cual brinda la opción de establecer una fecha en el calendario gráfico que se muestra).

De acuerdo con las principales características del marco de trabajo Symfony2 se seleccionó este como framework a utilizarse en la programación del lado del servidor debido a que toma las mejores maneras del patrón arquitectónico modelo-vista-controlador. Además de minimizar el tiempo de desarrollo de las aplicaciones al comprender las interacciones con las aplicaciones y garantiza la seguridad al brindarle a la herramienta una estructura bien organizada.

Para la programación en el lado del cliente se utilizará como framework de presentación JQuery por poseer licencia libre, existir gran documentación tanto en inglés como en español y contar con una

comunidad en aumento. También una de sus ventajas primordiales es que integra un conjunto de componentes predefinidos para el desarrollo de aplicaciones.

1.7 Sistema de gestión de base de datos

Un sistema de gestión de base de datos (SGBD) se define como el conjunto de programas dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. (MapTools.org, 2006).

2.7.1 PostgreSQL 9.1

PostgreSQL es el sistema de gestión de base de datos relacional orientada a objeto.

Principales ventajas:

- Estabilidad, flexibilidad y alto rendimiento.
- Administración efectiva de seguridad.
- Ágil navegación y administración de base de datos.
- Excelentes herramientas visuales y de texto para elaboración de consultas.
- Se puede extender su funcionalidad.
- Permite crear o migrar aplicaciones desde Access, Visual Basic, Visual Fox Pro, y Visual C/C.
- Varias Interfaces de Programación: ODBC, JDBC, C/C++, SQL Embebido, Perl, Python, PHP.
- Impresionantes opciones de exportación e importación de datos.

Por las características presentadas anteriormente se decidió utilizar como sistema gestor de base de datos PostgreSQL, por ser altamente configurable, con lo cual puede personalizarse de acuerdo al escenario donde será utilizado y multiplataforma (cuenta con versiones para sistemas operativos Unix-like y Windows).

2.8 Herramienta para administración de datos: PgAdmin III 1.10.0

PgAdmin III es una herramienta de código abierto para la administración de bases de datos PostgreSQL, cuenta con:

- Herramienta de consulta SQL.
- Interfaz administrativa gráfica.
- Editor de código procedural.

Es considerada la más completa y popular de código abierto. Es capaz de gestionar versiones a partir de PostgreSQL 7.3 ejecutándose en cualquier plataforma. Está diseñado para responder a las

necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La conexión al servidor puede hacerse mediante conexión TCP/IP y puede encriptarse mediante SSL (Secure Sockets Layer - Protocolo de Capa de Conexión Segura) para mayor seguridad.

2.9 Entorno de desarrollo integrado (IDE) NetBeans 7.3

En la selección del Entorno de Desarrollo Integrado (IDE), se identificaron varios indicadores que determinarían su elección, como: su privacidad en cuanto al uso o no de licencias, ventajas de acuerdo al completamiento de código, lenguajes que soporta y existencia de documentación.

NetBeans IDE 7.3 es un reconocido entorno de desarrollo integrado disponible para Windows, MacOS, Linux y Solaris. Consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general para compilar cualquier tipo de aplicación. Es una herramienta para que los programadores puedan escribir, depurar y ejecutar programas. Está escrito en Java, es un producto libre y gratuito sin restricciones de uso. NetBeans IDE 7.3 es recomendable para crear aplicaciones web con la nueva versión PHP 5.3 o con la estructura Symfony (Oracle Corporation and/or its. NetBeans, 2012).

Algunas de las ventajas que posee NetBeans:

- Posee una interfaz sencilla e intuitiva, característica fundamental en un IDE.
- Posee todas las herramientas para crear aplicaciones profesionales ya sean de escritorio, empresariales, web y móviles, no solo en Java sino también en C/C++ y Ruby.
- Permite hacer diagramas utilizando la notación de UML. Propone un esqueleto para organizar el código fuente, el editor conjuntamente integra los lenguajes como HTML, JavaScript y CSS.
- El editor de PHP, es mucho más ágil y a la vez robusto, contiene más ayuda en línea, reconocimiento de sintaxis y todo lo que provee la última versión de PHP (Rolando Castellar, 2009).

Después de identificar las principales características que presenta el NetBeans 7.3 se seleccionó como IDE para la implementación de la herramienta propuesta, pues ofrece un eficiente auto-completado de código, plantillas, control de versiones. Además es multiplataforma y de código abierto.

2.10 Servidor Web

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo, de acuerdo a los comandos solicitados.

2.10.1 Servidor web Apache 2.0

Apache es un servidor web altamente configurable, robustez y estabilidad. Es un sistema de código abierto para plataformas Windows, Unix, Macintosh y otras que implementa el protocolo HTTP.

Características principales:

- Es una tecnología gratuita de código fuente abierta.
- Es personalizable, la arquitectura modular de Apache permite construir un servidor hecho a la medida.
- Es un servidor altamente configurable de diseño modular. Permite aumentar fácilmente su capacidad e instalar cualquier módulo para cumplir una función específica (Ciberaula, 2011).

Principales mejoras de Apache 2.0:

- Rapidez y estabilidad en sistemas que no son tipo Unix, tales como, OS/2 y Windows.
- Los módulos de Apache pueden escribirse para que se comporten como filtros que actúan sobre el flujo de contenidos tal y como salen del servidor, o tal y como son recibidos por el servidor.

2.11 Conclusiones parciales

Una vez definidas las herramientas y tecnologías con el objetivo de desarrollar una herramienta para la gestión de los subprocesos de pruebas y revisiones del SQA en el Centro GEySED permitió escoger las más óptimas según las características analizadas de cada una. Además de analizar como metodología a utilizar RUP, diseñada para proyectos de larga duración con el propósito de elaborar los artefactos ingenieriles necesarios para un mejor entendimiento de las funcionalidades de la herramienta. La definición del lenguaje de modelado UML y la herramienta para la modelación Visual Paradigm servirán para la confección de los diagramas del diseño y la representación del sistema. Todo lo anteriormente contribuyó a la continuidad de la investigación, permitiendo el modelamiento e implementación de la herramienta mediante el uso de las mismas.

Capítulo

3

Análisis y diseño del sistema

3.1 Introducción

En este capítulo se realiza la descripción de la herramienta que se va a desarrollar para la gestión de los subprocesos del SQA en el Centro GEySED. Para ello se describe todo lo referente al modelo de negocio en el cual se analizan todas las entidades y conceptos presentes en el entorno donde se aplica el medio. Se especifican los requisitos funcionales y los no funcionales que cumple dicho sistema. Se expone en forma de diagramas los casos de usos referentes a estos requisitos, así como sus descripciones.

3.2 Descripción del Modelo de Negocio

3.2.1 Modelo de Negocio

El uso del modelado del negocio en el proceso de desarrollo de *software* ayuda a mejorar la comprensión del problema y de su dominio, lo cual facilita la identificación, análisis y especificación de los requisitos de *software*.

El Modelado del Negocio se realiza en la primera fase de la metodología RUP -Fase de inicio, y consiste en tener un conocimiento preciso de lo que actualmente se hace en los procesos que serán considerados en el nuevo sistema.

El Modelado del Negocio tiene como objetivos:

- Comprender la estructura y la dinámica de la organización en la cual se va a implantar el sistema.
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.
- Derivar los requerimientos del sistema que va a soportar la organización (Pressman, 2001).

Las herramientas que se pueden usar en ésta etapa son: diagrama de actividades, diagrama de casos de uso, diagrama de componentes. El modelado del negocio también abarca la fase de Elaboración, en las actividades relativas al análisis funcional del sistema, pero no con la intensidad que se da en la fase de Inicio.

En resumen, el objetivo del Modelo de Negocio (Model of Business) es describir los procesos existentes u observados, con el propósito de comprenderlos. Se especifican aquí qué procesos del negocio soportará la herramienta. Además de identificar sus trabajadores, sus responsabilidades y las operaciones que llevan a cabo.

3.2.2 Actores y trabajadores del negocio

“Un **actor del negocio** es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados” (EcuRed, 2011).

Actor	Descripción
1. Jefe de departamento 2. Integrante del proyecto	<ul style="list-style-type: none"> • Es el encargado de consultar toda la información de todos los proyectos. • Es el encargado de consultar toda la información del proyecto al que pertenece.

Tabla 1 Actores del negocio y su descripción

Un **trabajador del negocio** representa un rol que juega una persona (o grupo de personas), una máquina o un sistema automatizado; actuando en el negocio. Son los que realizan las actividades, interactuando con otros trabajadores del negocio y manipulando entidades.

Trabajador	Descripción

<p>1. Asesor de calidad</p> <p>2. Revisor</p> <p>3. Probador</p> <p>4. Jefe de proyecto</p> <p>5. Especialista del proyecto</p>	<ul style="list-style-type: none"> • Es el encargado de manipular toda la información de todos los proyectos. • Es el encargado de manipular toda la información relacionada a las revisiones del proyecto que es asignado. • Es el encargado de manipular toda la información relacionada a las pruebas del proyecto que es asignado. • Es el encargado de manipular toda la información relacionada a las revisiones y a las pruebas incluyendo las no conformidades del proyecto al que pertenece pero sin eliminar ninguna información. • Es el encargado de manipular toda la información que se le especifique de un proyecto determinado.
---	---

Tabla 2 Trabajadores del negocio y su descripción

3.2.3 Procesos de negocio

“Un proceso de negocio es un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. Cada proceso de negocio tiene sus entradas, funciones y salidas. Es una colección de actividades estructurales relacionadas que producen un valor para la organización, sus inversores o sus clientes. Es, por ejemplo, el proceso a través del que una organización realiza sus servicios a sus clientes” (Gidoc Integral, 2011).

La descripción detallada de los casos de uso del negocio permite una mejor comprensión de los procesos que ocurren en este, a continuación se muestra el diagrama de casos de uso del negocio y un resumen detallado de la descripción de estos procesos. En los anexos se pueden encontrar las descripciones de los principales casos de uso del negocio que se presentan a continuación.

3.2.4 Diagrama de Casos de Uso del Negocio

CAPÍTULO: 3 ANÁLISIS Y DISEÑO DEL SISTEMA

Representa gráficamente los procesos del negocio y su interacción con los actores.

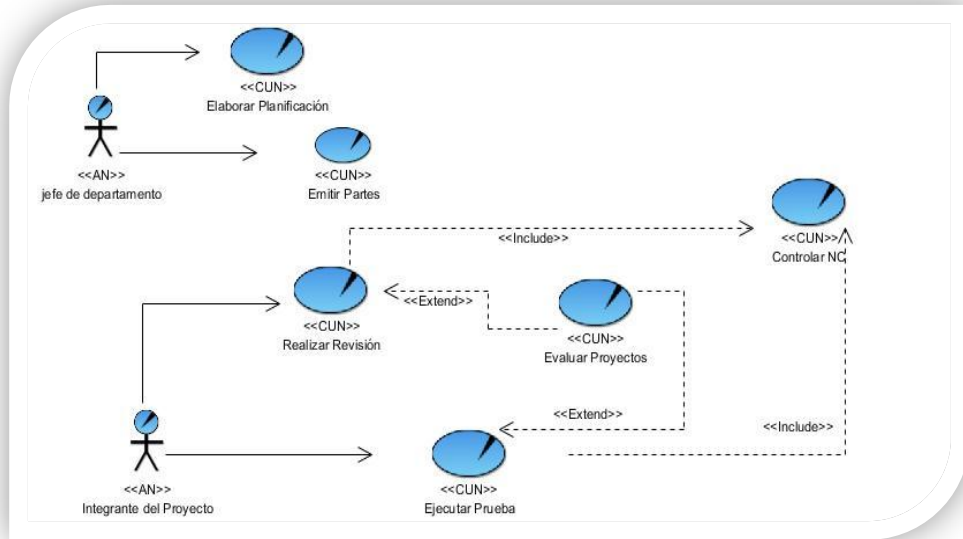


Diagrama 1: Diagrama de CU del Negocio

3.2.4 Descripción textual del CUN <Elaborar Planificación>

Caso de Uso del Negocio	Elaborar Planificación	
Actores	Jefe de departamento	
Resumen	El caso de inicio comienza cuando el actor indica cuándo debe realizarse la planificación y termina cuando se lleva a cabo dicho proceso.	
Casos de Uso asociados		
Acción del actor	Respuesta del proceso de negocio	
1. Indica al asesor de calidad cuándo es que se va a realizar la planificación		
	2. Se realiza la planificación según el proceso y el proyecto.	
	3. Cada jefe de proyecto recibe la documentación y el cronograma de tarea.	

El Diagrama de Objetos muestra las relaciones que se establecen entre los trabajadores y las entidades del negocio. Sirve para ilustrar, describir y documentar la existencia de las entidades. A continuación se muestra el diagrama de objeto de forma general:

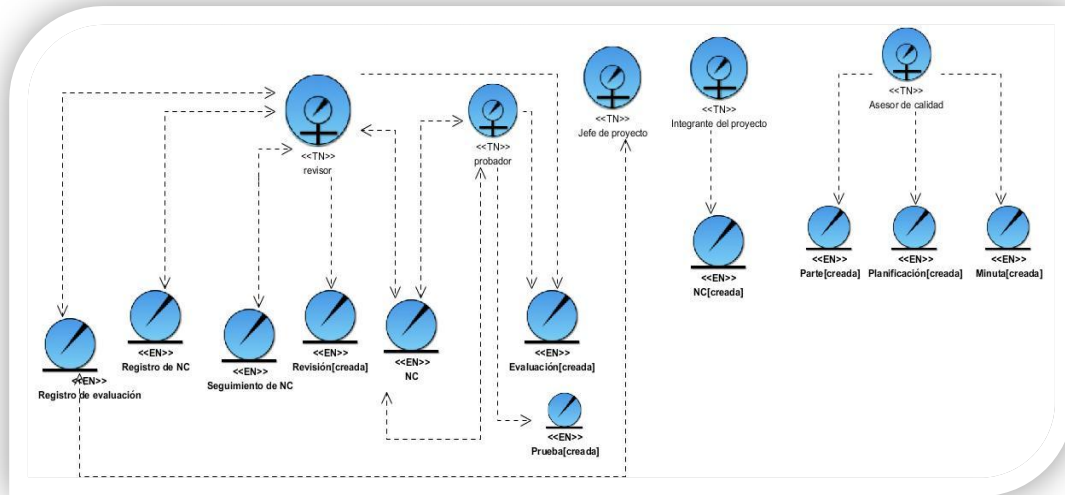


Diagrama 3: Diagrama general de objeto

3.3 Modelado del sistema

En el modelado del sistema se identifican los requerimientos funcionales y no funcionales que tendrá la herramienta, así como los actores del sistema, el diagrama de casos de uso, y la descripción de los mismos, permitiendo mejorar la comprensión de estos.

3.3.1 Levantamiento de requisitos

3.3.2 Requisitos funcionales del sistema

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen (Trujillo, 2010). La especificación de requisitos funcionales tiene el objetivo de realizar una descripción del requisito, especificar la complejidad y prioridad para el cliente. Mostrar el prototipo, los campos con sus reglas o restricciones y las observaciones.

A continuación se exponen los requisitos funcionales que la herramienta a desarrollar debe tener, la descripción de cada uno de estos requisitos funcionales se exponen en la “planilla de especificación de requisitos” que forman parte de los entregables de la tesis:

Requisitos funcionales:

- RF1. Realizar Planificación
 - 1.1. Adicionar Planificación
 - 1.2. Eliminar Planificación
 - 1.3. Modificar Planificación
 - 1.4. Listar planificación
- RF2. Mostrar Alerta de Planificación
- RF3. Realizar las NC de Pruebas
 - 3.1. Adicionar las NC de Pruebas
 - 3.2. Eliminar las NC de Pruebas
 - 3.3. Modificar las NC de Pruebas
- RF4. Mostrar Partes Semanal
- RF5. Realizar evaluación
- RF6. Exportar Planilla
- RF7. Realizar seguimiento de NC
- RF8. Realizar revisión
- RF9. Realizar minuta de reunión
- RF10. Realizar pruebas
- RF11. Realizar las NC de Revisión
 - 11.1. Adicionar las NC de Revisión
 - 11.2. Eliminar las NC de Revisión
 - 11.3. Modificar las NC de Revisión
- RF12. Controlar Proyectos
 - 12.1. Adicionar Proyectos
 - 12.2. Eliminar Proyectos
 - 12.3. Modificar Proyectos
 - 12.4. Listar proyectos
- RF13. Controlar Usuario
 - 13.1. Adicionar Usuario
 - 13.2. Eliminar Usuario
 - 13.3. Modificar Usuario
 - 13.4. Listar usuarios

- RF14. Controlar Roles
 - 14.1. Adicionar Roles
 - 14.2. Eliminar Roles
 - 14.3. Modificar Roles
 - 14.4. Listar roles

3.3.3 Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener (Trujillo. 2010).

La herramienta a desarrollar cuenta con los siguientes requisitos no funcionales:

RNF1- Disponibilidad

- El sistema debe estar disponible todo el tiempo para sus usuarios, descontando el tiempo en que se encuentre en mantenimiento.
- El período entre fallos recuperables, como por ejemplo fallos en el servidor principal no debe exceder las 24 horas.

RNF2- Soporte

- La aplicación debe estar documentada y proveer el código fuente previendo futuras modificaciones en el mismo para potenciar su alcance o eficiencia.
- Consta con la documentación necesaria para el aprendizaje sobre el uso de la herramienta informática.

RNF3- Interfaz

- Herramienta sencilla.
- Fácil de usar para el usuario.
- Funcionalidades explícitas.

RNF4- Requisitos de hardware requerido para desplegar y utilizar la aplicación web

- Se requiere contar con los periféricos mouse, teclado e impresora (opcional).
- La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo TCP/IP.

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

PC Cliente

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

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 256MB.

PC Servidor

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

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 512MB.

RNF5- Requisitos de *software*

- Servidor Web Apache 2.2 o superior, con módulo PHP 5.3.
- Navegador web Mozilla Firefox 13.0 o superior (para la pc cliente).
- El sistema debe ser multiplataforma con enfoque en tecnologías basadas en *software* no propietario.

Servidor para instalar el Gestor de Base de Datos

- SO: GNU/Linux preferentemente Ubuntu GNU/Linux 12.4 o superior.
- PostgreSQL v9.1
- Administrador de PostgreSQL: PgAdmin III.

RNF 6- Requisitos de usabilidad

- 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.

RNF 7- Seguridad

- 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.

3.3.4 Actores del sistema.

CAPÍTULO: 3 ANÁLISIS Y DISEÑO DEL SISTEMA

Los actores del sistema son terceros que interactúan con este sin ser parte del mismo. Cada trabajador del negocio es candidato a ser actor del sistema (*Software*, Curso 2009-2010).

Los actores del sistema:

- No son parte del mismo.
- Pueden intercambiar información con éste.
- Pueden representar un rol que juegan una o varias personas, un equipo o un sistema informatizado.
- Pueden ser un recipiente pasivo de información.

Los actores del sistema son los siguientes:

Actores del sistema	Descripción
1. Asesor de calidad	<ul style="list-style-type: none"> • Es el encargado de manipular toda la información de todos los proyectos. • Es el encargado de manipular toda la información relacionada a las revisiones de todos los proyectos. • Es el encargado de manipular toda la información relacionada a las pruebas de todos los proyectos.
2. Revisor	
3. Probador	

Tabla 4 Actores del sistema y su descripción

3.3.5 Diagrama de Casos de Uso del Sistema.

Un Diagrama de Casos de Uso del Sistema representa gráficamente a los procesos y su interacción con los actores (Pressman, 2009).

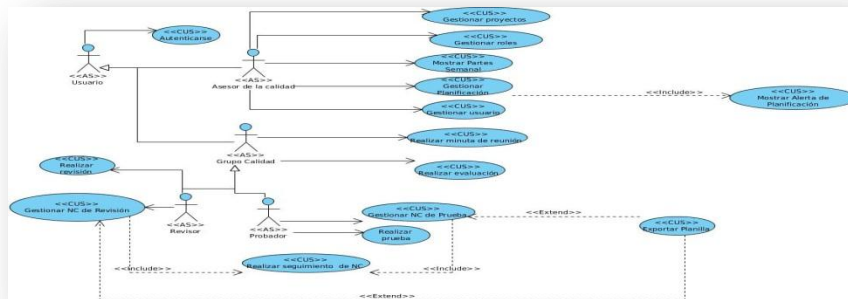


Diagrama 4: Diagrama del CU del Sistema

3.3.6 Casos de Uso del Sistema.

Los Casos de Uso del Sistema son artefactos que describen las acciones y reacciones, el comportamiento del Sistema desde el punto de vista del usuario. Establecen un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades que debe cumplir el sistema (Pressman,2009).

A continuación se muestran los casos de uso del sistema de la herramienta que se propone:

Casos de Uso del Sistema

- CU1. Gestionar Planificación (RF1.1, RF1.2 y RF1.3).
- CU2. Mostrar Alerta de Planificación. (RF2)
- CU3. Gestionar las NC de Pruebas (RF3.1, RF3.2 y RF3.3).
- CU4. Mostrar Partes Semanal. (RF4)
- CU5. Realizar evaluación.(RF5)
- CU6. Exportar Planilla. (RF6)
- CU7. Realizar seguimiento de NC. (RF7)
- CU8. Realizar revisión.(RF8)
- CU9. Realizar minuta de reunión.(RF9)
- CU10. Realizar pruebas.(RF10)
- CU11. Gestionar las NC de Revisión (RF11.1, RF11.2, RF11.3, RF11.4).
- CU12. Gestionar Proyectos (RF12.1, RF12.2, RF12.3, RF12.4).
- CU13. Gestionar Usuario (RF13.1, RF13.2, RF13.3, RF13.4).
- CU14. Gestionar Roles (RF14.1, RF14.2, RF14.3, RF14.4).

Seguidamente se evidencia la descripción de uno de los principales casos de usos del sistema, el resto se representan en los anexos:

Descripcion del CU Realizar Revisión

Objetivo	Realizar evaluación
Actores	Revisor/Probador
Resumen	El caso de uso inicia cuando el probador selecciona la opción Planificación, el sistema busca en la base de datos la lista de planificaciones de las revisiones y pruebas registradas en el sistema. El caso de uso termina cuando se lleva a cabo la evaluación de la revisión o de la prueba.

CAPÍTULO: 3 ANÁLISIS Y DISEÑO DEL SISTEMA

Complejidad	Media	
Referencias	RF9	
Prioridad	Media	
Precondiciones	El usuario está autenticado en el sistema. Existe conexión a la base de datos.	
Postcondiciones	<i>Se realizó la evaluación.</i>	
Flujo de eventos		
Flujo básico: Realizar revisión		
	Actor	Sistema
1.	Selecciona en el menú la opción "Planificación".	
2.		Muestra una interfaz donde aparece un listado con todas las planificaciones que contiene el sistema, así como las opciones (Crear una planificación, Editar y Eliminar) para la gestión de la planificación. Además muestra las opciones de Registro, No Conformidades, Evaluaciones, Seguimiento y Minuta.
3.	<p>Selecciona dentro de la opción "Revisiones" la revisión a la que desea realizarle la evaluación.</p> <p>Observación: En caso de que desee realizarle la evaluación a una prueba se realiza la misma operación, pero seleccionando la opción "Pruebas".</p>	
4.		Muestra una interfaz con la opción de calcular la evaluación.
5.	Selecciona la opción "calcular" del parámetro "Evaluación final" para evaluar la prueba o la revisión.	

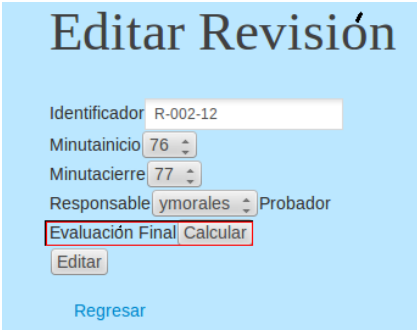
6.		<p>Busca los datos en la base de datos para realizar el cálculo según la métrica y muestra el cálculo. Termina el caso de uso.</p>
		
<p>Flujos alternos</p>		
<p>5a. Evento: Regresar</p>		
	Actor	Sistema
5 a .	<p>Selecciona la opción "Regresar"</p>	
6 a .		<p>Muestra la lista de revisiones o de pruebas existentes en el sistema. Termina el caso de uso.</p>
<p>Relaciones</p>	<p>CU Incluidos</p>	<p><i>No aplica</i></p>
	<p>CU Extendidos</p>	<p><i>No aplica</i></p>
<p>Requisitos no funcionales</p>	<p><i>RNF 3 y 6</i></p>	
<p>Asuntos pendientes</p>	<p><i>No aplica</i></p>	

Tabla 5: Descripción del CU Realizar Revisión

3.4 Descripción de la arquitectura de *software*

En el desarrollo de la herramienta propuesta se ha utilizado la arquitectura cliente-servidor, haciendo uso del patrón Modelo-Vista-Controlador, ya que Symfony está basado en dicho patrón, por sus múltiples ventajas.

3.4.1 Arquitectura cliente-servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema (Reynoso, 2004).

Las características del **cliente** (remitente de una solicitud) son:

- Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación.
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Las características del **servidor** (receptor de la solicitud) son:

- Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación.
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- No es frecuente que interactúen directamente con los usuarios finales. La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, servidor de Bases de Datos (SQL, CBASE, ORACLE), entre otros.

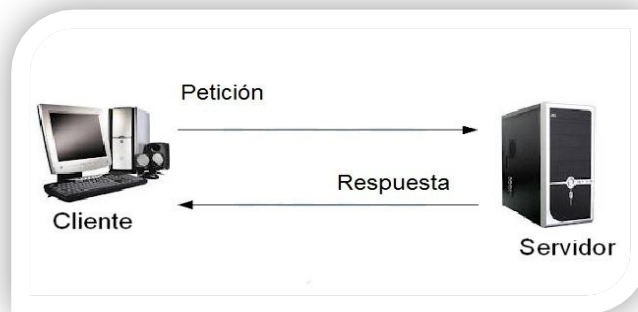


Fig. 5: Arquitectura Cliente –Servidor.

3.4.2 Patrón de arquitectura de Symfony2

La arquitectura de Symfony2 se centra en la popular arquitectura Modelo-Vista-Controlador (MVC), esta es la usada por la mayor parte de los frameworks para PHP.

Sin embargo para su creador:

Symfony2 no es un framework MVC. Symfony2 sólo proporciona herramientas para la parte del Controlador y de la Vista. La parte del Modelo es responsabilidad de uno mismo, aunque existen librerías para integrar fácilmente los ORM más conocidos, como Doctrine y Propel. (Potencier, 2011) En cualquier caso, resulta esencial conocer cómo se aplican los principios fundamentales de la arquitectura MVC a las aplicaciones Symfony2. (Eguiluz, 2011)

Esta arquitectura permite dividir nuestras aplicaciones en tres grandes capas:

- **Vista:** Todo lo que se refiera a la visualización de la información, el diseño, colores, estilos y la estructura visual en sí de nuestras páginas.
- **Modelado:** Es el responsable de la conexión a la base de datos y la manipulación de los datos mismos. Esta capa está pensada para trabajar con los datos como así también obtenerlos, pero no mostrarlos, ya que la capa de presentación de datos es la **vista**.
- **Controlador:** Su responsabilidad es procesar y mostrar los datos obtenidos por el Modelado. Es decir, este último trabaja de intermediario entre los otros dos, encargándose también de la lógica de negocio.

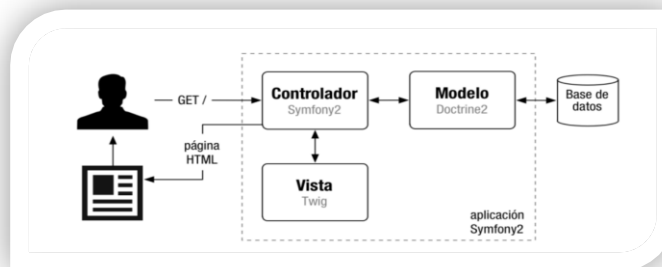


Fig. 6 Esquema simplificado del funcionamiento interno de Symfony2 con el patrón MVC (MSDN, 2006)

Cuando el usuario solicita ver la portada del sitio, internamente sucede lo siguiente:

El sistema de enrutamiento determina qué Controlador está asociado con la página de la portada.

- Symfony2 ejecuta el Controlador asociado a la portada. Un controlador no es más que una clase PHP en la que puedes ejecutar cualquier código que quieras.
- El Controlador solicita al Modelo los datos de la oferta del día. El modelo no es más que una clase PHP especializada en obtener información, normalmente de una base de datos.
- Con los datos devueltos por el Modelo, el Controlador solicita a la Vista que cree una página mediante una plantilla y que inserte los datos del Modelo.
- El Controlador entrega al servidor la página creada por la Vista. (Eguiluz, 2011)

A pesar de llegar a hacer cosas muy complejas con Symfony2, el funcionamiento interno siempre es el mismo: 1) el Controlador manda y ordena, 2) el Modelo busca la información que se le pide, 3) la Vista crea páginas con plantillas y datos (Eguiluz, 2011)

. Ventajas del MVC:

- Clara separación entre interfaz, lógica de negocio y de presentación.
- Sencillez para crear distintas representaciones de los mismos datos.
- Reutilización de los componentes.
- Más claridad de diseño.
- Facilita el mantenimiento.

Debido a las ventajas antes mencionadas se decide hacer uso de este patrón de arquitectura para lograr una mejor organización en el desarrollo de la herramienta.

3.5 Patrones de diseño

Un patrón es un esquema o micro-arquitectura que supone una solución a problemas, cada patrón es adecuado para ser adaptado a un cierto tipo de problema. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Ramiro Lago, 2012).

3.5.1 Patrón de asignar responsabilidades GRASP

Los patrones GRASP (General Responsibility Assignment Software Patterns, describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

Para la construcción de una herramienta para la gestión de los procesos de Aseguramiento de la Calidad de Software (SQA) en el centro GEYSED se definió utilizar los siguientes patrones GRASP:

- **Experto:** Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). Se evidencia en las clases modelos, ejemplo: la clase Usuario, que es la que contiene toda la información referente a la manipulación de los datos de los usuarios.
- **Alta cohesión:** este patrón define que la información que almacena una clase debe de ser coherente y está en mayor medida relacionada con la clase. Una clase con mucha cohesión es cómoda ya que es bastante fácil darle mantenimiento, comprender y reutilizar. En la herramienta queda evidenciado cuando cada clase solo se encarga de los eventos a los cuales se le ha asignado la responsabilidad, ejemplo: la clase Minuta.
- **Bajo acoplamiento:** este patrón 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 las clases ChequeoController, EvaluacionController, NoConformidadesController, ProyectoController, RegistroController, RolController, UsuarioController, DefaultController, MinutaController, PlanificacionController, PruebaController, RevisionController y SeguimientoController heredan solamente de

Controller, que es una clase estable en cuanto a su implementación por lo que se garantiza que exista un grado moderado de acoplamiento entre las clases.

- **Creador:** el patrón creador permite identificar quién debe ser el responsable de la instanciación de nuevos objetos o clases. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. (UML y patrones., 2003)

En Symfony2 en la clase Controller se definen y ejecutan todas las acciones. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Controller es "creador" de dichas entidades.

- **Controlador:** el patrón controlador se utilizó para que sirviera como intermediario entre cada una de las capas. Son todas aquellas clases controladoras declaradas en el módulo que son intermediarias entre la vista y los modelos. Ellas son las encargadas de ejecutar las funcionalidades para dar respuesta a la petición del cliente (UML y patrones, 2003).

En Symfony2 todas las peticiones web son manejadas por el controlador frontal, que es el punto de entrada único de toda la aplicación en un entorno determinado.

3.5.2 Patrones de diseño GoF

Los patrones GoF (Group of For) describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. Un patrón de diseño es una solución a un problema de diseño.

De estos patrones se emplearon para el diseño de la propuesta de solución los patrones estructurales.

Los **patrones estructurales** separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan para formar estructuras más grandes.

- **Patrón decorador**

Propósito: Añadir responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. (Facultad de informática - Universidad Politécnica de Madrid, 2005)

Este patrón añade funcionalidad a una clase de una forma dinámica y transparente. 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, para no tener que repetirlo en cada página. El contenido de las plantillas se integra en el archivo `index.html.twig`

- **Patrón fachada**

Propósito: Proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La “fachada” satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas (Facultad de informática - Universidad Politécnica de Madrid, 2005).

El uso de este patrón GoF se puede observar en la utilización de las funciones de jQuery `$get ()` y `$post ()` del archivo `Planificación.js`.

3.6 Modelo de análisis

El modelado del análisis constituye un proceso de vital importancia en el desarrollo de un *software*, representa una abstracción del sistema que se está modelando. Ayuda a comprender con una mayor profundidad acerca de los procesos que internamente se desarrollan. Fundamentalmente se basa en la comprensión de los requisitos funcionales identificados refinándolos y estructurándolos, por lo cual se facilita la visión de las funciones reales del sistema basándose en las actividades que se realizan en las fases de Diseño y la Implementación. Sin embargo este paso no es obligatorio, ya que la metodología es altamente configurable y que por tanto hay artefactos que pueden obviarse durante el desarrollo.

El modelo del análisis es opcional. En él, se describen un conjunto de Clases del Análisis que se utilizan para realizar una descripción abstracta de la realización de los casos de uso del modelo de casos de uso. Las clases del análisis luego evolucionan hacia otras clases más detalladas en el Modelo del Diseño. (El Proceso Unificado de Desarrollo de *Software.*, 2008)

Teniendo en cuenta lo anterior se decide hacer una transición del flujo de trabajo de Modelo de casos de uso al **Modelo de diseño** sin necesidad de realizar el **Modelo de análisis**. Teniendo en cuenta que Symfony2 tiene sus características relacionado con el diseño y la arquitectura de *software*, que los requisitos son bien conocidos, el hecho de que los lenguajes de programación, los frameworks y en general las tecnologías sobre las que se estará desarrollando el sistema sean ya elementos dominados, se prescindirá del flujo de trabajo Modelo de Análisis.

3.7 Modelo de diseño

En la elaboración y confección de un *software* es importante realizar previamente un correcto diseño para facilitar la implementación del mismo, ya que las bases siempre estarán fomentadas en gran medida en la eficacia de lo que se ha diseñado anteriormente. Un modelo de diseño es básicamente un modelo físico que proporcionará la estructura y la forma que tendrá el sistema que se está construyendo. Se confeccionó un diagrama de clase del diseño por cada caso de uso. En los anexos se exponen algunos diagramas de clase del diseño de los principales casos de uso.

A continuación se muestra el diagrama de diseño para el CU Realizar Prueba:

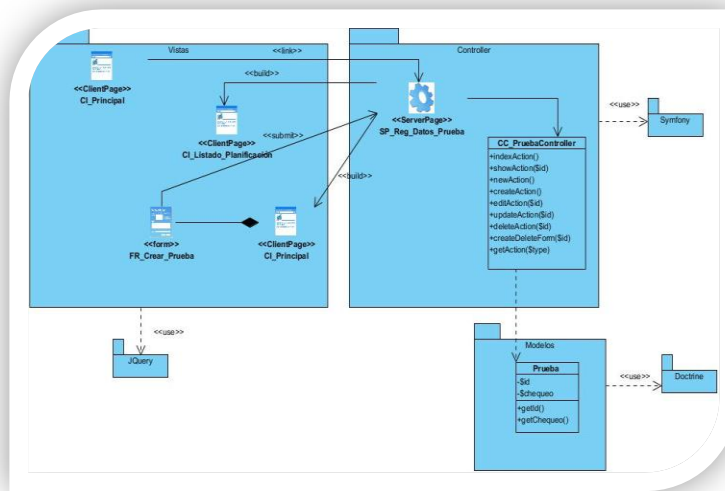


Diagrama 5 Diagrama de clase del diseño del CU Realizar Prueba

3.8 Descripción de la herramienta propuesta.

Para agilizar los procesos de SQA en el Centro GEySED se pretende desarrollar una herramienta que sea capaz de permitirles a los usuarios de cualquier proyecto de dicho centro acceder a los documentos relativos a cada uno para consultarlos, actualizarlos y/o solicitarlos.

La herramienta ofrecerá la posibilidad de hacer una búsqueda básica por contenido a preferencia del usuario para obtener el documento que desee. Una vez realizada la búsqueda se obtendrán un conjunto de resultados de acuerdo a la necesidad del mismo, y el sistema deberá brindar a los usuarios una vista previa de dichos resultados para que éste verifique cuál es el documento que desea solicitar o modificar. Además permitirá que el usuario en dependencia del rol, tenga el acceso

de actualizar el estado, así como exportar en un formato Excel cada uno de los documentos consultados.

3.9 Conclusiones parciales

Al realizar un completo análisis de las características de los procesos del negocio del SQA en el Centro GEySED, se identificaron 23 requisitos funcionales especificados en 14 casos de usos. La identificación de los actores del sistema y la descripción de los casos de usos más significativos permitieron modelar el diseño del sistema para una mejor comprensión de cómo va a estar implementada la herramienta. La selección de la arquitectura cliente-servidor permitió realizar la estructura lógica de la herramienta permitiendo agregar y modificar nuevas funcionalidades sin afectar la solución implementada.

4.2.1 Descripción del diagrama entidad-relación

El diagrama entidad-relación está compuesto por 19 tablas las cuales están relacionadas entre sí. Las principales clases que serán descritas son las siguientes:

Clase	Atributos
1. usuario_rol	<ul style="list-style-type: none"> rol_id (es el identificador del usuario_rol) usuarioid (es el identificador del usuario) que viaja como llave foránea. rolid (es el identificador del usuario) que viaja como llave foránea
2. planificación	<ul style="list-style-type: none"> id (es el identificador de la planificación) proyecto_id (es el identificador del usuario) que viaja como llave foránea fecha_inicio (atributo que pertenece a la clase) fecha_fin (atributo que pertenece a la clase) responsable (atributo que pertenece a la clase)

4.3 Modelo de Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros. (Jacobson, 2000.)

4.3.1 Diagrama de Componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación.

También se utilizan para mostrar las dependencias de compilación de ficheros, relaciones de derivación entre los ficheros que son resultado de la compilación, dependencias entre elementos de implementación y los correspondientes elementos de diseños que son implementados. (Booch, Grady, Rumbaugh, James y Jacobson, 1999)

A continuación se muestra el diagrama de componente:

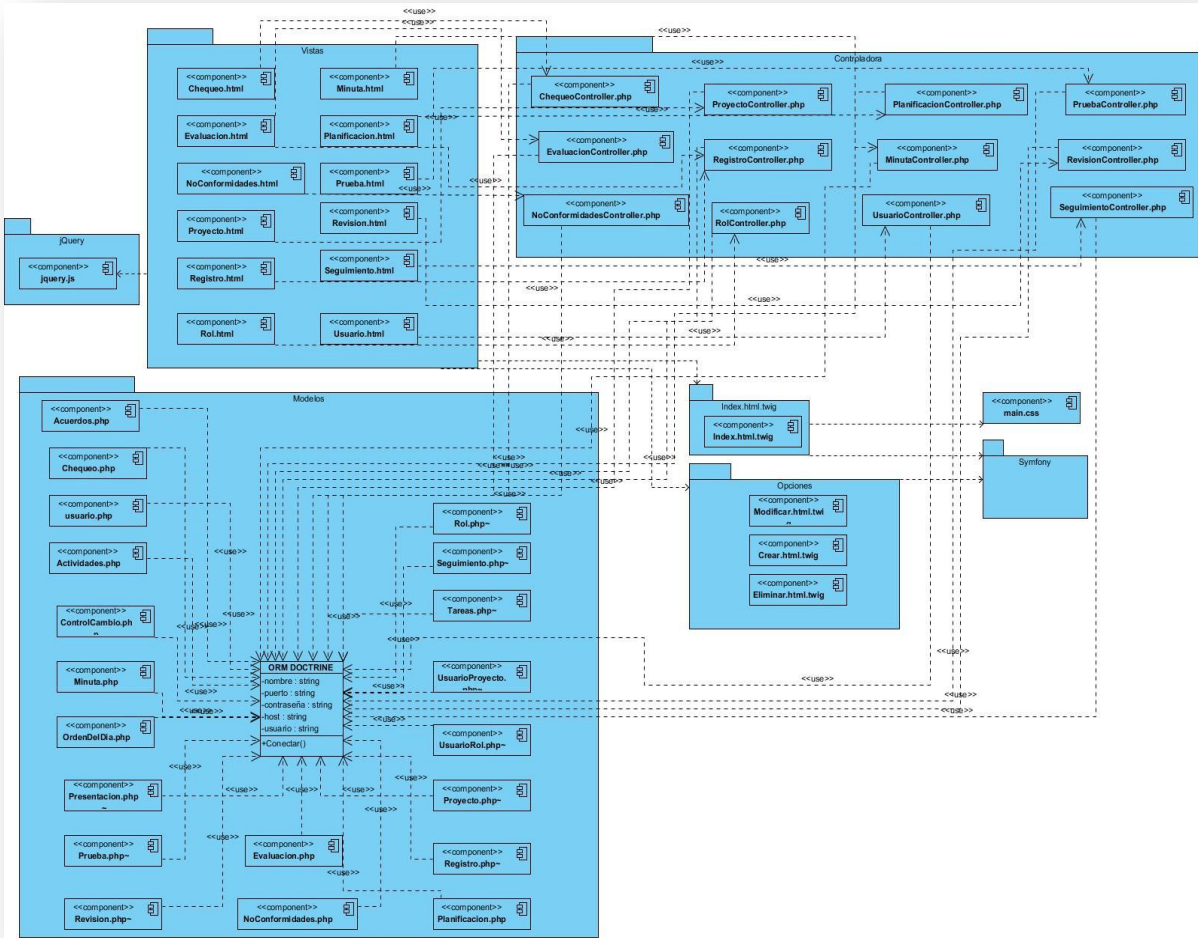


Diagrama 7: Diagrama de componente

4.3.1.1 Descripción del diagrama de componente

El diagrama de componente está representado según el patrón MVC, incluyendo los frameworks Symfony y JQuery utilizados en la implementación de la herramienta, dicho diagrama queda estructurado de la siguiente forma:

- En el paquete “Vista” se representan los componentes .HTML.
- En el paquete “Controladora” se representan los componentes .PHP que son las controladoras de las clases modelos.
- En el paquete “Modelo” se representan los componentes .PHP que son las entidades encargadas de la manipulación de los datos en el sistema.

Modelo de Despliegue

El modelo de despliegue describe la distribución física del sistema, muestra cómo están distribuidos los componentes de software entre los distintos nodos de cómputo. Permite comprender la correspondencia entre la arquitectura software y la arquitectura hardware. (Metodología de la investigación,2011) Representa típicamente un procesador o un dispositivo sobre el que se pueden desplegar los componentes. En este caso, se muestran todos los activos físicos que deben estar presentes para implantar la propuesta de solución.

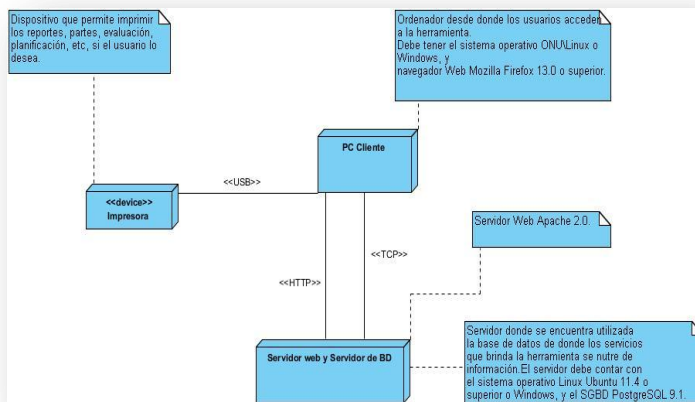


Fig. 7: Diagrama de Despliegue.

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

Nodos y enlaces de comunicación	Descripción
Impresora	Dispositivo que permite imprimir los reportes, partes, evaluación, planificación, etc., si el usuario lo desea.
PC Cliente	Ordenador desde donde los usuarios acceden a la herramienta. Debe tener el sistema operativo ONU/Linux o Windows, y navegador Web Mozilla Firefox 13.0 o superior.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

Servidor Web	Servidor Web Apache 2.0.
Servidor de BD	Servidor donde se encuentra utilizada la base de datos de donde los servicios que brinda la herramienta se nutren de información. El servidor debe contar con el sistema operativo Linux Ubuntu 11.4 o superior o Windows, y el SGBD PostgreSQL 9.1.
HTTPS	Protocolo utilizado para la conexión entre las pc cliente y el servidor web.
TCP/IP	Protocolo utilizado para la conexión entre el servidor web y el servidor donde se encuentra la base de datos.
USB	Puerto utilizado para la conexión entre la pc cliente y la impresora.

Tabla 6: Descripción de los nodos y enlaces de comunicación.

4.4 Validación del sistema

Este epígrafe tratará acerca de las pruebas de aceptación del cliente realizadas para la validación de la herramienta para la gestión de los subprocesos del SQA en el Centro GEySED. Las pruebas que se desarrollarán serán del tipo: pruebas de aceptación final del cliente, al efectuarse la entrega de la versión correspondiente. Los artefactos, condiciones necesarias y documentos rectores de esta actividad, así como todo lo relacionado a su completo y efectivo cumplimiento.

A continuación se abordan los aspectos significativos de esta prueba y la recolección y valoración de no conformidades y solicitudes de cambio de la prueba de aceptación final del cliente.

4.4.1 Prueba de aceptación final del cliente

Las pruebas de aceptación final del cliente son las encargadas de asegurar el cumplimiento de los Procesos Elementales del Negocio, los Casos de Uso, lo aceptado en los prototipos y lo planteado en el proyecto técnico.

4.4.1.2 Objetivos generales de las pruebas

- Ejecutar pruebas de *software* en un ambiente real y bajo la supervisión del equipo de desarrollo.
- Probar el rendimiento de la aplicación.
- Detectar casos críticos.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

- Determinar la factibilidad de despliegue del *software* desarrollado.

4.4.1.3 Fases para la realización de las pruebas

Todas las pruebas se realizarán de forma manual y se probarán todas las funcionalidades implementadas en el módulo según la metodología del cliente. Estas pruebas se harán teniendo en cuenta los siguientes pasos: (Rivera, 2010)

- Organización de los escenarios de pruebas y capacitación del equipo de pruebas.
- Realización de pruebas de funcionalidad.
- Realización de las pruebas de interfaz de usuario.
- Se realizarán Pruebas de Seguridad y Control de Acceso con el objetivo de validar la protección de la aplicación sensible a entradas no deseadas.
- Aplicación de la lista de chequeo y los principios de calidad.
- Conciliación de los documentos entregables resultados de las pruebas, aprobación y firma de los mismos.

A continuación se refleja el flujo de trabajo de la prueba de aceptación final del cliente:

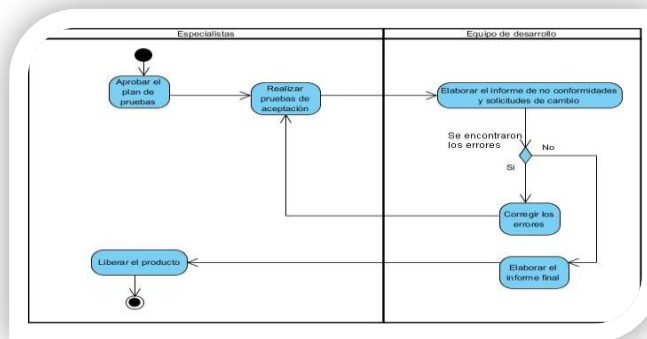


Fig. 8 Descripción del flujo de trabajo

4.4.1.4 Descripción del flujo de trabajo

Las pruebas de aceptación final del cliente se inician con la aprobación y firma del Plan de Pruebas que incluye el plan de trabajo detallado, la relación de los datos del personal que intervendrán en las pruebas y la selección de las funcionalidades a probar. De manera opcional se puede asumir el paso donde se introduce a los especialistas funcionales al uso de las funcionalidades a probar.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

Garantizadas todas las condiciones técnicas y verificadas por la dirección representante de la parte especialista, se puede dar inicio al desarrollo de las pruebas de las funcionalidades planificadas.

Para ello, intervienen el asesor de calidad de forma directa con el *software* y como apoyo, el equipo de desarrollo, esclareciendo las dudas de las funcionalidades que lo requieran. Durante las pruebas se verificará el cumplimiento de los Procesos Elementales del Negocio, los Requisitos Funcionales y los Casos de Uso.

Las no conformidades que surjan durante las pruebas serán analizadas y valorada su solución, que le será asignada al equipo de desarrollo para ser ejecutada. La respuesta se realiza de manera inmediata, brindando una nueva solución al cliente. Las solicitudes de cambios que aparezcan serán analizadas y llevadas a la mesa de negociaciones entre ambas partes; las aceptadas y pactadas pasarán a ser nuevos requisitos.

Al concluir cada ciclo se reiniciará con una nueva revisión de la aplicación, partiendo de las No Conformidades pendientes a solución y de las solicitudes aceptadas, verificándose la solución dada y la integridad del sistema. Terminada la revisión se elabora un informe final sobre el resultado de las pruebas y se libera el sistema.

4.4.2 Resultados de las pruebas de aceptación final del cliente

Los resultados de las pruebas de aceptación realizadas a la herramienta para la gestión de los subprocesos de pruebas y revisiones del SQA una vez concluida la implementación de su versión 2.0 fueron las siguientes: las inquietudes encontradas fueron de tipo No Conformidad (NC) ya que se originaron por una incorrecta implementación de un requisito previamente pactado. No se encontraron inquietudes de tipo Solicitud de Cambio. En general se encontraron 3 No Conformidades dentro de la herramienta que consistían en: faltas de ortografía en algunas interfaces, la información aparecía un poco desorganizada y el sistema en algunas ocasiones no previsualiza la alerta de la planificación correctamente; cada una de estas no conformidades fueron correctamente resueltas por el equipo de desarrollo encargado.

4.4.3 Prueba de la herramienta propuesta

El principal objetivo que tiene la ingeniería de *software* es desarrollar un producto informático con calidad. Para dar cumplimiento a ello se le realiza al sistema un conjunto de pruebas llamadas pruebas de *software*. Estas se hacen con el objetivo de encontrar errores en el producto y de esta manera conocer si el mismo tiene calidad o no en correspondencia a los requisitos funcionales identificados.

Las mismas pueden ser realizadas a lo largo del proceso de desarrollo del *software*, aunque por lo general se efectúan luego de haberse finalizada la implementación del sistema (Mantenimiento Avanzado de Sistemas de Información, 2002).

4.4.3.1 Prueba de Caja Negra

Prueba de Caja Negra (Black-Box Testing) son pruebas funcionales. Las pruebas se aplican sobre el sistema empleando un determinado conjunto de datos de entrada y observando las salidas que se producen para determinar si la función se está desempeñando correctamente por el sistema bajo prueba. Las herramientas básicas son observar la funcionalidad y contrastar con la especificación (Pressman, 2002).

Existen diferentes técnicas de prueba de Caja negra descritas por Pressman para validar la funcionalidad del sistema sin entrar a analizar su ejecución interna.

1. Métodos de prueba basados en grafos.
2. Análisis de valores límites.
3. Tabla ortogonal.
4. Partición equivalente.

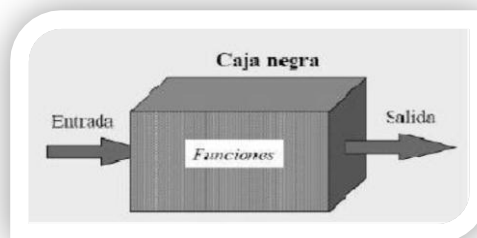


Fig. 9: Representación de técnica de prueba de Caja negra (Pressman, 2002).

Con el objetivo de garantizar la calidad del producto, demostrar que las funcionalidades del *software* son operativas, que la entrada de acepta adecuadamente y se produce una salida correcta, al sistema propuesto se le realizaron **pruebas de caja negra** y dentro de esta prueba se realizó la técnica de:

- **Partición equivalente:** Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el *software*.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada:

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

- Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.
- Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (Pressman, 2002).
- Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Los resultados de la prueba de caja negra se ven reflejados una vez llenado los diseños de casos de pruebas, los mismos serán mostrados a continuación:

4.4 Diseño de pruebas

Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Este diseño se elabora previamente a realizar las pruebas exploratorias, o de interfaz como también pueden llamarse, se parte de la descripción de los casos de usos del sistema, como apoyo para las revisiones. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión, además quedan plasmadas las revisiones realizadas al caso de prueba; así como un registro de todo aquello que no satisface y corresponde a la calidad del *software* (Pressman, 2000).

A continuación se muestra un **caso de prueba** para el CU “**Gestionar Proyecto**”

Descripción general				
El caso de uso inicia cuando el usuario selecciona adicionar, modificar o eliminar un proyecto, para ello introduce los datos necesarios, el caso de uso termina cuando se adiciona, modifique o elimine un proyecto.				
Condiciones de ejecución				
El usuario está autenticado y tiene permisos para gestionar los proyectos.				
SC 1 <Crear un Proyecto>				
Escenario	Descripción	Nombre	Respuesta del sistema	Flujo central

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

EC 1.1 Crear proyecto correctamente	Se selecciona la opción "Crear un Proyecto" donde el usuario introduce el nombre del proyecto y luego presiona el botón "Crear".	V	Se crea el proyecto satisfactoriamente, se muestra el listado actualizado de los proyectos.	Proyecto / Lista de proyectos / Crear un proyecto
		PRIMICIA		
EC 1.2 Regresar	El usuario presiona la opción Regresar	NA	Muestra la lista de proyectos existentes en el sistema.	Proyecto / Lista de proyectos / Crear un proyecto/ Regresar
EC 1.3 Campos vacíos	El usuario desea crear un proyecto dejando campos vacíos.	I	Muestra un mensaje indicando que debe llenar el campo.	Proyecto / Lista de proyectos / Crear un proyecto
SC 2 <Editar proyecto>				
Escenario	Descripción	Nombre	Respuesta del sistema	Flujo central
EC 2.1 Editar proyecto correctamente	Se selecciona la opción "Editar" , según el proyecto a modificar, donde el usuario modifica el nombre del	V	Se modifica el proyecto satisfactoriamente, se muestra el listado actualizado de los proyectos.	Proyecto / Lista de proyectos / Editar
		Video Vigilancia		

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

	proyecto y luego presiona el botón "Editar"			
EC 2.2 Regresar	El usuario presiona la opción Regresar	NA	Muestra la lista de proyectos existentes en el sistema.	Proyecto / Lista de proyectos / Editar / Regresar
EC 2.3 Campos vacíos	El usuario desea modificar un proyecto dejando campos vacíos.	I	Muestra un mensaje indicando que debe llenar el campo.	<i>Proyecto / Lista de proyectos / Editar</i>
SC 3 <Eliminar proyecto>				
Escenario	Descripción	Respuesta del sistema	Flujo central	
C 3.1 Eliminar proyecto correctamente	Se selecciona la opción "Eliminar", según el proyecto que se vaya a eliminar.	Se elimina el proyecto satisfactoriamente, se muestra el listado actualizado de los proyectos.	Proyecto / Lista de proyectos / Eliminar	

Descripción de las variables				
No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	Este campo no puede estar vacío

4.4.1 Resultado del modelo de prueba de caja negra

A partir de la constatación del modelo de prueba de caja negra realizada en la versión 2.0 se pudo determinar que: se realizaron 4 iteraciones, los resultados de la misma fueron las siguientes:

1. En la primera iteración se encontraron 16 NC, las más significativas fueron las siguientes:
 - Errores en algunas validaciones de fechas.
 - Errores en la implementación de algunos vínculos entre las interfaces.
2. En la segunda iteración se encontraron 10 NC, la más significativa fue la siguiente:
 - Errores en la implementación de algunos vínculos entre las interfaces.
3. En la tercera iteración se encontraron 4 NC, las más significativas fueron las siguientes:
 - Errores en algunas validaciones de los campos.
 - Errores en algunas validaciones de fechas.
4. En la cuarta iteración los resultados fueron satisfactorios.

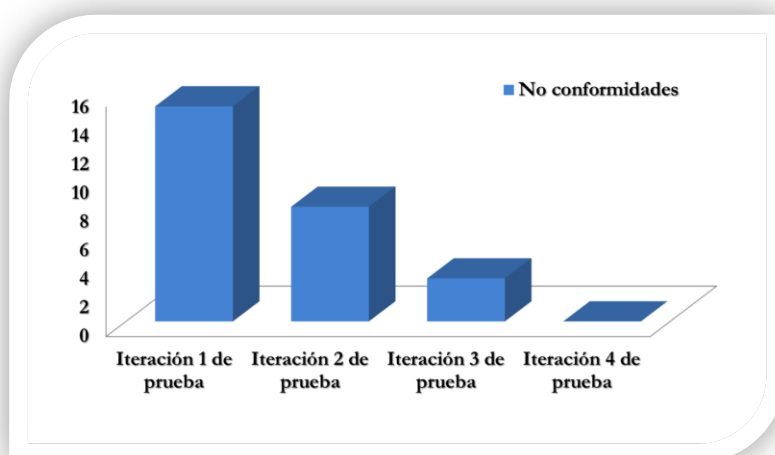


Figura 10: Resultados del método de prueba de caja negra.

4.5 Conclusiones parciales

Una vez analizadas los diferentes tipos de pruebas, se determinó utilizar el método de prueba de caja negra específicamente la técnica de partición equivalente, resolviéndose las NC detectadas lo cual permitió verificar el correcto funcionamiento de todos los requisitos funcionales. La prueba de aceptación final del cliente contribuyó a la validación de la herramienta verificando que los objetivos pactados con el cliente fueran resueltos satisfactoriamente.

CONCLUSIONES GENERALES

Una vez culminada la investigación es posible afirmar que se cumplieron todos los objetivos trazados para la misma:

- La integración de Symfony2, JQuery y PostgreSQL permitió la implementación de una herramienta para la informatización de los subproceso de pruebas y revisiones en el Centro GEySED.
- La realización de las pruebas de aceptación final del cliente y la prueba de caja negra permitieron verificar el nivel de cumplimiento de los requisitos funcionales del sistema y la aprobación final por parte del cliente con respecto a la herramienta desarrollada.
- Se contribuyó a la agilización de los subprocesos de pruebas y revisiones del Centro GEySED a partir de la implementación de la herramienta para la informatización de dichos subprocesos.
- Se obtuvo un producto con calidad de acuerdo al cumplimiento de los procesos elementales del negocio, los casos de uso, lo aceptado en los prototipos y lo planteado en el proyecto técnico y a la aprobación de los valores válidos e inválidos de las entradas existentes en la herramienta.

RECOMENDACIONES

Recomendaciones

- Se recomienda para próximas versiones de la herramienta, la informatización del subproceso de planificación para garantizar la integración de todos los subprocesos que conforman el aseguramiento de la calidad del software.
- Implementar el subproceso de auditoría que desarrolla el Centro de Calisoft correspondiente a las Políticas de Calidad, con el objetivo de informatizar los subprocesos que conforman el SQA.
- Implementar la funcionalidad de archivar un proyecto creado una vez terminados los subprocesos de prueba y revisión que conforman el SQA en el Centro GEySED.

REFERENCIAS BIBLIOGRÁFICAS

1. **DRAE**. Diccionario de la Real Academia de la Lengua Española. 22. a edición. España: s.n., 2011. <http://buscon.rae.es/diccionario/drae.htm>.
2. **Minguet, Melián J.M.** La Calidad del software y su medida. Madrid, España.: s.n., 2008.
3. **Pressman**. Calidad de Software. 2002.
4. **Informática D.C.** Control de Calidad en los Sistemas. 2000.
5. **Pressman, R. S.** Ingeniería de Software Un enfoque Práctico. 1998.
6. **Lovelle, J. R. C.** Aseguramiento de la calidad. 1999.
7. **Mendoza, L. E.** Sistemas de Información III. 2002.
8. **Villena, A. M. and MARTÍN, A. J. M.** Modelos de calidad de software. 2000.
9. **Hambling, B.F.** Plan de gestión para el desarrollo de una herramienta de administración de pruebas para el área de aseguramiento de la calidad. [Online] 2011. <http://www.uci.ac.cr/Biblioteca/Tesis/PFGMAP814.pdf>.
10. **Bañares, J. P.** Introducción a ISO/IEC 15504. 2000.
11. **Gómez, L. and HOYOS, P.** GreenSQA. 2006.
12. **Melián, J.M.** Gestión Documental Lógica. [Online] 2006. <http://gestiondocumentallogica.blogspot.com/p/plataforma-documental-logicaldoc.html>.
13. **Alejandro De Coss. 2006.** Métricas-de-Procesos-y-Proyecto. Métricas-de-Procesos-y-Proyecto. [En línea] 2006. <http://www.gdl-mexcomp.com/Documents/metricas%20de%20software.pdf>.
14. **AGAPEA.COM.** 2009. PHP. [En línea] 2009. <http://www.agapea.com/libros/Introduccion-a-PHP-5-isbn-8441518033-i.htm>.
15. **EcuRed. 2009.** El Lenguaje Unificado de Modelado. [En línea] 2009.
16. **Introduction to OMG's. 2010.** Unified Modeling Language. . [En línea] 11 de Febrero de 2010. http://www.omg.org/gettingstarted/what_is_uml.htm.
17. **Lenguajes.NET. 2010.** PHP. [En línea] 2010. <http://www.peru.blogalaxia.com/post/hypertext>.
18. **MapTools.org. 2006.** PHP MapScrip. [En línea] 2006. <http://www.dspace.espace.edu.ec/bitstream/123456789/.../18T00358%20UDCTFIYE.pdf>.

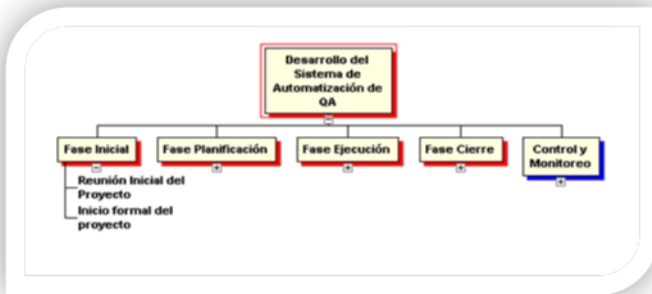
19. **Martínez, Rolando. 2010.** Características de Java. [En línea] 2010. http://www.slideshare.net/nano_trujillo/mdulo-01-introduccion-a-la-tecnologia-java.
20. **Millernar Aurias. 2006.** BuenMaster. [En línea] 2006. <http://www.buenmaster.com/?a=536>.
21. **MSDN. 2006.** MVC. [En línea] 2006. <http://msdn.microsoft.com/es-es/library/bb972251.aspx#M19>.
22. **Rodríguez, Miguel Arlandy. 2008.** Java Vs PHP. [En línea] 2008. <http://www.adictosaltrabajo.com/.../tutoriales.php?...PHPVsJava>.
23. **Rolando Castellar. 2009.** NetBeans. IDE. [En línea] 2009. <http://netbeans.org/features/ide/index.html>.
24. **Soto., José Arturo Moro. 2010.** Herramientas UML. [En línea] 2010. <http://www.jams.name/2010/04/18/herramientas-uml-cual-utilizar/>.
25. **Zarue, Augusto Morell. 2006.** Eclipse. [En línea] 2006. <http://www.blog.elartedeprogramar.cl/page/3/>.
26. **GreenSQA. 2006.** GreenSQA. [En línea] 2006. <http://www.greensqa.com/portal/>.
27. **Manager.ORG. 2012.** Free Download. Visual Paradigm para UML. [En línea] [En línea] 2007, 16 de Febrero de 2012. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5cuenta_de_Plataforma_de_Java_14715_p/.
28. **Oracle Corporation and/or its affiliates. NetBeans. 2012.** affiliates, Oracle Corporation and/or its. NetBeans. [En línea] [En línea] 2010., 10 de Febrero de 2012. http://netbeans.org/community/releases/68/index_es.html.
29. **Billy Reynoso, Carlos. 2004.** Introducción a la Arquitectura de Software. Buenos Aires: s.n. Introducción a la Arquitectura de Software. . 2004.
30. Desarrollo web ágil con Symfony2. 2011. **Eguiluz, Pérez Javier. 2011.** 2011.
31. **EcuRed. 2011.** [En línea] [En línea] [, 10 de Junio de 2011. http://www.ecured.cu/index.php/Flujo_de_Trabajo_Modelo_del_Negocio#Actores_del_Negocio.
32. **Marley, Jimi. 2011.** ¿Por qué elegir PHP?. [En línea] [En línea] 2010, 22 de Noviembre de 2011.] http://www.programacion.com/articulo/por_que_elegir_php_143.
33. **Pérez, Javie. 2012.** ¿Qué es JavaScript? . [En línea] [En línea] 2009., 10 de Diciembre de 2012. <http://www.librosweb.es/javascript/index.html>.

34. **Trum, Rafael 2012.** Patrones de arquitectura. [En línea] En línea] 2007, 24 de Marzo de 2012. http://www.proactiva-calidad.com/java/patrones/index.html#algunos_patrones..
35. **Schmuller, Joseph. 2012..** UML en 24 Horas. . [En línea] 2 de Febrero de 2012. <http://www.scribd.com/doc/38392190/UML-en-24-Horas>.
36. **Software, Ingeniería de. Curso 2009-2010..** Ingeniería de Software 1. Conferencia 6. Fase de Inicio. Disciplina Requisitos. Curso 2009-2010.
37. **Mañas, José A. 2002.** Pruebas de software. [En línea] 16 de Marzo de 2002. <http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s21>.
38. **Pressman. 2000.** Ingeniería de Software. Un enfoque práctico. 2000.
39. **Software, Ingeniería de. 2009-2010.** Conferencia 6. Fase de Inicio. Disciplina Requisitos. 2009-2010.
40. **Booch, Grady, Rumbaugh, James y Jacobson. 1999.** El Lenguaje Unificado de Modelado. Madrid : Ivar, 1999.
41. **Eguiluz, Pérez Javier. 2011.** Desarrollo web ágil con Symfony2. 2011.
42. **Potencier, Fabien. 2009.** El tutorial Jobeet. s.l.: libros web-es, 2009. 291839016X.
43. **Becerra, Yesleny., 2013.** Entrevista realizada al Ing. Yesleny Becerra Torreira. Asesora de Calidad del Centro GEySED. Universidad de las Ciencias Informáticas, Centro GEYSED.
44. **Rosalba.** [En línea] 2010. [Citado el: 20 de Octubre de 2011.] <http://rosalbarouse1.blogspot.com/>.

ANEXOS

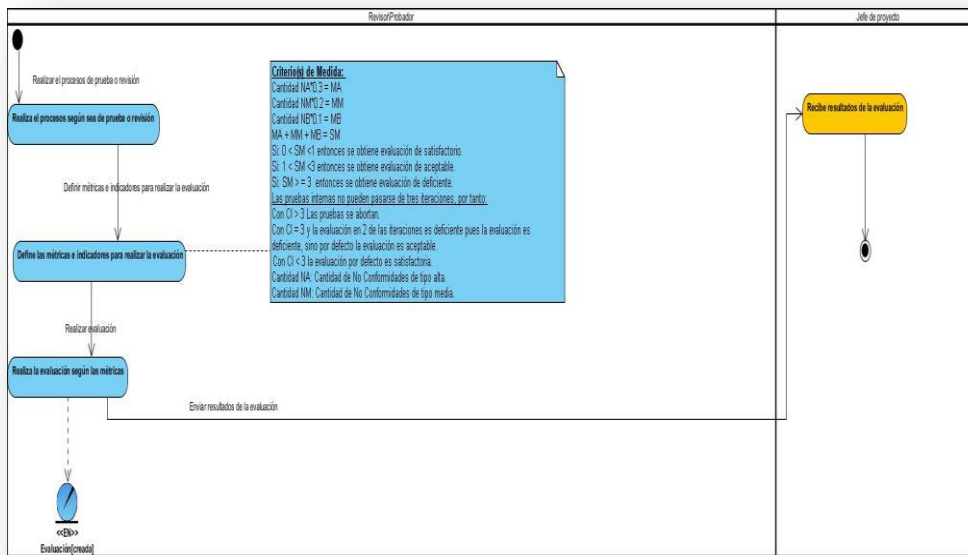
Anexo 1: EDT DEL PROYECTO DESARROLLO DE UNA HERRAMIENTA DE SOFTWARE PARA LA OFICINA DE QA.

Detalle de la Fase Inicial del EDT



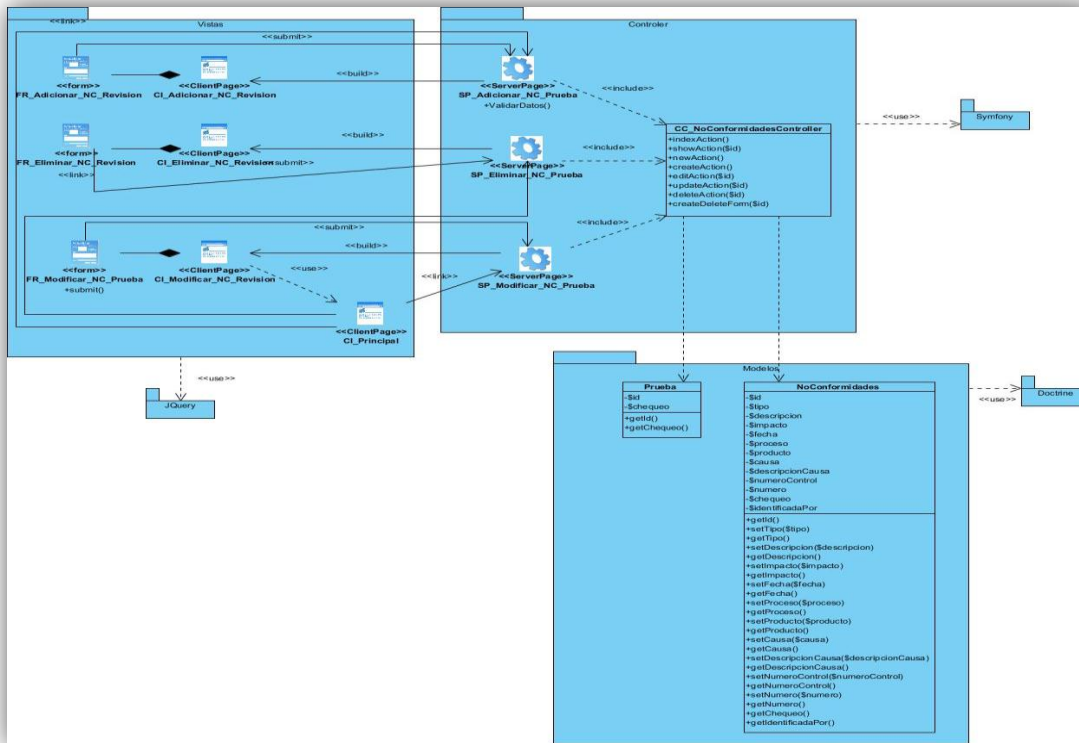
Anexo 2: DIAGRAMAS DE ACTIVIDADES DE CU DEL NEGICIO

1. CUN “Evaluar proyecto”



Anexo 3: DIAGRAMAS DE CLASE DEL DISEÑO

1. CU “Gestionar NC de prueba”



Anexo 4: Descripciones de los CU del negocio

1. CUN Elaborar Planificación

Caso de Uso del Negocio	Elaborar Planificación	
Actores	Jefe de departamento	
Resumen	El caso de inicio comienza cuando el actor indica cuándo debe realizarse la planificación y termina cuando se lleva a cabo dicho proceso.	
Casos de Uso asociados		
Acción del actor	Respuesta del proceso de negocio	
1. Indica al asesor de calidad cuándo es que se va a realizar la planificación		
	2. Se realiza la planificación según el proceso y el proyecto.	
	3. Cada jefe de proyecto recibe la documentación y el cronograma de tarea.	

	4. En caso de que se planifique una revisión, se le envía al jefe de proyecto la minuta de reunión. En caso contrario ver Otras secciones. Termina el caso de uso.
Otras secciones	4a) En caso de que sea una prueba, el especialista procede a instalar o dar permisos en el sistema. Termina el caso de uso
Mejoras propuestas	No aplicable.