Universidad de las Ciencias Informáticas
Facultad 7

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

Título: Herramienta para automatizar el proceso de evaluación del programa de mejoras en los proyectos productivos de la UCI, basados en el modelo CMMI

Autoras: Amarilis Antonia Peña González

Yudisleidy Mesa Valencia

Tutora: Ing. Deborat Pérez Montalván

Cuidad de La Habana, junio 2010

"Año 52 de la Revolución"

INTRODUCCIÓN

En la actualidad mucho de lo que antes se realizaba de forma manual en estos momentos es automatizado, debido a esto la informática ha ganado terreno en las diferentes esferas de la industria, así como en la sociedad.

La eficiencia de las organizaciones se mide según la eficacia de sus procesos, en el mundo se está creando una necesidad de desarrollar productos de alta calidad debido a la competencia que existe entre las empresas. El software como producto no está exento de esta necesidad y por lo cual asegurar la calidad de su proceso de construcción se ha convertido en una prioridad.

Los éxitos alcanzados por muchas organizaciones han demostrado que aún los mejores profesionales necesitan de un ambiente organizado, disciplinado y estructurado. Las instituciones que no proveen de dicho ambiente, arriesgan a su personal a interminables horas de trabajo para solucionar problemas.

A nivel mundial, el crecimiento de la complejidad de los proyectos y su volumen asociados a la mala calidad del trabajo, al personal inadecuado, a que las cantidades de recursos a consumir en tiempo son superiores a los estimados, a que los cambios no son controlados adecuadamente y que los proyectos se concluyan en fases posteriores a lo planificado, inciden de manera directa en la obtención de un software con calidad y en la satisfacción de las expectativas de los clientes.

Existen diferentes normas y modelos para construir software con precisión y de alta calidad, estas definen buenas prácticas de gestión e ingeniería de software, las que ayudan a organizar, dirigir y controlar el trabajo.

CMMI es uno de los más usados y completos del mundo actual que define diferentes procesos y actividades por los ingenieros de software para desarrollar productos con un elevado nivel de calidad. Posibilita la normalización y control del proceso productivo, donde se obtienen cronogramas con planificaciones más reales, calendarios completamente predecibles, el fomento del trabajo disciplinado, distribuido y colaborativo al mismo tiempo, la detección de riesgos desde etapas tempranas y la correcta mitigación de estos. Las buenas prácticas que propone contribuyen a la disminución del tiempo de desarrollo y recursos invertidos en arreglos de defectos y re-trabajo, mayor tolerancia al cambio e incremento de la capacidad de adopción y adaptación de nuevas tecnologías.

Un programa de mejora basado en CMMI facilita una alineación de los requerimientos y los principios del modelo permitiéndole a la organización la obtención de sus metas y objetivos de negocio. Ofrece una mayor confianza a los clientes y consumidores sobre los productos y servicios ofrecidos por la

organización, entrando en el mercado competitivo del software. El reto de introducir el software cubano en este mercado, requiere mucha dedicación y un alto grado de calidad en el trabajo realizado; para alcanzar calidad en el producto de software y con ello la satisfacción del cliente es importante tener un proceso de desarrollo con calidad que incluya un estricto control.

En el país se trabaja arduamente para lograr un desarrollo vertiginoso en la industria informática, que tan vinculada está al desarrollo de la economía. La Universidad de las Ciencias Informáticas (UCI) como pilar fundamental dentro de esta industria, tiene un papel promotor, por lo que ha realizado diferentes estudios de cómo asegurar la calidad de los procesos que conforman el software.

La UCI no está ajena a los problemas presentes en la industria de software y en aras de solucionar las dificultades anteriormente mencionadas se lleva a cabo un programa de mejoras de software, la misma utiliza como referencia CMMI con el objetivo de aumentar la madurez y capacidad de la organización.

El programa de mejoras que se desarrolla actualmente en la universidad tiene como objetivo la definición e implementación de los procesos necesarios para cubrir las 7 áreas de procesos del nivel 2 en la representación escalonada del modelo, buscando con ello la reducción de varios de los problemas que están vigente en la producción de software en la universidad.

Una vez definidos los procesos disciplinados para el desarrollo de software, se generaliza un marco de trabajo homogéneo en cada uno de los proyectos, que permite organizar y priorizar las actividades del programa de mejoras de procesos. Posteriormente se procede a realizar el pilotaje de estos procesos, bajo un ambiente controlado, para conocer con más detalles cómo funcionan antes de llevar a cabo la institucionalización de los mismos. En la UCI el método usado actualmente para darle seguimiento a los proyectos pilotos que se encuentra dentro del alcance del programa de mejoras, es la realización periódica de revisiones, donde se evalúa el avance del proyecto en la implementación de los procesos definidos teniendo en cuenta el modelo CMMI. La evaluación se realiza con el objetivo de conocer hasta que punto cumple con las prácticas que plantea el modelo. Es muy importante que al final de la revisión, el especialista realice un reporte donde se plasmen los hallazgos encontrados. Esta actividad puede requerir mucho tiempo y esfuerzo de parte del especialista quien debe estudiar detalladamente el resultado de la evaluación para determinar las fortalezas y oportunidades de mejoras encontradas del proyecto.

Con el fin de probar un programa de mejoras y poder controlar los resultados de manera eficiente se definió que solo estuvieran dentro del alcance 3 centros de los que existen actualmente en la UCI, pero con el propósito de asegurar la calidad de todos sus productos y ganar en competitividad la UCI está

interesada en que todos sus centros alcancen el nivel 2 de CMMI, pero está consciente de que no todos lo pueden lograr a la vez, siguiendo exactamente los mismos procesos, todo puede depender de si implementan o no buenas prácticas para el desarrollo de software.
La UCI desea incluir posteriormente en un segundo programa de mejoras a los centros restantes pero, no cuenta con una infraestructura que le permita conocer las características especificas de cada proyecto ni cómo funciona detalladamente cada uno de ellos. Hasta el momento solo puede identificar algunas particularidades de los proyectos a través del diagnóstico que realiza el Centro Calisoft, teniendo en cuenta algunos indicadores muy generales.
Para determinar qué proyectos pueden entrar en el programa de mejoras 2 es necesario conocer las fortalezas y oportunidades de mejora respecto al modelo CMMI de cada uno de los posibles proyectos a incluir, de forma tal que se escojan aquellos que reflejen un mayor grado de acercamiento a las buenas prácticas que define el modelo. Realizarlo de esta forma implicaría un proceso menos traumático para el proyecto ya que no significaría tener que cambiar drásticamente su proceso de desarrollo de software solo incorporar algunas buenas prácticas. Utilizando los mismos procesos con un mínimo de variaciones. El desarrollo de una herramienta, que facilite y agilice la realización de este proceso de evaluación, pues se podría volver insostenible hacerlo de forma manual cuando aumente el número de proyectos a evaluar y existiendo una herramienta automatizada que ayude a evaluar el grado de acercamiento de los proyectos que integran los centros restantes, a lo definido en el actual programa de mejoras, la tarea se volvería menos engorrosa.

Teniendo en cuenta la situación anterior, el **problema a resolver** radica en: ¿Cómo facilitar el proceso de evaluación del programa de mejoras implementado en los proyectos productivos de la UCI basados en CMMI?

Este problema tiene como **Objeto de estudio**: El programa de mejoras implementado en los proyectos productivos de la UCI, basado en el modelo CMMI y como **Campo de acción**: Proceso de evaluación del programa de mejoras implementado en los proyectos productivos de la UCI, basado en el modelo CMMI.

Para lo cual se plantea como **Objetivo General**: Desarrollar una herramienta que permita automatizar el proceso de evaluación del programa de mejoras implementado en los proyectos productivos de la UCI, basado en el modelo CMMI.

Para darle cumplimiento a este objetivo propuesto se plantean como Tareas de investigación:

- Profundizar en el estudio del modelo CMMI.
- Profundizar en el estudio del programa de mejoras que se implementa en la UCI.
- Realizar entrevistas al personal capacitado.
- Identificar parámetros para la evaluación de proyectos basada en el modelo CMMI
- Definir un método matemático para evaluar el grado de cumplimiento de los proyectos con el modelo CMMI.
- Identificar tecnologías y herramientas para el desarrollo del sistema.
- Realizar el diseño del sistema.
- Implementar el sistema.
- Realizar pruebas al sistema.

El presente trabajo de diploma está estructurado de la siguiente forma:

- **Capitulo 1. Fundamentación teórica:** Este capítulo recoge toda la fundamentación teórica que sustenta el desarrollo de la investigación para el posterior desarrollo del sistema propuesto. En el mismo se hace un estudio del estado del arte de sistemas existentes con características similares en cuanto a objetivos de uso y de funcionamiento. Se hace además referencia a las tendencias actuales así como a las tecnologías y las metodologías que se aplican durante el del proceso de investigación y de desarrollo del sistema.
- **Capitulo 2. Características del Sistema**: Se analizan los flujos de trabajo de Modelado del Negocio, Especificación de los Requerimientos de software, Análisis y Diseño del Sistema. Se muestran los principales procesos a través de casos de uso, los actores que intervienen, sus relaciones y una descripción resumida de cada uno de ellos, además se muestran los requerimientos funcionales y no funcionales del sistema.

- **Capitulo 3. Análisis y diseño del sistema:** Se obtienen el Modelo de Análisis y el Modelo de Diseño. Así como los diagramas de clases del análisis y del diseño, diagramas de interacción, descripción de las clases de diseño y el diseño de la base de datos.
- **Capítulo 4. Implementación y pruebas:** En este capítulo se obtiene el Modelo de Implementación, el diagrama de despliegue, la arquitectura, recoge los diagramas de componentes, así como una descripción general de los mismos. Muestra las pruebas a realizarle a la aplicación así como el diseño de los casos de prueba. En este capítulo se pretende mostrar los resultados obtenidos luego de aplicar la herramienta a cada proyecto. Presentar un conjunto de características e imágenes, así como una breve descripción de su funcionamiento que demuestren como facilitó la herramienta implementada al ahorro de esfuerzo y tiempo en la evaluación de los proyectos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se abordan los conceptos fundamentales asociados a la calidad del software y al programa de mejoras. Además, se hace un análisis de algunas herramientas existentes (o más importantes) utilizadas para evaluar el nivel de madurez y capacidad de la organización.
También se muestra un estudio de las tendencias, tecnologías, y metodologías más usadas en la actualidad, así como las herramientas a utilizar para el desarrollo de la aplicación, especificando aquellas que serán utilizadas en la modelación e implementación de la posible solución del problema a resolver.

1.1. Conceptos fundamentales asociados al dominio del problema.

1.2.1. Calidad de Software

“Grado con el que un sistema, componente o proceso cumple con los requerimientos especificados y las necesidades o expectativas del cliente o usuario”.

“Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario”.

1.2.2. CMMI

Es un modelo de referencia para el crecimiento de capacidades y madurez, que se enfoca tanto en procesos de Administración como de Ingeniería de Sistemas y Software. Fue creado por el SEI (Software Engineer Institute) de la universidad Carnegie Mellón. Está compuesto por 22 Áreas de proceso (AP). El método usado por los creadores del modelo para certificar el ascenso de una organización por los diferentes niveles es el conocido como SCAMPI.

1.2.3. Áreas de proceso del Nivel 2 de CMMI

Agrupan un conjunto de prácticas relacionadas entre sí cuya ejecución permite alcanzar una serie de objetivos o metas. Establecen la capacidad de la organización o Madurez Organizacional. Cada área pertenece a un nivel de madurez predeterminado (CMMI por Etapas) o a una categoría predeterminada (CMMI continuo). El nivel 2 está compuesto por 7 áreas de procesos tales como:

- Gestión de Requerimientos (REQM)
- Planificación de proyectos (PP)
- Monitorización y Control de proyectos (PMC)
- Medición y Análisis(MA)
- Aseguramiento de la calidad (PPQA)
- Gestión de la configuración (CM)
- Administración de Acuerdos con proveedores (SAM)

1.2.3.1. Administración de Requerimientos (REQM)

El propósito de la gestión de requerimientos es gestionar los requerimientos de los elementos del proyecto y sus componentes e identificar inconsistencias entre estos requerimientos, el plan de proyectos y los elementos de trabajo. Tiene como objetico administrar los requerimientos.

1.2.3.2. Planeación de proyectos (PP)

El propósito de la planificación de proyectos es establecer y mantener planes que definen las actividades del proyecto. Entre sus objetivos de encuentra establecer las estimaciones, desarrollar un plan de proyecto y obtener un compromiso.

1.2.3.3. Monitoreo y Control de proyectos (PMC)

El propósito del monitoreo y control de proyectos es proporcionar compresión del estado del proyecto para que se puedan tomar acciones correctivas cuando la ejecución de proyecto se desvíe del plan. Tiene como objetivo que los resultados actuales y el progreso del proyecto se supervisan con respecto al plan de proyecto, así como administrar las acciones correctivas hasta su cierre.

1.2.3.4. Medición y Análisis (MA)

El propósito de la medición y el análisis es desarrollar y sostener una capacidad de medición que sea usada para ayudar a las necesidades de información de la administración. Tiene como objetivo alinear las mediciones y las actividades del análisis, así como proporcionar los resultados de las mediciones.

1.2.3.5. Aseguramiento de la calidad del proceso y el producto (PPQA)

El propósito del aseguramiento de la calidad es proporcionar a los equipos de trabajo y a la dirección, visibilidad objetiva de los procesos y sus productos asociados. Tiene como objetivos evaluar objetivamente la ejecución de los procesos y productos, así como proporcionar viabilidad objetiva.

1.2.3.6. Administración de la configuración (CM)

El propósito de la gestión de la configuración es establecer y mantener la integridad de los productos utilizando la identificación, el control, la contabilidad de estado y las auditorias a la configuración. Tiene como objetivo establecer la línea base, supervisar y controlar los cambios y establecer la integridad.

1.2.3.7. Administración de Acuerdos con proveedores (SAM)

El propósito de la Administración de Acuerdos con proveedores es administrar la adquisición de productos con proveedores con los que existe un acuerdo formal. Tiene como objetivo establecer y mantener acuerdos con proveedores y satisfacer los acuerdos con este.

1.2.4. Programa de mejora continua en la UCI

Un programa de mejora continua es una secuencia de instrucciones para llevar a cabo los adelantos continuos en los proyectos productivos. Con el mismo se consigue lograr una mayor calidad y eficiencia en dichos proyectos.

Para mejorar los procesos de desarrollo de software de la UCI se lleva a cabo el proyecto de mejoras bajo el modelo CMMI nivel 2 presentados por el SIE CENTER, que incluye:

- ✓ Evaluación inicial de los procesos actuales de producción de software de la UCI.
- ✓ Consultoría, implementación e institucionalización de los procesos de Nivel 2 de CMMI.
- ✓ Evaluación formal basada en el SCAMPI (Standard CMMI Appraisal Method for Process Improvement).

1.2.5. SCAMPI.

El Standard CMMI Appraisal Method for Process Improvement (SCAMPI) es el método oficial SEI para proveer puntos de referencia de sistemas de calificación en relación con los modelos CMMI. SCAMPI se usa para identificar fortalezas y debilidades de los procesos, revelar riesgos de desarrollo/adquisición, y determinar niveles de capacidad y madurez. Se utilizan ya sea como parte de un proceso o programa de mejoramiento, o para la calificación de posibles proveedores. El método define el proceso de evaluación constando de preparación; las actividades sobre el terreno; observaciones preliminares, conclusiones y valoraciones; presentación de informes y actividades de seguimiento.

1.2. Herramientas estudiadas

El modelo CMMI constituye un marco de referencia de la capacidad de las organizaciones de desarrollo de software en el desempeño de sus diferentes procesos, proporciona una base para la evaluación de la madurez y una guía para implementar una estrategia para la mejora continua de los mismos.

Según se fue desarrollando CMMI se crearon herramientas de evaluación del software con el fin de hacer menos tedioso y más ágil el trabajo al aplicar este modelo, algunas de ellas no están enfocadas a todas las áreas de procesos. Por lo que se decidió realizar un estudio detallado sobre el desarrollo de herramientas que faciliten una amplia valoración y evaluación de los procesos de software encaminados al proceso de mejoras para el nivel 2 de CMMI. Realizando un análisis de las características, ventajas y desventajas que nos ofrece cada una de estas.

1.3.1. Appraisal Wizard

Soporta evaluaciones para gran parte de los modelos CMM y métodos de evaluación propuestos por el SEI a lo largo de la historia (entre ellos, todos los CMMI y SCAMPI). Está pensada para cubrir todas las necesidades del método SCAMPI, requiriendo amplios conocimientos del mismo por parte del usuario. Requiere que el usuario ingrese todos los valores que se asignan en las distintas instancias de evaluación

(prácticas, objetivos, áreas de proceso) y no cuenta con la capacidad de sugerir valores facilitando las tareas de ingreso de datos. Al brindar un soporte tan amplio, detallado y una interfaz de usuario difícil y poco amigable, la herramienta en cuestión no es para nada sencilla de utilizar, ya que está diseñada principalmente para usuarios expertos.

1.3.2. CMM Quest

El funcionamiento de esta consiste en asignar valores a los objetivos, tiene como inconveniente que no permite valoraciones a nivel de prácticas generales y específicas y no brinda soporte para el método SCAMPI .Su utilización presenta un grado medio de dificultad. Brinda una interfaz de usuario fácil, muy amigable y está realizada para usuarios novatos. Soporta modelos de CMMI-SE/CW (Representación continua).Utiliza como método de evaluación ISO/IEC 15504. Se han realizados varias versiones de esta herramienta, aportando cada una de ellas nuevas características.

CMM Quest v1.2 Personal Edition

Contiene todas las funciones de CMM Quest y aporta nuevas características que necesita para realizar la evaluación. Permite guardar su evaluación en un archivo y brinda la posibilidad de imprimir los gráficos y generar un informe de Word o HTML. Permite usar textos predefinidos, para las notas del evaluador. Soporta archivos de idiomas de múltiple evaluación.

CMM v1.2 Quest Corporate Edition

La Corporate Edition incluye licencias de uso múltiple de Inhouse CMM Quest v1.2. Además contiene una licencia de SYNEVAL, que ofrece una evaluación eficaz de más de dos evaluaciones, lo cual brinda una desventaja, pues no todas las empresas tienen condiciones económicas óptimas para realizar el financiamiento de estas licencias. Además, el Corporate Edition ofrece la posibilidad de adaptar la herramienta de evaluación de sus necesidades individuales con la herramienta de software addidional SynEdit. Con SynEdit puede editar y mejorar los procesos, enlace de las explicaciones a su compañía de los propios documentos y la adaptación de todos los íconos y leyendas.

Es capaz de crear todo un proceso y sistema de gestión de calidad. Es posible traducir la herramienta de evaluación en su conjunto, así como las partes del cuestionario en cualquier idioma deseado. El diseño de

la herramienta de evaluación podrá ser adaptado a las necesidades de las organizaciones. Ofrece como desventaja el pago de las licencias para el posterior uso de la herramienta.

1.3.3. IME Toolkit

Consiste en realizar evaluaciones que asignan valores numéricos a las prácticas, en base a los cuales la herramienta genera puntajes para las áreas de proceso. Su utilización presenta un grado medio/bajo de dificultad. No brinda soporte para el método SCAMPI. No posee guías de asistencia para la evaluación. Presenta una interfaz de usuario medianamente amigable y está realizada para usuarios expertos. Soporta modelos de CMMI-SE/CW.

1.3.4. Evaluación Asistida de CMMI SW

La función de esta herramienta consiste en asignar valores a nivel de práctica, objetivos, áreas de proceso o nivel de madurez. Brindar soporte a las reglas del método simplificando y facilitando las tareas del evaluador. Cuenta con una interfaz gráfica donde se muestra un árbol de navegación con los distintos componentes del modelo.

La herramienta permite obtener un reporte en el cual se incluyen todas las valoraciones efectuadas por el usuario. Posee la capacidad de almacenar y recuperar archivos, las evaluaciones en curso. Se basa en el método de evaluación SCAMPI y usa como modelo de referencia CMMI-SW. Presenta algunas desventajas tales como: no posee soporte para la evaluación del modelo CMMI-SW en su representación continua ya que actualmente solo soporta la representación por niveles, no brinda soporte para la evaluación de todos los demás modelo de la familia CMMI, la capa de presentación de la herramienta no funciona en modo web. La misma presenta como su principal desventaja el hecho de ser propietaria, lo cual representa un inconveniente, pues no todas las empresas disponen de una economía capaz de sustentar esta serie de gastos.

1.3.5. SPICE

Ofrece una base para una evaluación muy detallada del estado actual del proceso de una organización. Por su gran nivel de descomposición de los procesos e indicadores, proporciona evaluaciones objetivas y con resultados repetibles, especialmente cuando es realizada por evaluadores entrenados y cualificados. Asigna valores a las prácticas base y a las prácticas genéricas directamente. Plantea 37 preguntas acerca de su producción de software. Contiene más de 300 definiciones y explicaciones que abarcan la

gestión de la calidad del software y la ingeniería de software. Contiene un generador de informes basado en plantillas de WinWord.

La herramienta presenta un mediano grado de dificultad al ser utilizada. Se basa en el método de evaluación ISO/IEC TR 15504:1998.Presenta algunas desventajas tales como: requiere un gran esfuerzo para realizar las evaluaciones y por tanto un alto coste, no contiene una estrategia de mejora del proceso, lo cual puede verse como positivo o negativo dependiendo de lo que se quiera.

1.3.6. SPQA.web

Desarrollada y mantenida por el "Grupo de Mejora de Procesos Software" del grupo de investigación IDIS, de la Facultad de Ingeniería Electrónica y Telecomunicaciones. Universidad del Cauca.
Colombia. Es una herramienta Web gratuita que sirve de apoyo a las valoraciones rápidas de procesos software y tiene dos características fundamentales: soporta diferentes instrumentos de valoración obtenidos de los modelos de procesos de referencia, y permite gestionar los instrumentos de valoración de acuerdo con las necesidades del entorno (por ejemplo la aparición de actualizaciones o nuevos estándares de mejora, ó por las necesidades propias capturadas en las empresas a las cuales brinda soporte el "Grupo de Mejora de Procesos Software").

Actualmente la herramienta SPQA.web tiene un instrumento que soporta la valoración de algunas áreas de procesos del modelo CMMI y otro instrumento que soporta la valoración de los procesos del estándar ISO/IEC 12207:2002[4]. El método de evaluación está basado en el estándar ISO/IEC 15504:2004. Para la evaluación de los procesos se asignan directamente valores a las prácticas base (o específicas) y a las prácticas genéricas. Esta herramienta presenta algunas desventajas tales como: no asocia a las preguntas ningún tipo de información aclarativa, ya que algunas preguntas no son muy claras y no abarca todas las áreas de procesos de CMMI.

1.3.7. MEDPROY

Herramienta desarrollado en la UCI, es una aplicación capaz de evaluar (en porciento) el grado de acercamiento de los proyectos productivos al programa de mejoras establecido en la universidad y dirigido por CALISOFT .La misma permite generar reportes de los proyectos evaluados para que puedan mejorar, adicionar o eliminar variables según la propuesta de solución predeterminada hasta el momento. Facilita la evaluación de variables, posibilitando conocer el estado real del proyecto evaluado.

Es una herramienta muy sencilla, el ambiente de trabajo es muy cómodo, tiene un ambiente amigable y clásico de las aplicaciones de Windows. El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente de aplicación de escritorio en sentido general. Identifica al usuario antes de que pueda realizar cualquier acción sobre el sistema, protege la información manejada por el sistema de accesos no autorizados, garantiza que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que esté activo. Utiliza como lenguaje de programación C#.Tiene como desventaja que solo abarca dos de las 7 áreas de procesos que describe CMMI para el nivel 2, no aplica la propuesta de solución a todos los proyectos productivos de la UCI, no realiza actualizaciones a la aplicación para que soporte otros niveles de CMMI, no permite que los proyectos realicen evaluaciones internas.

1.3.8. Propuesta

Las herramientas analizadas anteriormente soportan un único modelo de procesos y método de evaluación y tienen como característica común que son comerciales. No todas las empresas tienen la posibilidad de adquirir herramientas comerciales debidas a limitantes financieras, resaltando las dos últimas como herramientas gratuitas que sirven de apoyo a las valoraciones rápidas de procesos software.

Por otra parte, la ausencia de herramientas amigables para la evaluación también complica las tareas de los evaluadores expertos, ya que en la mayoría de los casos se necesita que estos tengan amplios conocimientos y estén familiarizados con el modelo y la herramienta, a pesar de que muchas veces deben solventar de forma manual la ausencia de tales herramientas.

Ante este panorama, se planteó la necesidad de una herramienta que facilite el proceso de selección de los proyectos que por sus características ,se acercan más a lo que se ha estado definiendo en el actual programa de mejoras, teniendo en cuenta que la herramienta que se necesita se debe poder acceder desde cualquier lugar, para que mediante permisos los usuarios o proyectos pueden ejecutar diferentes acciones, por ejemplo, que puedan responder las encuestas y después ver cómo está el estado de su proyecto, ver estadísticas de los mismos, gestionar los proyectos y las preguntas para los cuestionarios. Por lo que surge la necesidad de desarrollar una aplicación Web para que pueda ser accedida desde cualquier parte sin tener en cuenta la plataforma en la que se encuentra el usuario. El principal objetivo de este trabajo de diploma es realizar una herramienta dinámica, que sea amigable y entendible para los

usuarios, que posibilite actualizaciones que permitan soportar otros niveles de CMMI, asignar un rol para que cada proyecto posibilite realizar evaluaciones internas del mismo.

1.3. Metodologías de desarrollo del software

Para garantizar la calidad de un software durante el desarrollo de este, es vital seguir un conjunto de pasos, que llevan a utilizar una metodología de desarrollo de software. Actualmente el desarrollo de software ha alcanzado un alto nivel debido a la competencia que existe, por lo que los desarrolladores se han visto en la necesidad de buscar técnicas mediante las cuales se logren estandarizar el trabajo de las aplicaciones que se desarrollan. Teniendo en cuanta lo anterior se ha creado un conjunto de metodologías para el desarrollo del software que hace posible que todo el personal de un proyecto se vincule y pueda entenderse. Las metodologías se pueden agrupar en tradicionales o formales, y ágiles. Las tradicionales centran su atención principalmente en desarrollar una documentación completa de los proyectos así como el cumplimiento de un plan de Proyecto, el cual debe ser definido al comienzo del mismo. Las de tipos ágiles proponen como de vital importancia la respuesta a los cambios y en mantener una buena relación con el cliente.

1.4.1. Programación Extrema (XP)

La metodología Extreme Programming (XP) o Programación Extrema, es una de variantes de las metodologías ágiles más destacadas y con más aceptación en la comunidad internacional de desarrollo. Los fundamentos de la misma según su creador son: mejorar la comunicación, buscar la simplicidad, buscar retroalimentación en que también va el trabajo y que siempre hay que proceder con valentía. Considera un aspecto natural, inevitable e incluso deseable, los cambios de requerimientos sobre la marcha.

Una de las herramientas más importantes en la metodología XP es el desarrollo orientado a pruebas, que utiliza las pruebas unitarias como eje de todo desarrollo. Las interacciones suelen ser muy cortas y se promueve a los programadores a buscar soluciones y experiencia con ellas, programar sin miedo a descomponer el sistema.

Se considera como una adopción de las mejores metodologías de desarrollo teniendo en cuenta lo que se pretende realizar con el proyecto, y de manera dinámica aplicarlo durante el ciclo de vida del software.

1.4.2. SCRUM

Es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza ejecutable que incorpora una nueva funcionalidad.

En SCRUM el equipo se enfoca solamente en construir software de calidad. La gestión de proyecto se focaliza en definir las características del software a construir y remover cualquier obstáculo que pudiera entorpecer el trabajo del equipo de desarrollo. Se busca que los equipos de desarrollo sean lo más productivos posible. *(*Schwaber, 2004)

Scrum es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requerimientos son cambiantes o poco definidos, donde la innovación, la competitividad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

1.4.3. Proceso Unificado de Rational (RUP)

Es una metodología formal o proceso de ingeniería de software que provee un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga la necesidad del usuario final dentro de un tiempo y presupuesto previsible.

RUP mejora la productividad del equipo ya que permite que cada miembro del grupo sin importar su responsabilidad específica acceda a la misma base de datos de conocimiento. Esto hace que todos compartan el mismo lenguaje, la misma visión y el mismo proceso acerca de cómo desarrollar software. Describe detalladamente todas las actividades, roles, responsabilidades, productos de trabajo y herramientas para definir el quién, cómo, qué y cuándo en un proyecto de desarrollo de software. Representa un ideal de referencia para todo el equipo de trabajo, además es un proceso de desarrollo de software que recoge un conjunto de actividades, las cuales son necesarias para transformar los requerimientos del cliente en un sistema de software. Está dividido por fases, donde en cada una de estas se producirá una o varias iteraciones.

RUP divide el proceso en 4 fases principales, de las cuales se realizan varias iteraciones según el proyecto y en las que se hace un mayor énfasis en las distintas actividades. (JHONATAN MINA, 2008)

RUP aplica 6 prácticas principales para el desarrollo de sistemas. Estas prácticas son:

- Desarrollo de software en forma iterativa: permite ir creciendo en el entendimiento del problema a través de refinamientos sucesivos. Esto también permite introducir cambios tácticos en los requerimientos, características del sistema o en los tiempos.
- Gestión de requerimientos: es necesario para garantizar que al final el software tenga la calidad requerida.
- Uso de arquitecturas basadas en componentes: el uso de una arquitectura basada en componentes hace que el sistema pueda reutilizar componentes desarrollados con antelación y facilitar el trabajo que se realiza.
- Modelización visual del software: El proceso le demuestra cómo modelar visualmente software para capturar la estructura y el comportamiento de arquitecturas y de componentes.
- Verificación de calidad del software: RUP le asiste en el planeamiento, el diseño, la puesta en marcha, la ejecución, y la evaluación de las pruebas de confiabilidad, funcionalidad, performance de la aplicación y el sistema.
- Control de cambios: La capacidad de manejar los cambios asegurándose que cada cambio sea aceptable, y pudiendo continuar con los mismos es esencial en un ambiente en el cual el cambio es inevitable.

1.4. Lenguaje de Modelado

Se pueden capturar las partes esenciales de un software mediante el uso de las notaciones gráficas en la modelación de una aplicación de software. El lenguaje de modelación es independiente del lenguaje de programación que se desea emplear en el desarrollo del software. Con el transcurso de los años se ha producido un auge en la Programación Orientada a Objeto (POO), lo que ha conllevado a la creación de diferentes lenguajes de modelado orientado a Objetos.

1.5.1. Lenguaje de Modelado OO i*

La notación i* fue creada en la primera mitad de la década de los 90. Permite expresar de forma clara y sencilla los objetivos de los actores que aparecen en los modelos y la dependencia entre ellos. Consta con una notación gráfica que permite tener una visión intuitiva y unificada del entorno modelado mostrando tales actores y dependencias.

Tiene la desventaja de no tener una definición única del lenguaje. Además, las definiciones existentes no son tan claras como se desearía ya que contienen ambigüedades y contradicciones.

1.5.2. Lenguaje de modelado OO UML

UML es un lenguaje que provee un conjunto de herramientas que permiten modelar (analizar y diseñar) sistemas Orientados a Objeto. (Torres, 2005).

Las herramientas que provee UML para el desarrollo de sistemas OO son:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de estados.
- Diagrama de secuencias.
- Diagrama de actividades.
- Diagrama de colaboraciones.
- Diagrama de componentes.
- Diagrama de distribución.

Características de UML:

- Permite adaptarse fácilmente a los usuarios, así como a otros usuarios de otros métodos.
- Permite modelar nuevos elementos en el proyecto que se desarrolla.
- UML provee una expresividad e integridad holística mejorada, respecto a otros lenguajes de modelamiento visual.
- UML es fácil de aprender y usar, ya sea respecto a las técnicas más avanzadas, es decir, estereotipos y propiedades, así como algunos cambios en la anotación y semánticas.
- UML será la elección obvia para realizar nuevos proyectos, especialmente cuando se incremente la disponibilidad de herramientas, libros y cursos.

1.5. Framework de desarrollo

Es una estructura de trabajo y soporte definida, mediante la cual otro software pude utilizar para llevar un trabajo más organizado y desarrollado en la realización de software. Dentro de él se pueden encontrar soportes de programas, bibliotecas y un lenguaje interpretado entre otros software para facilitar el desarrollo y unir las diferentes estructuras y componentes de un proyecto.

Es una arquitectura de software que permite relacionar las entidades del dominio del problema que se está resolviendo mediante el software. La estructura y metodología de trabajo permite realizar software más rápido y mejor organizado ya que presenta librerías y códigos que pueden ser utilizados, lo cual agiliza el trabajo de los desarrolladores.

1.6.1. Symfony

Es un Framework para aplicaciones Web desarrollado en PHP5 por Fabien Potencier y patrocinado por los laboratorios de Sensio, Agencia francesa Web muy conocida por sus puntos de vista innovador en el desarrollo Web. Acelera la creación y mantenimiento de las aplicaciones Web, y permite la reutilización de código, lo cual agiliza el desarrollo de la aplicación. Su diseño permite optimizar las aplicaciones y mantener una organización de la misma. Permite reducir el tiempo de desarrollo de una aplicación ya que proporciona varias herramientas y clases las cuales puedes ser reutilizada por los desarrolladores. Permite el desarrollo por capas, ya que se basa en el patrón arquitectónico Modelo Vista Controlador (MVC).

1.6.2. Kumbia

Es un framework para desarrollo de aplicaciones Web escrito en PHP5. Se basa en las prácticas de desarrollo Web como “Don't Repeat Yourself” (DRY) y el Principio “Keep It Simple, Stupid” (KISS). Fomenta la eficiencia y velocidad en la creación y mantenimiento de aplicaciones Web, remplazando códigos repetitivos. Intenta proporcionar un medio fácil para el desarrollo de aplicaciones robustas para entonos educativos y comerciales, gracias a que es flexible y configurable.

1.6.3. CakePHP

Es un Framework para aplicaciones Web desarrollado en PHP, creado sobre los conceptos de “Ruby on Rails”. Empezó su desarrollo en el 2005 cuando Ruby On Rails estaba ganando popularidad y desde entonces ha generando muchos proyectos y su comunidad ha crecido. Facilita el usuario la interacción con las Bases de datos mediante el uso de Active Récord, además se basa en el patrón arquitectónico Modelo Vista controlador (MVC), lo que le permite dividir la aplicación en capas. Tiene como características fundamentales:

- Compatible con PHP4 y PHP5.
- CRUD de la base de datos integrado.
- URLs amigables.
- Sistema de plantillas rápido y flexible.
- Helpers para AJAX, Java script, HTML, forms y más.
- Trabaja en cualquier subdirectorio del sitio.
- Validación integrada.
- Scaffolding de las aplicaciones.
- Access Control Lists.
- Sanitización de datos.
- Componentes de seguridad y sesión.

Comparación entre lo Framework estudiados

Característica	**Kumbia**	**Symfony**	**Cake**
MVC Nativo	**X**	**X**	**X**
Enrutamiento Avanzado	**X**	**X**	**X**
Scaffold (Generadores de código)	**X**	**X**	**X**
Scaffold Avanzado (Generadores de Formularios)	**X**		
Correo Electrónico	**X**	**X**	**X**
Mapeo Objeto-Relacional	**X**	**X**	**X**
Asociaciones ORM	**X**	**X**	**X**
Eventos ORM	**X**	**X**	**X**
Sistema de Plantillas	**X**	**X**	**X**
Integración con Smarty	**X**	**X**	**X**
Generación de Reportes (Múltiples Formatos)	**X**		

HTML Helpers	X	X	X
Plug-Ins (Integración Terceros)	X	X	X
i18n (Internacionalización)		X	X
Efectos Visuales	X	X	X
Integración AJAX	X	X	X
Componente de Sesiones	X	X	X
Loggers	X	X	
Documentación en Inglés		X	X
Documentación en Español	X	X	
ACL (Access Control Lists)	X	X	X
Soporte MySQL	X	X	X
Soporte PostgreSQL	X	X	X
Soporte Oracle	X	X	

Soporte SQL Server	**X**	
Soporte SQLite	**X**	**X**

1.6. Lenguaje de Programación para la Web

En la actualidad existen diferentes lenguajes de programación, estos han surgido debido a las tendencias y necesidades de las plataformas existentes. A continuación se hará un análisis de las más importantes.

Desde el comienzo de Internet, fueron surgiendo diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes estáticos. En el transcurso del tiempo, las tecnologías fueron desarrollándose y surgieron nuevos problemas a dar solución. Esto dio lugar a desarrollar lenguajes de programación para la web dinámica, que permitieran interactuar con los usuarios y utilizaran sistemas de bases de datos. A continuación se presenta una introducción a los diferentes lenguajes de programación para la web.

1.7.1. Lenguaje HTML

Desde los inicios de la Internet han surgido y publicado muchos sitios Web usando el lenguaje HTML de sus siglas en inglés (HyperText Markup Language), que quiere decir Lenguaje de Marcas de Hipertexto, desarrollado por el World Wide Web Consortium (W3C). Este lenguaje marcado predomina en todos los sitios Web. Se usa para describir la estructura y el contenido en forma de texto, así como poner en el texto objetos como márgenes, entre otros. Es un lenguaje etiquetado, rodeado por corchetes angulares (<,>). Puede describir hasta cierto punto, la apariencia del documento, puede incluir script como por ejemplo Javascript.

1.7.2. PHP

Procesador de Hipertexto, es gratuito e independiente de plataforma, rápido, con una amplia librería de funciones y mucha documentación. Fue creado en 1994 por Rasmus Lerdorf. Se ejecuta del lado servidor Web, pero puede ser utilizado desde una interfaz de líneas de comandos o en la creación de otros programas como aplicaciones de interfaz gráfica, mediante el uso de bibliotecas Qt o GTK+. Fue diseñado

originalmente para la creación de páginas Web. Es un lenguaje orientado a objeto, lo que permite la reutilización de código y un rápido desarrollo. Posee compatibilidad con las bases de datos más comunes, como MySQL, Oracle y PostgreSQL entre otros. Proporciona soporte para diferentes protocolos de comunicación conocidos entre los cuales se tienen:

- HTTP (Protocolo de Transferencia de Hipertexto).
- IMAP (Protocolo de Acceso a Mensajes de Internet).
- FTP (Protocolo de Transferencia de Archivos).
- LDAP (Protocolo Ligero de Acceso a Directorios), entre otros.

Es publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

1.7.3. JavaScript

Es un lenguaje de script no interpretado, no requiere de un compilador que se ejecuta de lado del cliente, por lo que es interpretado por el navegador y fue desarrollado por Netscape Communication. Es orientado a Objeto, y permite acceder a objetos en aplicaciones Web. JavaScript ha tenido influencia de diferentes lenguajes y fue diseñado con sintaxis similar a Java, aunque más fácil de utilizar por programadores principiantes. Permite realizar las validaciones del lado del cliente, permitiendo el acceso innecesario al servidor Web, posibilitando que dicho servidor tenga más funcionalidad a la hora de responder peticiones de los usuarios. Es interpretado por la mayoría de los navegadores más comunes usados como Netscape, Internet Explorer, Mozilla Firefox, Opera, entre otros.

1.7. Herramientas de Desarrollo

Para desarrollar normalmente solo es necesario un editor de texto, un intérprete o compilador y una Terminal de líneas de comando, pero siempre es más rápido y fácil si se usan algunas herramientas o un Entorno de Desarrollo Integrado (IDE) simplificando el trabajo y ahorrando tiempo de desarrollo. Al seleccionar una herramienta, es importante que cumpla con los requerimientos que se necesitarían para desarrollar el software, en esto se debe tener en cuenta la tecnología que se va a usar y la plataforma de desarrollo.

1.8.1. Rational Rose

Rational Rose es una herramienta de modelado visual con plataforma independiente, que permite la comunicación entre los miembros del equipo. Tiene como ventaja que utiliza la notación estándar en la arquitectura de software (UML), posibilita a los arquitectos y desarrolladores de software visualizar todo el sistema mediante el uso de un lenguaje común. Mantiene la consistencia de los modelos del sistema de software. Permite chequear las sintaxis UML y realizar ingeniería inversa. Posibilita la generación de documentación automáticamente y la generación de código a partir de los modelos. A través de él los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con los demás componentes del proyecto.

1.8.2. Visual Paradigm

Es una herramienta CASE que utiliza UML como lenguaje de modelado, soporta un ciclo de vida completo de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Es conocido por las siglas VP UML. Permite una rápida construcción de aplicaciones con una excelente calidad, mejores y a un costo mínimo de desarrollo. Posee características que hacen del Visual Paradigm una herramienta profesional y ampliamente utilizada, entre las cuales se tienen:

- Modelado colaborativo mediante el uso de CVS (Sistema de Control de Versiones) y Subversión.
- Capacidades de ingeniería directa e inversa.
- Generación de código (Modelo a código, diagrama a código).
- Generación de bases de datos (Transformación de diagramas de Entidad-Relación en tablas de base de datos).
- Importación y exportación de ficheros XML de Intercambio de Metadatos (XMI).
- Generador de informes para generación de documentación.

1.8.3. Net Bean

Es un Entorno de Desarrollo Integrado (IDE), libre, de código abierto (Open-Source) para desarrolladores de software. Presenta todas las herramientas que se necesita para crear Aplicaciones de escritorio profesionales, aplicaciones Web y aplicaciones móviles con el lenguaje Java, C / C + +, e incluso los

lenguajes dinámicos como PHP, Javascript, Groovy y Ruby. El IDE NetBeans es fácil de instalar y utilizar directamente, se ejecuta en varias plataformas, incluyendo Windows, Linux, Mac OS X y Solaris.
El NetBean posee un plug-ing que permite desarrollar aplicaciones Web en PHP, y la versión actual trae incorporado soporte para el desarrollo en PHP utilizando el Framework de Symfony, posee muchas funciones que hacen posible un mejor desarrollo con este Framework.

1.8. Sistemas Gestores de Bases de Datos

Una base de datos o banco de datos como también se conoce, no es más que un conjunto de datos consistente y usualmente persistente en el mismo contexto, almacenados sistemáticamente para su posterior uso, organizado en un modo específico que permita acceder y modificar a la información de forma fácil y rápida.
Actualmente debido al desarrollo tecnológico que ha surgido en el campo de la informática y la electrónica, la mayoría de la información de las bases de datos están en formato digital, por lo que ofrece un amplio rango de soluciones y propuestas al problema de almacenar datos.
Las bases de datos son accedidas principalmente a través de un Sistema Gestor de Base de Datos (SGBD), que no es más que un programa que permite crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Un SGBD debe permitir llevar un control de la redundancia, el acceso no autorizado, y el cumplimiento de las restricciones de integridad. Las características y propiedades de estos SGBD, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Los SGBD deben ser capaz de:

- Definir la estructura de una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

1.9.1. PostgreSql

Es un poderoso sistema de bases de datos relacional, de código abierto, publicada bajo licencia BSD. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de confiabilidad, integridad de datos y corrección. Corre sobre diversos sistemas operativos, incluyendo GNU/Linux, UNIX y Windows. Soporta el almacenamiento de números binarios de gran

tamaño, incluyendo imágenes, sonido y vídeo. Es totalmente compatible con ACID, tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas). Se incluye la mayoría de SQL: 2008 tipos de datos, incluyendo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, y TIMESTAMP. Ejecuta procedimientos almacenados en varios lenguajes de programación como Java, Perl, Python, Ruby, C, C + +, y PL / pgSQL, entre otros. PostgreSQL posee con sofisticadas funciones como la versión multi-Control de concurrencia (MVCC), punto en el tiempo de recuperación, tablespaces, replicación asincrónica, transacciones anidadas (puntos de retorno), en línea / backups en caliente, un planificador de consultas sofisticadas / optimizadas, y escribir por delante de registro para la tolerancia a fallos.

Algunas de sus principales características son, entre otras:

- Tamaño máximo de la base de datos ilimitada.
- Tamaño máximo de la tabla 32 TB.
- Tamaño máximo de fila de 1,6 TB.
- Tamaño máximo de campo 1 GB.
- Máximo Filas por tabla ilimitado.
- Máximo columnas por tabla 250 - 1600 dependiendo de los tipos de columna.
- Índices máximos por el cuadro ilimitado.

1.9.2. MySql

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiuso licenciado bajo la GNU GPL. Su diseño le posibilita soportar una gran carga de forma muy eficiente. MySQL fue creado por la empresa sueca MySQL AB, que mantiene el CopyRight del código fuente del servidor SQL, así como también de la marca.

MySQL popular herramienta de código abierto del mundo de software de base de datos, con más de 100 millones de copias de su software descargado o distribuido a lo largo de su historia. Con su velocidad, fiabilidad y facilidad de uso, MySQL se ha convertido en la opción preferida para la Web, Web 2.0, SaaS, empresas de telecomunicaciones, ya que elimina los problemas más importantes asociados con el tiempo de inactividad, mantenimiento y administración modernos y de aplicaciones en línea.

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

- Aprovecha la potencia de sistemas multiprocesadores a la hora de realiza las búsquedas de datos.
- Soporta gran cantidad de tipos de datos para las columnas.
- Posee API's que permiten el desarrollo en gran cantidad de lenguajes (C, C++, Java, PHP, etc.).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.*(*Pecos, 2002)

1.9. Propuesta

Después de realizar un estudio de las tecnologías existentes que pueden ser utilizadas para el desarrollo de la herramienta, se decide seleccionar las siguientes herramientas, las cuales se adaptan a nuestro perfil y dan respuestas a las necesidades que se tienen para la realización del software.

Se propone como lenguaje de programación del lado del servidor PHP en su versión 5.2, ya que es un lenguaje liviano para desarrollo de aplicaciones Web, uno de los más utilizados en el mundo, presenta una amplia documentación, es gratuito e independiente de la plataforma donde se quiera utilizar. Como lenguaje de programación del lado de cliente tenemos JavaScript y HTML, ya que JavaScript permite realizar validaciones del lado del cliente, es orientado a objeto y es el lenguaje más común que se utiliza para el trabajo del lado del cliente; y HMML como lenguaje etiquetado, el cual brinda la posibilidad de un mejor manejo del documento Web.

Para llevar a cabo el desarrollo en PHP se empleará el NetBean, como entorno de desarrollo integrado, para PHP en su versión 6.8, ya que esta versión incorpora un módulo para el trabajo con Symfony, muy amigable y fácil de utilizar. Como sistema gestor de Base de Daros se utilizará MySQL en su versión 5.3, ya que es muy sencillo de utilizar, se integra perfectamente con el framework Symfony, es libre y una de las más utilizadas en sistemas que no requieran de mucho procesamiento de información por parte del gestor de bases de datos.

Después de analizar los Framework existentes para PHP, se selecciona Symfony en su versión estable más reciente 1.12, ya que es un Framework que tiene una amplia comunidad a nivel mundial, mucha documentación en inglés y en español, también posee una gran variedad de plug-ing, que permiten

agilizar el trabajo. Es un framework gratuito y de código abierto. Es compatible con la SGBD que vamos a utilizar.

Como lenguaje de modelado se propone usar UML, ya que permite modelar aplicaciones orientadas a Objetos. Para el proceso de desarrollo se empleará la metodología RUP, la cual es una de las más utilizadas en la UCI, además documenta de manera muy buena el proceso de desarrollo de software y que permite adecuarla a nuestra solución. Para visualizar el proceso se utilizará como herramienta CASE Visual Paradigm UML 6.1.

Se han seleccionado las herramientas llevando a cabo la línea de desarrollo que se quiere implementar en la Universidad de que todos los recursos que utilizaremos para la realización de cualquier sistema sean en software libre. También estas herramientas son las más usadas en el mundo y en la Universidad.

Conclusiones

En este capítulo se analizó el estado actual de las aplicaciones de evaluación de proceso de desarrollo del software en el mundo, en Cuba y en la UCI. Se investigaron y compararon las principales tecnologías, herramientas y lenguajes existentes en el mundo. Se hace énfasis en las características más importantes en su selección para la elaboración de la herramienta que se pretende implementar.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

En el presente capítulo se abordan las características más relevantes del sistema. Se realiza además un análisis de los procesos que serán objetos de automatización. Se detallan los requerimientos funcionales y no funcionales que debe poseer el sistema, se muestran los casos de uso, los actores que intervienen, así como una descripción detallada de los mismos.

2.1. Objeto de automatización

Los procesos fundamentales que serán objeto de automatización son los relacionados con la evaluación de los proyectos basándose en el modelo CMMI.Uno de estos procesos es la Gestión de indicadores, con el objetivo de evaluar los avances del proyecto con respecto a estos. Así como gestionar reporte, en el cual se muestra el resultado de un proyecto después de ser evaluado.

2.2. Propuesta de Sistema

Las principales funcionalidades del sistema son:

- Gestionar Preguntas
- Gestionar Usuarios.
- Gestionar Rol.
- Gestionar Proyectos.
- Gestionar Proyectos.
- Realizar Evaluación.
- Generar Reportes.
- Obtener resultados.

Dicho sistema contará con un menú horizontal con submenú desplegable donde se encontrarán cada una de las funciones que conforman el sistema. También contará con soporte para exportar cada uno de los diferentes reportes a formatos PDF e imprimir la información que se genere en los mismos.

2.3. Algoritmo

Para saber el valor máximo de cada variable en específico, el algoritmo divide los valores máximos de las variables padres entre la cantidad de variables que lo componen.

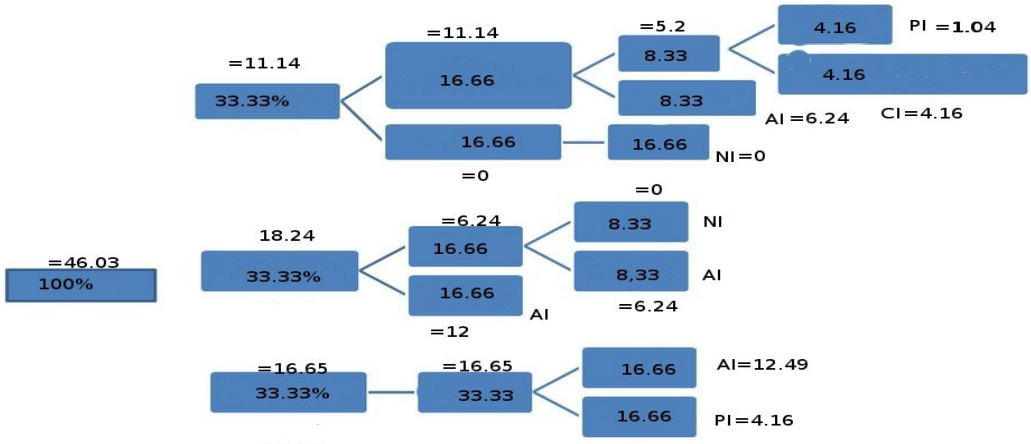
De acuerdo a lo que seleccione el evaluador se le dará a esta variable evaluada su valor real, tomando así valores cuantitativos como se muestran a continuación:

- ✓ Si selecciona “Completamente Implementada (CI)” el valor real es 100%.
- ✓ Si selecciona “Ampliamente Implementada (AI)”” el valor real es 75%.
- ✓ Si selecciona “Parcialmente Implementada (PI)”” el valor real es 25%.
- ✓ Si selecciona “No Implementada (NI)”” el valor real es 0%.

De acuerdo a lo que el evaluador seleccione, el algoritmo calcula lo que representa su valor real de su valor máximo. Cuando calcule todos los valores reales de cada variable evaluada se suman sucesivamente hasta llegar las preguntas padres y da el porciento de acercamiento en que se encuentra realmente el proyecto al programa de mejoras.

A continuación se describe un ejemplo del algoritmo para una mejor comprensión, para ello se basa su explicación en la figura. La variable padre (100%) es dividida entre la cantidad de hijos (100/3). Así sucesivamente con todas las variables que sean padres hasta llegar a las últimas preguntas hijas.

En el momento de dar la evaluación el sistema verifica que todas las preguntas estén respondidas, si es así automáticamente, calcula el %, según la respuesta dada para cada pregunta, en el caso que sea parcialmente implementada tiene como valor 25%, se procede a calcular el 25% del valor de la pregunta evaluada, una vez calculado se suman los valores hasta llegar al padre y este muestra el resultado en que se encuentra el proyecto.



2.4. Modelo del negocio

El modelo del negocio se centra en los límites o fronteras de las organizaciones y el establecimiento de las entidades con quien necesita comunicarse. Se encarga de definir los flujos de trabajo internos de la organización y de encontrar fórmulas para la optimización de estos. Uno de los elementos que define el modelo de negocio son los actores, los cuales son las entidades externas que interactúan con la organización. (Larman, 2004)

2.5. Descripción de los Actores

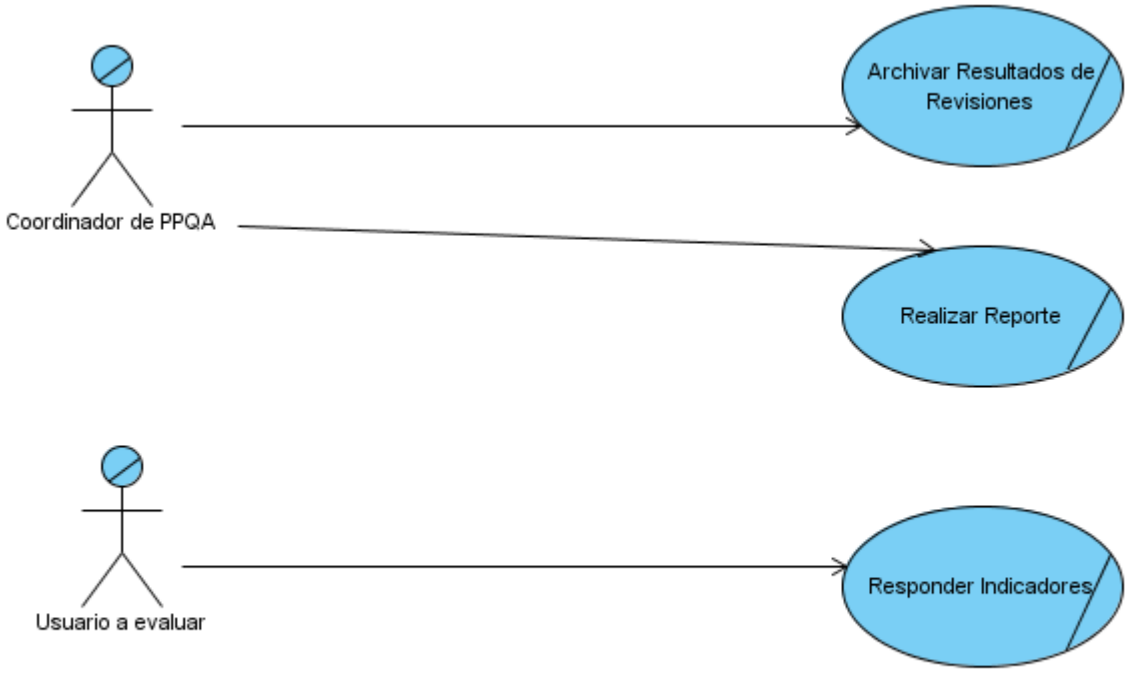
Actores	Justificación
Usuario a evaluar	Persona que da respuesta a los indicadores.
Coordinador de PPQA	Persona que distribuye los diferentes proyectos a evaluar entre los especialista de calidad.

2.6. Descripción de los Trabajadores

Trabajadores	Justificación
Especialista de calidad	Persona que lleva a cabo el proceso de evaluación.

2.7. Diagrama de Casos de Uso del Negocio

El diagrama de casos de uso del negocio representará los casos de uso del negocio y los actores del negocio.



2.8. Descripción de los Casos de Uso del Negocio

A continuación se realiza una breve descripción de los casos de uso del negocio (CUN).

2.8.1. CUN Archivar resultados de revisiones

En este caso de uso se realiza el proceso de guardar el resultado de todos los proyectos revisados.

Descripción del CUN Archivar resultados de revisiones

Caso de Uso:	Archivar resultados de revisiones
Actores:	Coordinador de PPQA
Trabajadores:	Especialista de calidad
Resumen:	En este caso de uso se realiza el proceso de guardar el resultado de todos los

	proyectos revisados.
Precondiciones:	Que se haya ejecutado una revisión.
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Negocio
1. El Coordinador de PPQA solicita los documentos que contienen las respuestas de las revisiones realizadas. **3.** Archiva los documentos.	**2.** El Especialista de calidad entrega los documentos.
Flujos Alternos	
Acción del Actor	Respuesta del Negocio
Post condiciones	

2.8.2. CUN Realizar reporte

En este caso de uso se realizan los reportes de los proyectos encuestados.

Descripción del CUN Realizar reporte

Caso de Uso:	Realizar reporte
Actores:	Coordinador de PPQA
Trabajadores:	Especialista de calidad
Resumen:	En este caso de uso se realizan los reportes de estado de los proyectos con las evidencias encontradas durante la revisión
Precondiciones:	
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Negocio
1. El Coordinador de PPQA solicita los documentos que contienen las respuestas de las revisiones realizadas. 3. Realiza el reporte, en correspondencia a la información que aparece en los documentos que le fueron entregados.	2. El Especialista de calidad entrega los documentos.
Flujos Alternos	
Acción del Actor	Respuesta del Negocio
Post condiciones	

2.8.3. CUN Responder indicadores

En este caso de uso se lleva a cabo el proceso de resolver los indicadores que conforman la encuesta.

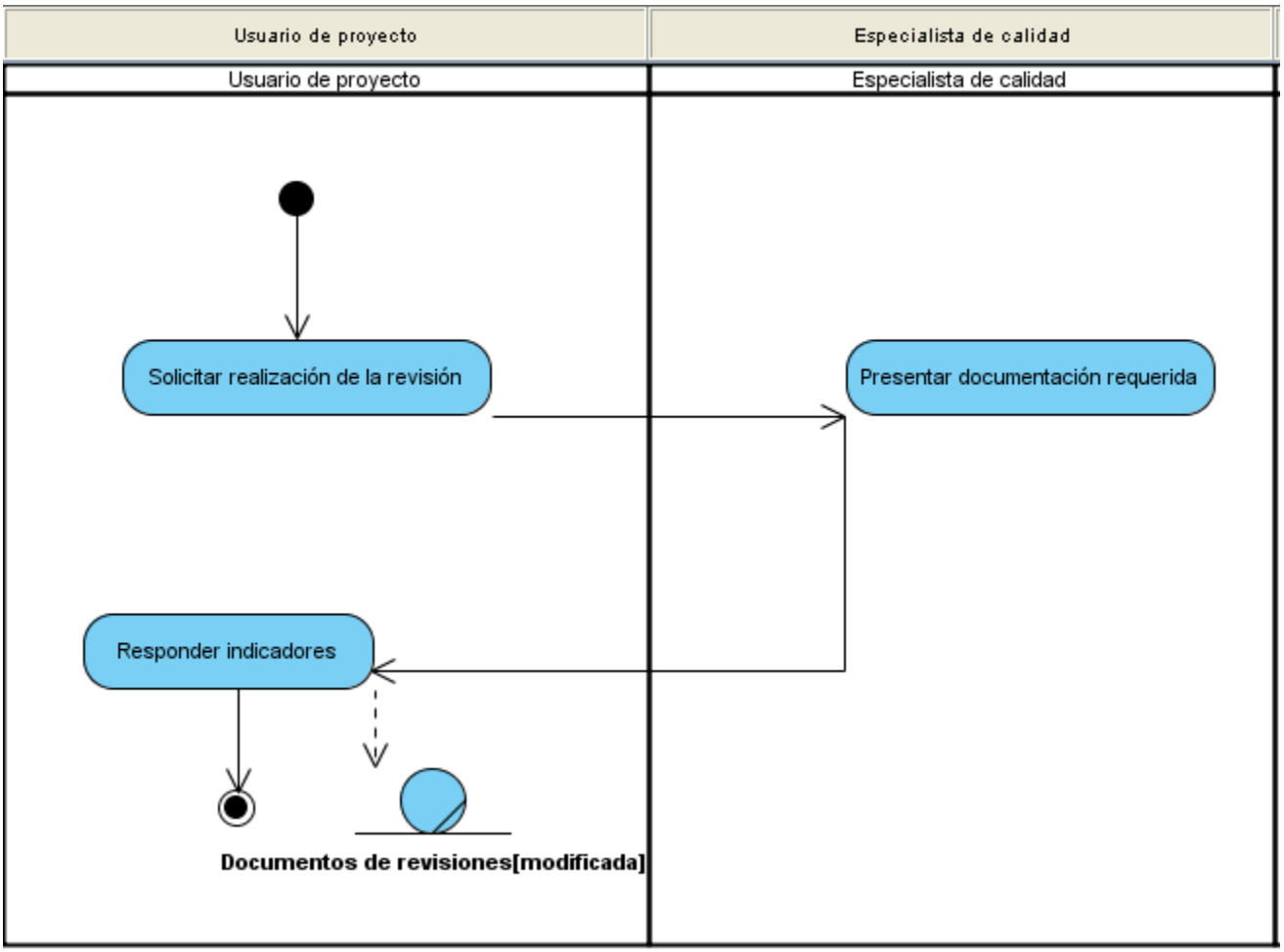
Descripción del CUN Responder indicadores

Caso de Uso:	Responder indicadores
Actores:	Usuario a evaluar
Trabajadores:	Especialista de calidad
Resumen:	En este caso de uso se lleva a cabo el proceso de resolver los indicadores que conforman la encuesta.
Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Negocio

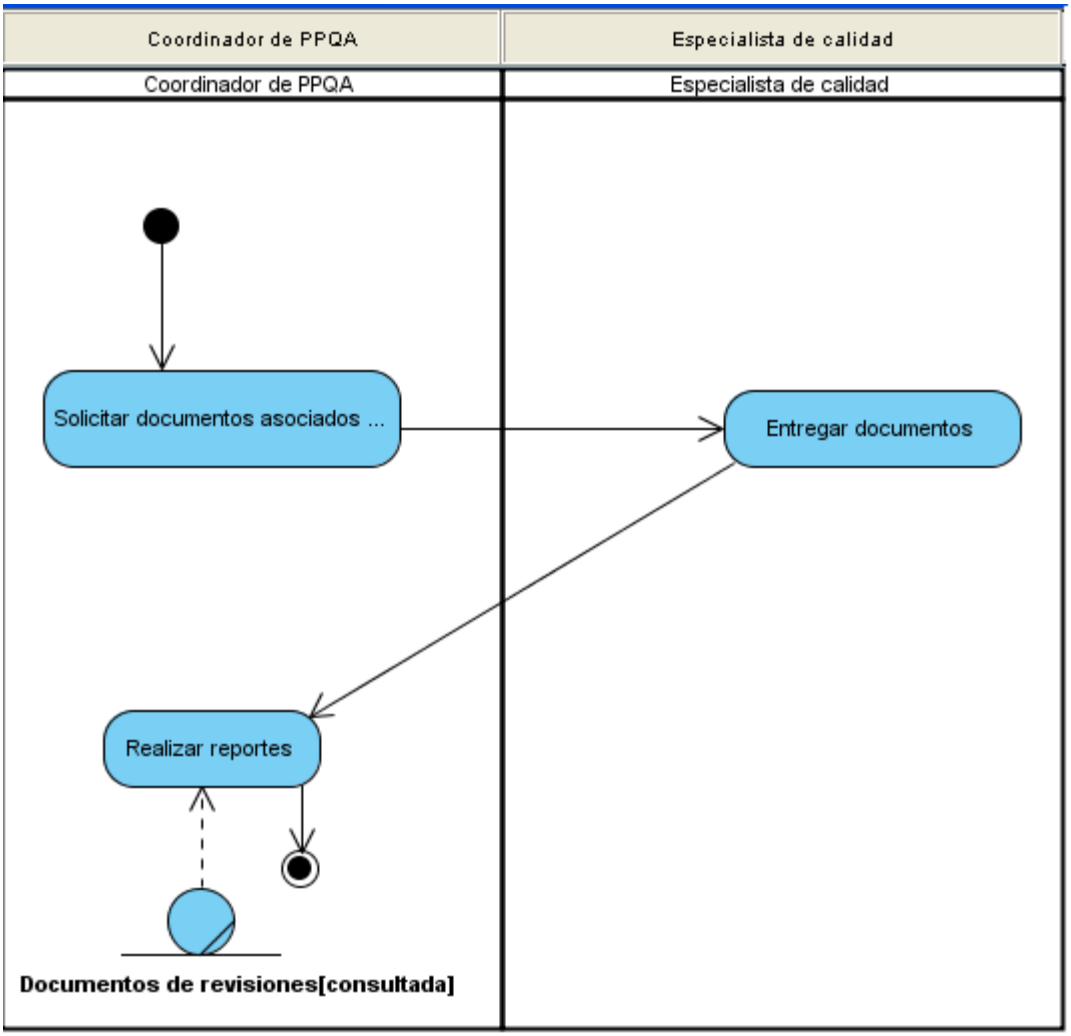
1. El usuario a evaluar solicita la revisión. 3. Responde los indicadores que conforman la encuesta.	2. El especialista de calidad se presenta con la documentación requerida para realizar las encuestas.
Flujos Alternos	
Acción del Actor	Respuesta del Negocio
Post condiciones	

2.9. Modelo de Objeto del Negocio

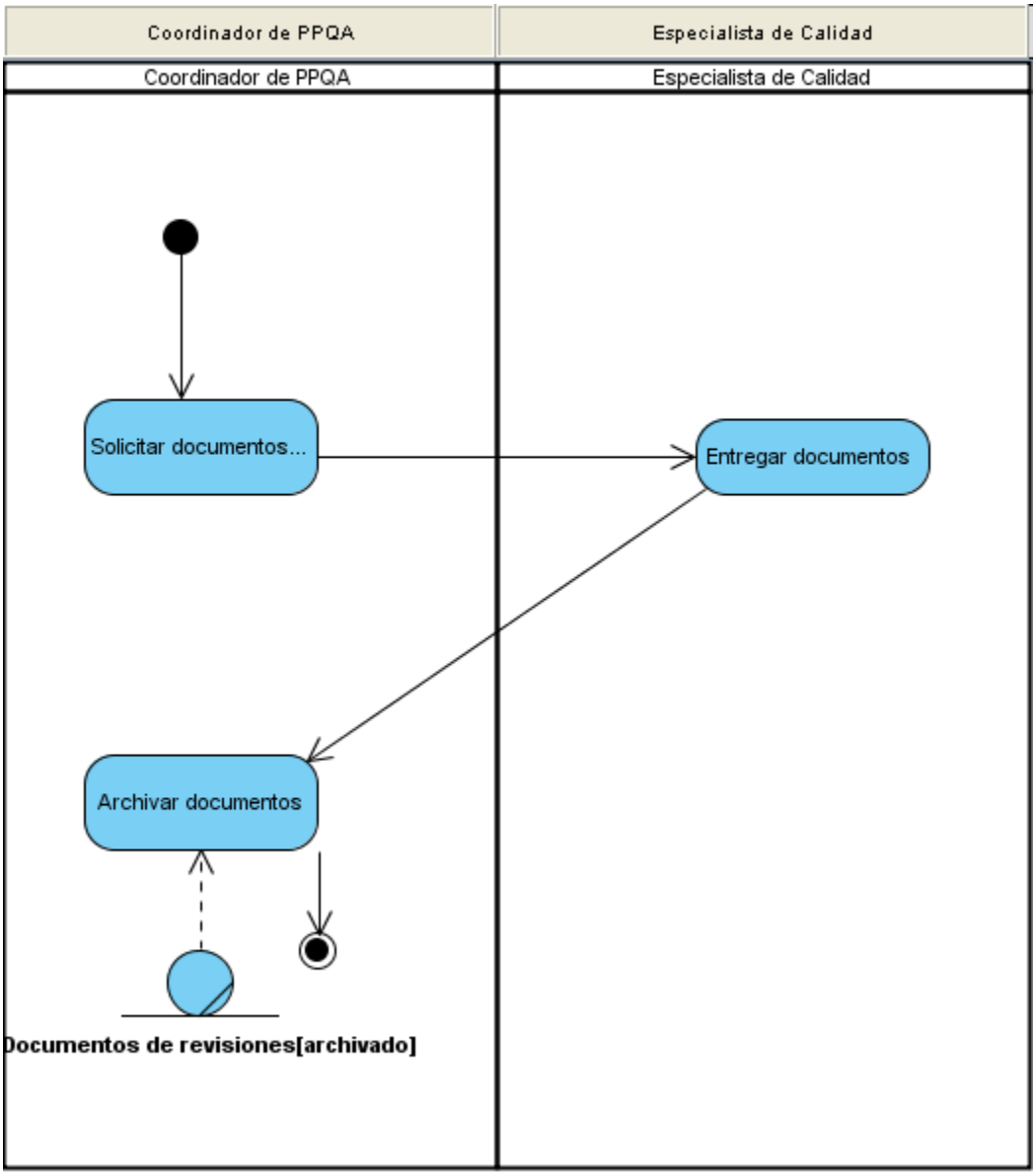
2.10. Diagrama de actividad del CUN Responder indicadores



2.11. Diagrama de actividad del CUN Realizar reporte



2.12. Diagrama de actividad del CUN Archivar resultados de revisiones



2.13. Especificación de los Requerimientos de Software

Con la especificación de los requerimientos de software se obtiene una descripción detallada de las necesidades de un producto informático. Estos se dividen en dos grupos para una mejor especificación, los requerimientos funcionales y los no funcionales. Los funcionales definen las capacidades que el sistema debe cumplir, o el qué debe hacer el sistema, mientras que los no funcionales son las propiedades que el mismo presenta, o el cómo debe hacerse. (Jacobson, 1999)

2.13.1. Requerimientos Funcionales

A continuación se muestran los requerimientos funcionales de la aplicación en su versión 1.0.

Id	Requerimientos	Descripción
RF 1.0	**Gestionar usuarios.**	Gestiona los usuarios que van a interactuar con la herramienta.
RF 1.1	Crear usuario.	Adiciona nuevos usuarios.
RF 1.2	Modificar usuario.	Modifica usuarios existentes.
RF 1.3	Eliminar usuario.	Elimina usuarios existentes.
RF 2.0	**Gestionar proyectos.**	Gestiona los proyectos que serán evaluados.
RF 2.1	Crear proyecto.	Crear nuevos proyectos.
RF 2.2	Modificar proyecto.	Modifica proyectos existentes.
RF 2.3	Eliminar proyecto.	Elimina proyectos existentes.
RF 3.0	**Gestionar preguntas.**	Gestiona las preguntas a evaluar.
RF 3.1	Crear preguntas.	Adiciona nuevas preguntas.
RF 3.2	Modificar preguntas.	Modifica preguntas existentes.

RF 3.3	Eliminar preguntas.	Elimina preguntas existentes.
RF 4.0	**Gestionar respuestas.**	Gestiona resultados de las respuestas.
RF 4.1	Crear respuesta.	Crear respuestas.
RF 4.2	Modificar respuesta.	Modifica respuestas existentes.
RF 4.3	Eliminar respuesta.	Elimina respuestas existentes.
RF 5.0	**Realizar evaluación.**	Realiza las evaluaciones por parte del especialista.
RF 5.1	Modificar evaluación.	Modifica una evaluación que no se encuentre terminada.
RF 6.0	**Gestionar rol.**	Gestiona los roles del sistema.
RF 6.1	Crear rol.	Crear rol.
RF 6.2	Modificar rol	Modifica el rol seleccionado.
RF 6.3	Eliminar rol.	Elimina el rol seleccionado.
RF 7.0	**Obtener resultado.**	Obtiene el resultado de la evaluación realizada.
RF 8.0	**Autenticar.**	Autentica los usuarios para poder acceder a la aplicación.

2.13.2. Requerimientos No Funcionales

Id	Requerimiento	Descripción
Usabilidad	Fácil empleo para usuarios sin experiencia	El sistema debe garantizar un acceso fácil y rápido. Podrá ser usado por cualquier usuario que posea pocos conocimientos informáticos y de un ambiente Web en sentido general. Debe tener similitud entre sus páginas y estar poco cargado con imágenes, lo

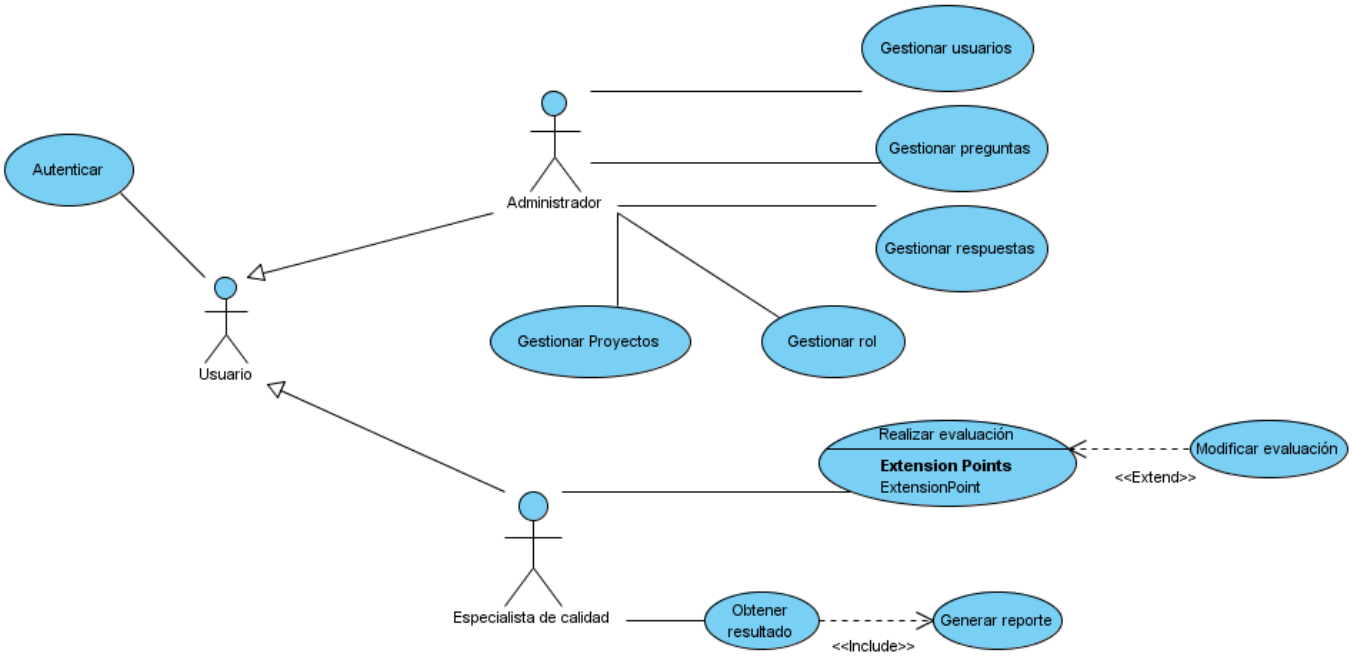
		que posibilita que se devuelvan respuestas de una manera eficiente en un tiempo mínimo, siendo más sencillo de entender y utilizar por el usuario.
Seguridad	Protección de la base de datos	
Seguridad	Reglas del negocio sobre la aplicación	La autenticación será la primera acción del usuario en el sistema y consiste en suministrar un nombre único de usuario y una contraseña que debe ser de conocimiento exclusivo de la persona que está involucrada en esta operación.
Rendimiento	Escalabilidad	El sistema debe ser capaz de mantener el mismo rendimiento y estabilidad a medida que aumenta la cantidad de datos a gestionar.
Soporte	Multiplataforma	El sistema debe descansar sobre un Servidor Apache/2.2.12, teniendo este un soporte para PHP 5.3, ya que el sistema debe ser multiplataforma y este servidor nos brinda esta funcionalidad.
Restricciones de Diseño	Gestor de bases de datos MySQL.	El sistema debe descansar sobre el gestor de bases de datos MySQL ya que es un gestor potente, seguro, liviano y estable que pertenece al movimiento de software libre.
Restricciones de Diseño	Compatibilidad con los diferentes navegadores	El sistema debe ser capaz de poderse ejecutar desde diferentes navegadores como Mozilla Firefox, Safari, Opera, e Internet Explorer.
Apariencia o Interfaz Externa		La interfaz debe ser sencilla y amigable ya que el usuario no es experto en el uso de las aplicaciones Web. Diseño sencillo, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema, además que exista paginación de reportes

		de búsqueda y listados.
Software		Para clientes: Se tendrá acceso al sistema a través del navegador Web. Firefox Mozilla 1.5 o superior que soporte DHTML y CSS2. Para Capa de Presentación: El servidor debe tener PHP Versión 5.3 Servidor HTTP, Apache. Sistema Operativo GNU/LINUX, distribución Debian 4 Etch , Windows 98 o superior y MAC. Para Capa de Datos: Servidor de Base de Datos MySQL Versión 5.1. Sistema Operativo GNU/LINUX, distribución Debian 4 Etch.
Hardware	Requerimientos mínimos para clientes	Ordenador Pentium o superior. 64 MB de Memoria RAM. Monitor VGA o superior. Teclado y Mouse. Procesador 486DX / 66 MHZ o superior. Disco duro de 60 GB o superior.
Portabilidad		Permitir que el sistema se ejecute sobre el Sistema Operativo Linux, Windows 98 o superior y MAC.

2.14. Definición de los Actores del Sistema

Actores	Justificación
Administrador	Es el actor que se encarga de gestionar todas las funcionalidades vinculadas a la administración de la aplicación.
Especialista de calidad	Persona que lleva a cabo el proceso de evaluación.

2.15. Diagrama de Casos de Uso del Sistema



2.16. Descripción de los Casos de Uso del Sistema

A continuación se realiza una descripción de los casos de uso del Sistema (CUS) Realizar evaluación y Gestionar usuarios. Los restantes pueden verse en el Anexo # 1.

Caso de Uso:	Realizar evaluación
Actores:	Especialista de Calidad
Resumen:	El caso de uso se inicia cuando el Especialista de Calidad selecciona la opción Nueva.
Precondiciones:	Debe existir un proyecto a evaluar.
Referencias	RF 5.0, RF 5.1

Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. EL Especialista de Calidad solicita la opción Nueva.	2. El sistema muestra una interfaz con los campos : Nombre del evaluado. Nombre del proyecto. Descripción del proyecto.
3. El especialista de calidad presiona el botón Siguiente.	4. El sistema muestra un menú desplegable con las 7 áreas de procesos pertenecientes al nivel 2 de CMMI.
5. El especialista de calidad selecciona el área de proceso que desea evaluar.	6. El sistema muestra un menú desplegable con las prácticas genéricas y específicas pertenecientes al área de proceso seleccionada.
7. El especialista de calidad, introduce los resultados y presiona el botón Guardar.	8. El sistema muestra un mensaje confirmando que terminó satisfactoriamente la evaluación.
Flujos Alternos Sección “Modificar evaluación”	
Acción del Actor	Respuesta del Sistema
7.1 El especialista de calidad presiona la opción Activas. 7.3 El especialista de calidad selecciona la encuesta que desea modificar o continuar, presionando el botón Modificar.	7.2 El sistema muestra un listado de todas las encuestas que se encuentran activas.

<table>
<tr><td>Caso de Uso:</td><td>Gestionar Usuario</td></tr>
<tr><td>Actores:</td><td>Administrador</td></tr>
<tr><td>Resumen:</td><td>El caso de uso se inicia cuando el administrador selecciona la opción crear usuario.</td></tr>
<tr><td>Precondiciones:</td><td></td></tr>
<tr><td>Referencias</td><td>RF1.0, RF 1.1, RF 1.2, RF 1.3</td></tr>
<tr><td>Prioridad</td><td>Crítico</td></tr>
<tr><td colspan="2">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>1. El administrador selecciona la opción Lista de Usuarios.</td><td>2. El sistema muestra una interfaz con las opciones Crear usuario, Modificar usuario y Eliminar usuario, así como un listado que contiene los siguientes datos de los usuarios existentes:
➢ Usuario
➢ Nombre completo
➢ Rol</td></tr>
<tr><td colspan="2">Prototipo de Interfaz</td></tr>
<tr><td colspan="2">Sección “Crear Usuario”</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>1. El administrador selecciona la opción Crear usuario.</td><td>2. El sistema muestra un formulario con los siguientes datos:
➢ Usuario</td></tr>
</table>

3. El administrador llena los campos y presiona el botón Crear.	➢ Correo ➢ Rol ➢ Proyecto ➢ Contraseña ➢ Repita contraseña ➢ Primer nombre ➢ Segundo nombre ➢ Primer apellido ➢ Segundo apellido 4. El sistema valida los datos entrados. 5. En caso de estar correctos, se crea el usuario. 6. Fin del flujo.
Flujos Alternos Sección “Crear usuario”	
Acción del Actor	Respuesta del Sistema
4.2 El administrador corrige los errores y presiona el botón crear.	4.1 En caso de haber errores, se muestra el formulario Crear usuario con los campos incorrectos señalados. 4.3 Fin del flujo alterno.
Prototipo de Interfaz	
Sección “Modificar usuario”	
Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción Modificar usuario.	2. El sistema muestra los datos del usuario seleccionado.

3. El administrador modifica los datos y presiona el botón guardar.	4. El sistema verifica los datos. 5. En caso de estar correctos, el sistema modifica el usuario. 6. El sistema muestra un formulario con el usuario modificado. 7. Fin del flujo
Flujos Alternos Sección “Modificar usuario”	
Acción del Actor	Respuesta del Sistema
4.2 El administrador corrige los errores y presiona el botón guardar.	4.1 En caso de haber errores, se muestra el formulario Modificar usuario con los campos incorrectos señalados. 4.3 Fin del flujo alterno.
Sección “Eliminar usuario”	
1. El administrador selecciona la opción Eliminar usuario. 3. El administrador presiona el botón Aceptar.	2. El sistema muestra un mensaje confirmación. 4. El usuario es eliminado.
Prototipo de Interfaz	
Post condiciones	1. Usuario creado en la base de datos.

Conclusiones

En el presente capítulo se ha realizado un estudio de la lógica del negocio del proceso de revisión que se realiza en el marco del programa de mejoras. Se identificaron los requerimientos a cumplir por parte del sistema, tanto funcional como no funcional. Se realizó la definición de los Casos de Usos del Sistema donde se encuentran las principales funcionalidades a implementar, además se llevo a cabo la descripción de cómo es la interacción entre los usuarios y los Casos de Usos del Sistema.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En este capítulo se desarrollan los diagramas de clases del diseño que participan en la realización de los casos de usos arquitectónicamente significativos. Se representa distintos diagramas de secuencia de dichos casos de usos. Se muestra una breve descripción de las clases Entidades y Controladoras empleadas y de las tablas de la base de datos, de la que se muestra a su vez los diagramas Entidad – Relación.

3.1. Flujos de Trabajo de Análisis y Diseño

El objetivo de esta disciplina es transformar los requerimientos a una especificación que describa cómo implementar el sistema. El análisis consiste en obtener una visión que se preocupa de ver qué hace el sistema de software a desarrollar, por lo cual se interesa solo en los requerimientos funcionales. Mientras que el diseño es un refinamiento del análisis que toma en cuenta los requerimientos no funcionales, por lo cual se centra en cómo el sistema cumple sus objetivos: tales como transformar los requerimientos al diseño del futuro sistema, desarrollar una arquitectura para el sistema, adaptar el diseño para que sea consistente con el entorno de implementación.

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los casos de uso con las interacciones de las clases de análisis. Durante esta fase se va refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes.

3.2. Modelo del Análisis

Este modelo es empleado con el fin de representar la estructura global del sistema, sirve como una abstracción del Modelo de Diseño. Es un primer intento por definir los conceptos clave que describen el sistema. Este artefacto es opcional, pero también tiene a su vez la propiedad de ser temporal. Su utilidad radica en que permite un acercamiento visual del sistema.

3.3.1. Diagrama de Clases del Análisis

El Diagrama de Clases del Análisis es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Está compuesto por clases y sus relaciones. Las clases del análisis se centran en los requerimientos funcionales, representan conceptos y relaciones del dominio. Estas poseen atributos y entre ellas pueden existir relaciones de asociación, agregación / composición, generalización/especialización.

RUP propone varias clasificaciones para estas clases, entre las que se destacan:

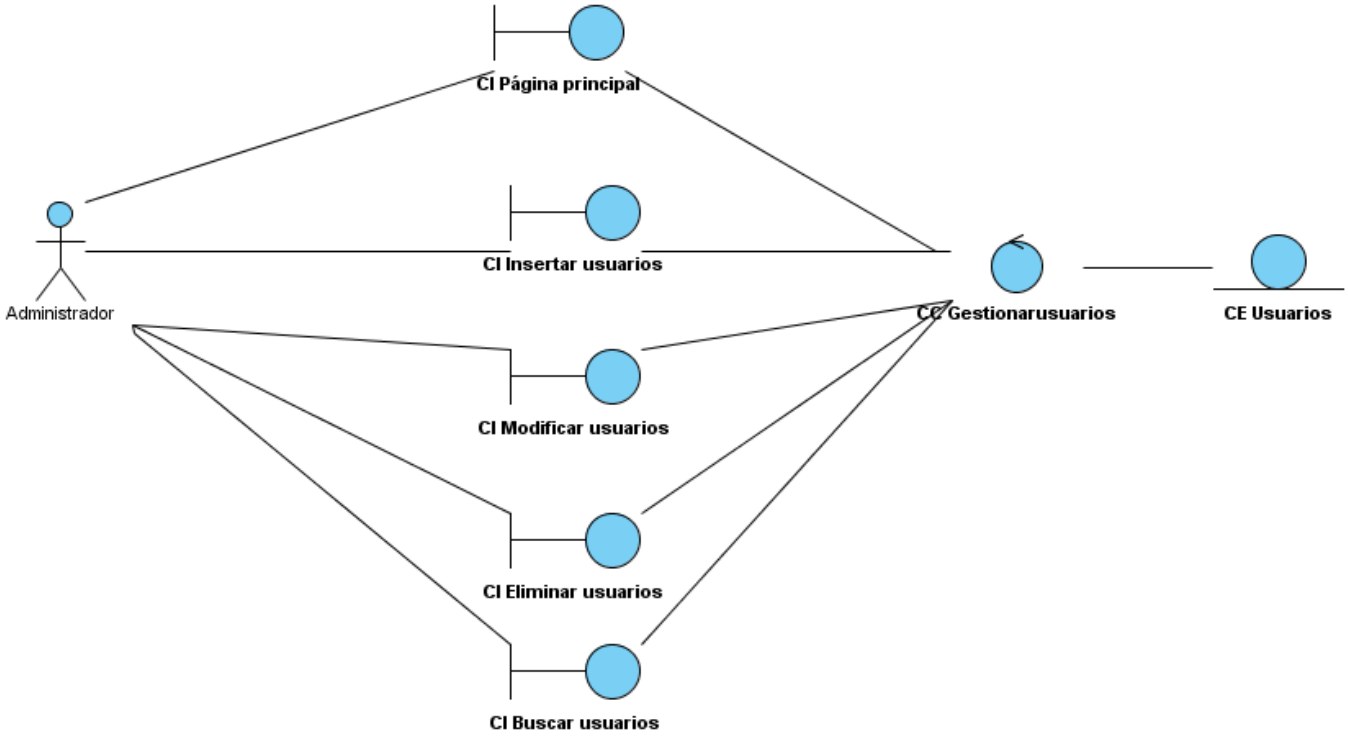
- **Clase Interfaz (CI):** Modela la interacción entre el sistema y sus actores.
- **Clase Controladora (CC):** Coordina la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.
- **Clase Entidad (CE):** Modela información que posee larga vida y es a menudo persistente.

La herramienta CASE Visual Paradigm, representa los tipos de clases a los que sé ha hecho referencia de la siguiente forma:

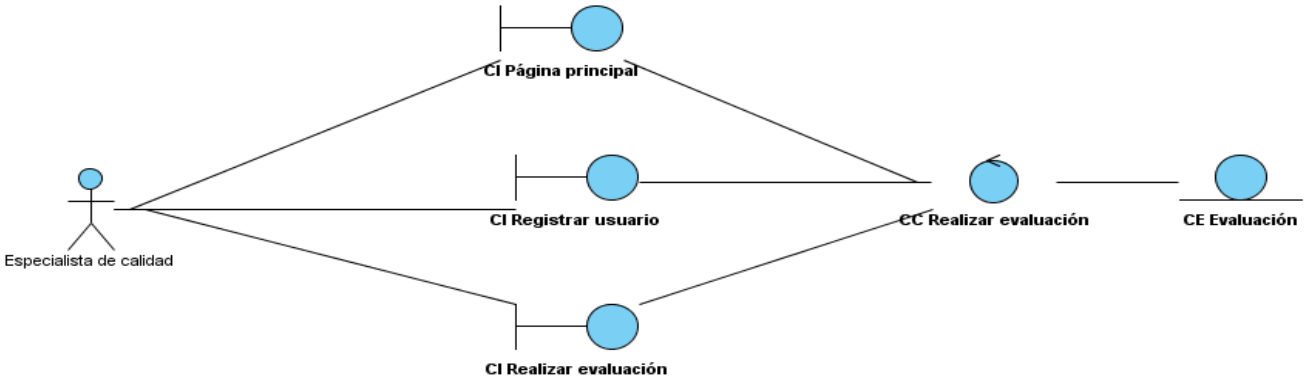
Figura 1: Representación de los tipos de clases del Análisis.

A continuación se muestran los diagramas de clases del análisis de los casos de uso Gestionar usuarios y Realizar evaluación, los restantes pueden verse en el anexo#2.

3.3.1.1. Diagrama de Clases del Análisis del CUS Gestionar usuarios



3.3.1.2. Diagrama de Clases del Análisis del CUS Realizar evaluación



3.3. Modelo de diseño

El Modelo de Diseño es un proceso para la definición detallada de un sistema con el fin de la realización física de los casos de uso para cubrir las funciones que realizará el sistema y otras restricciones del entorno de implementación que tienen impacto en el mismo, por tanto, en él se definen las clases del diseño que conformarán el sistema que se va a implementar.

3.4.1. Clases del diseño

- **Páginas servidoras:** Son las encargadas de la construcción de forma dinámica de las páginas clientes y sirven de enlace entre estas y el resto de las clases.
- **Páginas clientes:** Son las páginas encargadas de permitir a los usuarios interactuar con el sistema tanto para hacer solicitudes como para que sean mostradas las respuestas a las mismas.
- **Páginas controladoras:** Son las responsables de realizar las operaciones que responden a los procesos de negocio y dar respuestas a las solicitudes hechas por el usuario.
- **Clases entidad:** Son las responsables de la persistencia de los datos físicamente.

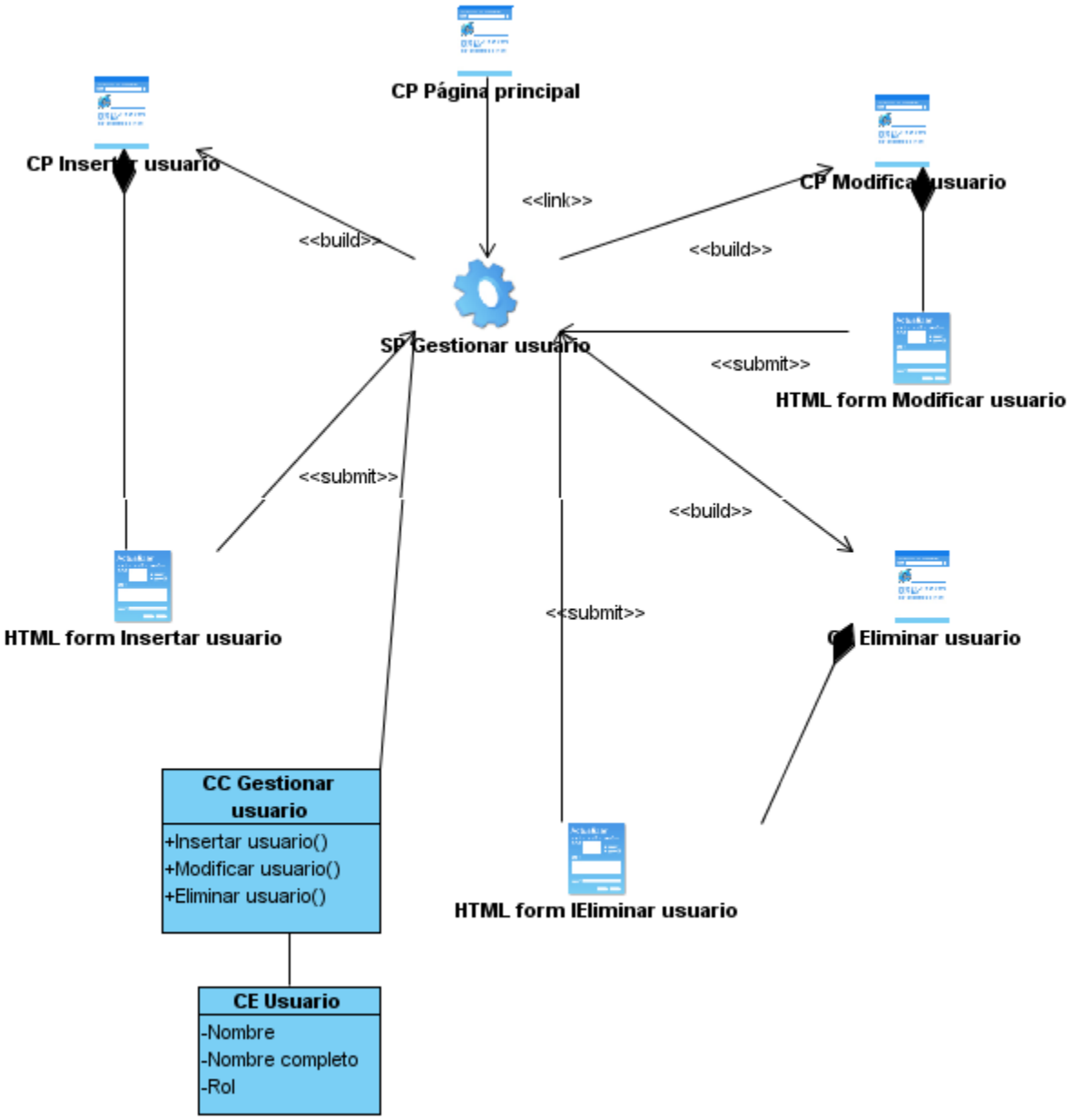
3.4.2. Diagrama de clases del diseño

En el diagrama de clases de diseño se especifican los atributos y métodos de cada clase .Son utilizados para modelar la vista de diseño estática de un sistema, esto incluye modelar el vocabulario del sistema, modelar esquemas o modelar las colaboraciones. Conforman la base para los diagramas relacionados:

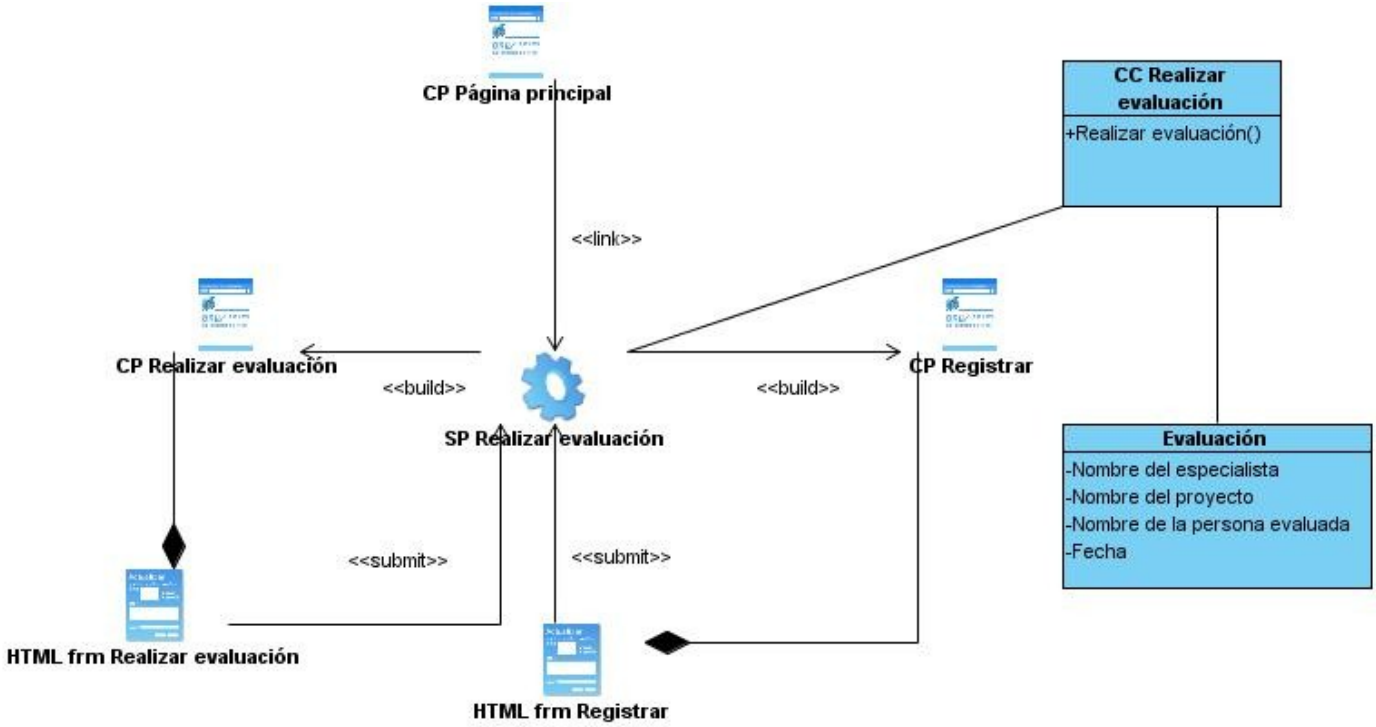
los diagramas de componentes y los diagramas de despliegue. Estos son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

A continuación se muestran los diagramas de clases del diseño con estereotipos web de los casos de uso Gestionar usuarios y Realizar evaluación, los restantes pueden verse en el **Anexo#3**.

3.4.2.1. Diagrama de Clases del Diseño con estereotipos web del CUS Gestionar usuarios



3.4.2.3. Diagrama de Clases del Diseño con estereotipos web del CUS Realizar evaluación



3.4. Descripción de las Clases

Nombre: CC Gestionar usuario	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Insertar usuario ()
Descripción:	Permite insertar un nuevo usuario.
Nombre:	Modificar usuario ()

Descripción:	Permite modificar un nuevo usuario.
Nombre:	Eliminar usuario ()
Descripción:	Permite eliminar un usuario.

Nombre: CE Usuario	
Tipo de clase: Entidad	
Atributo	**Tipo**
Nombre	String
Apellidos	String
Rol	String
Para cada responsabilidad:	

Nombre: CC Realizar evaluación	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Realizar evaluación ()
Descripción:	Se realiza una evaluación al proyecto determinado según los indicadores dados.

Nombre: CE Evaluación	
Tipo de clase: Entidad	
Atributo	**Tipo**
Nombre del especialista	String
Nombre del proyecto	String
Nombre de la persona evaluada	String
Fecha	String
Para cada responsabilidad:	

3.5. Arquitectura de Software

Con la evolución de los años se han ido desarrollando y descubriendo guías generales de aprendizaje para el desarrollo de programas. A estas guías se les nombrado de Arquitectura de Software, pues estas modelan la forma de desarrollar un determinado software informático. Esta, también dominada Arquitectura lógica está integrada por un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencias necesarias para guiar el desarrollo de un software. Establece las normas para que un equipo de desarrollo trabaje sobre una misma línea común permitiendo alcanzar los objetivos específicos de un sistema informático. A la hora de desarrollar un software se selecciona o se crea una arquitectura que rige el desarrollo del mismo. Algunas de las diferentes arquitecturas de software que existen son:

- Monolítica: Donde el software se estructura en grupos funcionales muy acoplados.
- Cliente-Servidor: Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.
- Arquitectura de tres niveles: Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación

(interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.

De las arquitecturas antes mencionadas se decidió tomar para el desarrollo de la aplicación la Arquitectura en tres Niveles o Capas.

3.6. Modelo Vista Controlador (MVC)

El MVC es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web para facilitar el ordenamiento del código y la simplicidad para crearlo, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

En resumen, el patrón de diseño MVC define una manera de organizar el código de acuerdo con su naturaleza. Este patrón separa el código en tres capas:

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

3.7. Patrones de Diseño

Un patrón de diseño es una descripción de un problema y su solución. Estos evitan la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados. Muchos patrones ofrecen además orientación sobre como asignar las responsabilidades a los objetos ante determinada categoría de problemas. Estos son los llamados patrones GRASP, siglas de Patrones Generales de Software para Asignación de Responsabilidades, aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software.

De este tipo de patrones serán usados durante el Modelado del Diseño: el Experto, Creador, Controlador, Alta Cohesión y Bajo Acoplamiento.

Experto: Indica que la responsabilidad de la creación de un objeto siempre se debe asignar al experto en información, es decir, la responsabilidad recae sobre la clase que conoce toda la información necesaria

para poder crearlo. El GRASP de experto en información es el principio básico de asignación de responsabilidades.

Beneficios: Se mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener.

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

La nueva instancia deberá ser creada por la clase que:

- Tiene la información necesaria para realizar la creación del objeto.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento.

Alta cohesión y bajo acoplamiento: Se puede separar, aunque están íntimamente ligados, de hecho, si se esfuerza en aumentar mucho la cohesión del software, es muy posible que se perjudique el acoplamiento aumentándolo, y por el contrario si se reduce mucho el acoplamiento, se verá disminuida la cohesión:

Alta cohesión: La información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase.

Bajo acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. 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, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

3.8. Seguridad

La seguridad en las aplicaciones es un tema muy importante y debe tomarse en cuenta antes de diseñar estas y no dejarla para el final como en ocasiones suele ocurrir; con el fin de obtener un producto con la seguridad requerida. La información sensible que se maneja, como es el caso de la información de los usuarios ha de protegerse, así como el acceso a la Base de Datos.

En la actualidad el desarrollo de aplicaciones Web debe realizarse empleando técnicas de programación seguras, para evitar los ataques de Identificación, Autorización, Ejecución de Comandos, Revelación de Información y Lógicos.

La información relacionada con cada uno de los usuarios será protegida con contraseña. Además se usará un método de encriptación para el manejo de identificadores.

El acceso a la base de datos se realizará a través de procedimientos almacenados en los que serán validados cada uno de los datos necesarios para que estos se ejecuten de manera correcta.

3.9. Interfaz

Se deberá garantizar un esquema de diseño, que por sus características dé continuidad a la herramienta en cuanto a logotipo, tipografía, e imagen corporativa, presentando un comportamiento flexible ante los cambios que pueda tener este. En cualquier momento de la historia de la navegación del usuario, este debe tener bien claro en qué punto de la estructura se encuentra y como volver al inicio o a puntos clave de la misma. El sistema además debe permitir que el usuario pueda retirarse del mismo en cualquier punto en que se encuentre de una forma sencilla.

3.10. Tratamiento de Errores

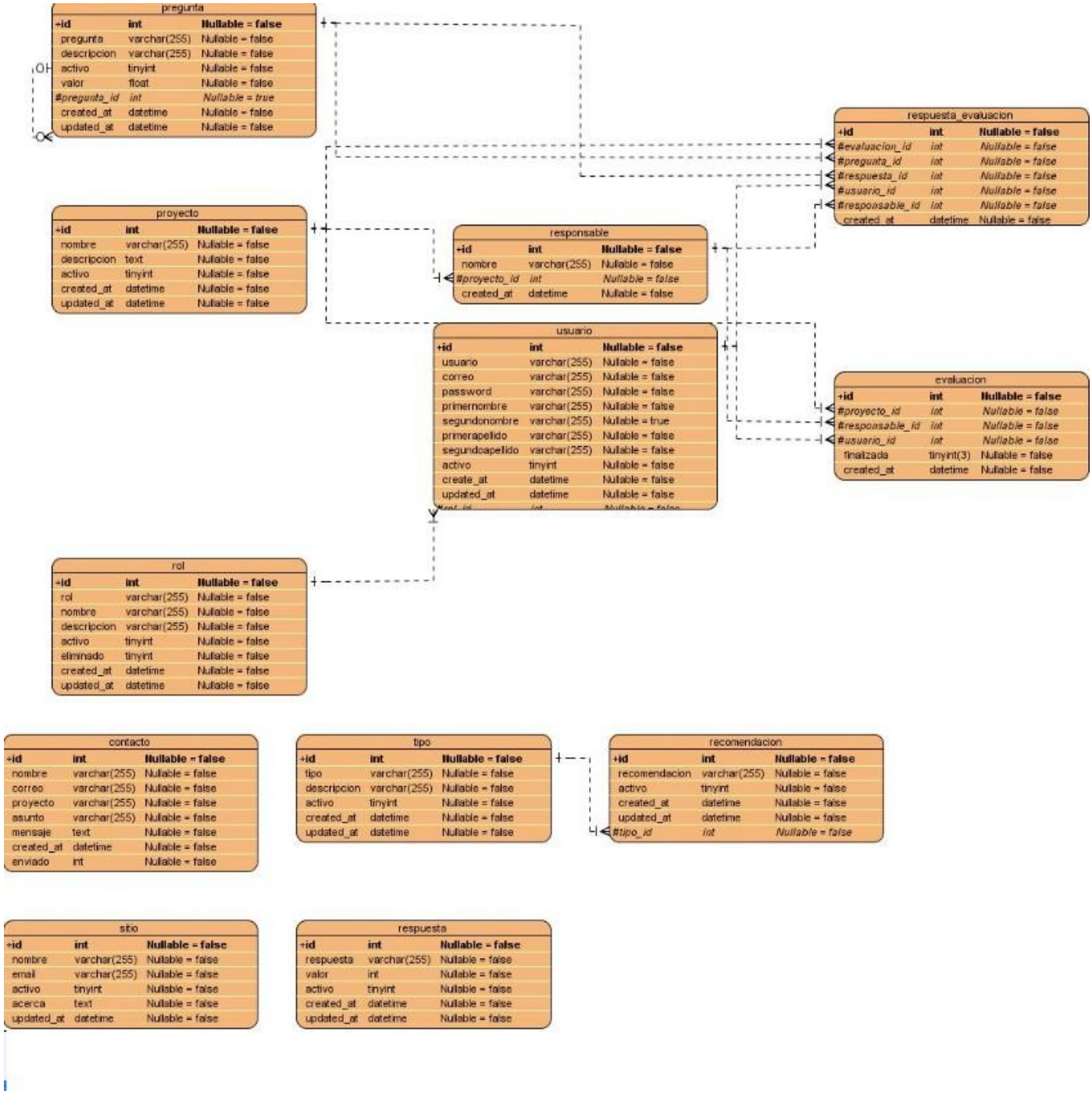
A partir de la identificación del proceso de tratamiento de errores la aplicación logra monitorear la ocurrencia de los mismos; mostrando mensajes de información al usuario que contengan el error ocurrido. Además se llevará un registro de los errores ocurridos para un posible seguimiento por parte de los administradores del sistema.

3.11. Diseño de Base de Datos

En la actualidad los sistemas web existentes son desarrollados utilizando tecnologías cliente-servidor. En la mayoría de los casos cuentan con bases de datos robustas que son las encargadas de almacenar todos los datos que en la misma se muestran. Se ha desarrollado una base de datos para almacenar toda la información que se controla.

3.13.1. Modelo de Datos.

El Modelo de Datos describe la representación lógica de las clases persistentes, previendo que información del sistema será soportada por una base de datos relacional.


pregunta
+id int Nullable = false
pregunta varchar(255) Nullable = false
descripcion varchar(255) Nullable = false
activo tinyint Nullable = false
valor float Nullable = false
#pregunta_id int Nullable = true
created_at datetime Nullable = false
updated_at datetime Nullable = false
respuesta_evaluacion
+id int Nullable = false
#evaluacion_id int Nullable = false
#pregunta_id int Nullable = false
#respuesta_id int Nullable = false
#usuario_id int Nullable = false
#responsable_id int Nullable = false
created_at datetime Nullable = false
proyecto
+id int Nullable = false
nombre varchar(255) Nullable = false
descripcion text Nullable = false
activo tinyint Nullable = false
created_at datetime Nullable = false
updated_at datetime Nullable = false
responsable
+id int Nullable = false
nombre varchar(255) Nullable = false
#proyecto_id int Nullable = false
created_at datetime Nullable = false
usuario
+id int Nullable = false
usuario varchar(255) Nullable = false
correo varchar(255) Nullable = false
password varchar(255) Nullable = false
primernombre varchar(255) Nullable = false
segundonombre varchar(255) Nullable = true
primerapellido varchar(255) Nullable = false
segundoapellido varchar(255) Nullable = false
activo tinyint Nullable = false
create_at datetime Nullable = false
updated_at datetime Nullable = false
evaluacion
+id int Nullable = false
#proyecto_id int Nullable = false
#responsable_id int Nullable = false
#usuario_id int Nullable = false
finalizada tinyint(3) Nullable = false
created_at datetime Nullable = false
rol
+id int Nullable = false
rol varchar(255) Nullable = false
nombre varchar(255) Nullable = false
descripcion varchar(255) Nullable = false
activo tinyint Nullable = false
eliminado tinyint Nullable = false
created_at datetime Nullable = false
updated_at datetime Nullable = false
contacto
+id int Nullable = false
nombre varchar(255) Nullable = false
correo varchar(255) Nullable = false
proyecto varchar(255) Nullable = false
asunto varchar(255) Nullable = false
mensaje text Nullable = false
created_at datetime Nullable = false
enviado int Nullable = false
tipo
+id int Nullable = false
tipo varchar(255) Nullable = false
descripcion varchar(255) Nullable = false
activo tinyint Nullable = false
created_at datetime Nullable = false
updated_at datetime Nullable = false
recomendacion
+id int Nullable = false
recomendacion varchar(255) Nullable = false
activo tinyint Nullable = false
created_at datetime Nullable = false
updated_at datetime Nullable = false
#tipo_id int Nullable = false
sitio
+id int Nullable = false
nombre varchar(255) Nullable = false
email varchar(255) Nullable = false
activo tinyint Nullable = false
acerca text Nullable = false
updated_at datetime Nullable = false
respuesta
+id int Nullable = false
respuesta varchar(255) Nullable = false
valor int Nullable = false
activo tinyint Nullable = false
created_at datetime Nullable = false
updated_at datetime Nullable = false

3.13.2. Descripción de las tablas

Las principales entidades de la base de datos son descritas a continuación.

Contacto

Nombre: Contacto		
Descripción: En esta tabla se almacenan los datos de las personas que envían correos al sitio.		
Atributo	**Tipo**	**Descripción**
Id	Int(11)	Se almacena un id que se genera automático.
Nombre	Varchar(225)	Se almacena el nombre del usuario.
Correo (PK)	Varchar(255)	Se almacena el correo electrónico del sitio.
Proyecto	Varchar(255)	Se almacena el proyecto al que pertenece esa persona.
Asunto	Varchar(255)	Se almacena el asunto del mensaje.
Mensaje	Text	Se almacena el texto que se le enviara al sitio.
Created_at	Datetime	Se almacena fecha y hora en que fue creado el mensaje.
Enviado	Int(11)	Se almacena el estado de envío del mensaje.

Sitio

Nombre: Sitio		
Descripción: En esta tabla se almacena la configuración e información del sitio		
Atributo	**Tipo**	**Descripción**
Id	Int (11)	Almacena el id del sitio.
Nombre	Varchar(255)	Almacena el nombre del sitio.
Email	Varchar(255)	Almacena el correo electrónico del sitio.
Activo	Tinyint(1)	Almacena si la configuración esta activa o no.
Acerca	Text	Almacena la información que se le muestra al usuario para que sepa de qué trata este sitio.
Updated_at	Datetime	Almacena la fecha en que se actualiza esta información.

Pregunta_Respuesta

Nombre: Pregunta_Respuesta		
Descripción: En esta tabla se almacena la relación de las preguntas y las respuestas.		
Atributo	**Tipo**	**Descripción**
Id	Int (11)	Almacena el id.
Pregunta_id	Int (11)	Almacena la pregunta a realizar.
Respuesta_id	Int (11)	Almacena la respuesta.
Created_at	Datetime	Almacena cuando se creó esta relación entre

		pregunta y respuesta.
Activo	Tinyint(1)	Almacena si esta activa esta relación.

Conclusiones

En el capítulo se trataron temas relacionados con la arquitectura seleccionada, el análisis y el diseño del sistema que se presenta, entre los cuales se encuentran el Modelo de Clases del Análisis y Modelo de Clases del Diseño con estereotipos web. También se especificaron temas de la base de datos a usar, se desarrolla el Modelo Entidad Relación que contiene todo el modelo de datos que se ha elaborado para el soporte de los datos que se mantendrán almacenados en la aplicación.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS

Introducción

Durante el diseño se refinan las clases y sus relaciones con el fin de dar paso a la implantación del sistema. También en esta etapa, se toman las medidas necesarias para lograr un diseño del sistema consistente con el entorno en el que se implementará.

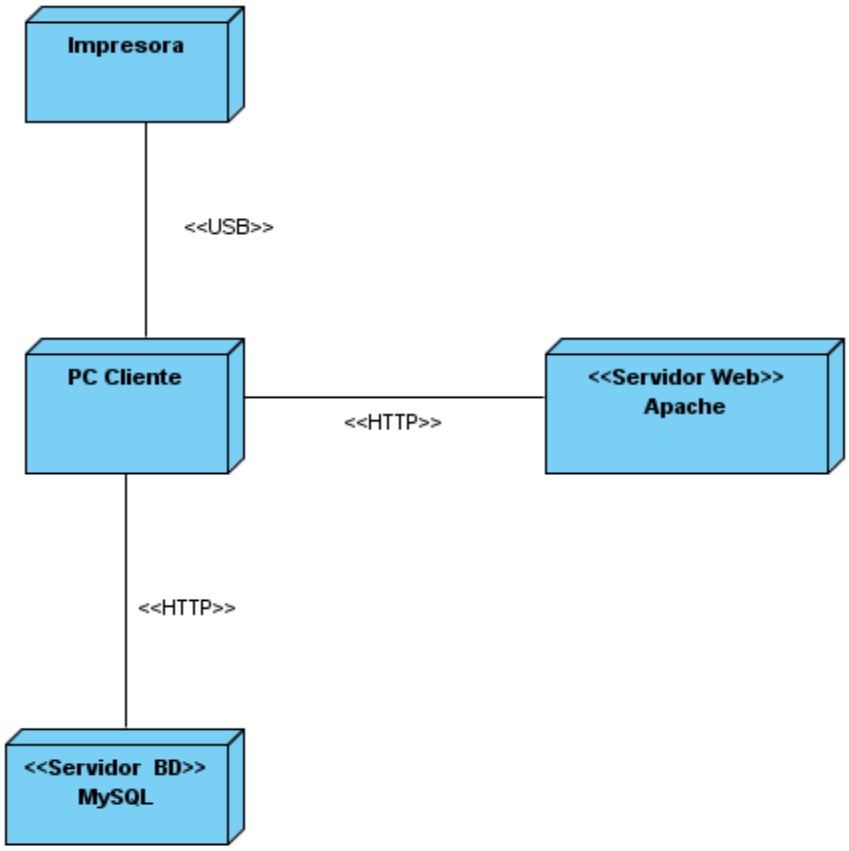
De una implementación correcta, que dé solución responda a los requisitos planteados, depende en gran medida que de las pruebas que se realizarán una vez implementado el sistema, se obtengan los resultados esperados por los desarrolladores.

4.1 Implementación

La implementación es el Flujo de Trabajo en que se desarrolla el sistema en términos de componentes: ejecutables, ficheros de código fuente, scripts, entre otros. Tiene como objetivo principal desarrollar la arquitectura y el sistema como un todo, así como definir la organización del código.

4.2 Modelo de Despliegue

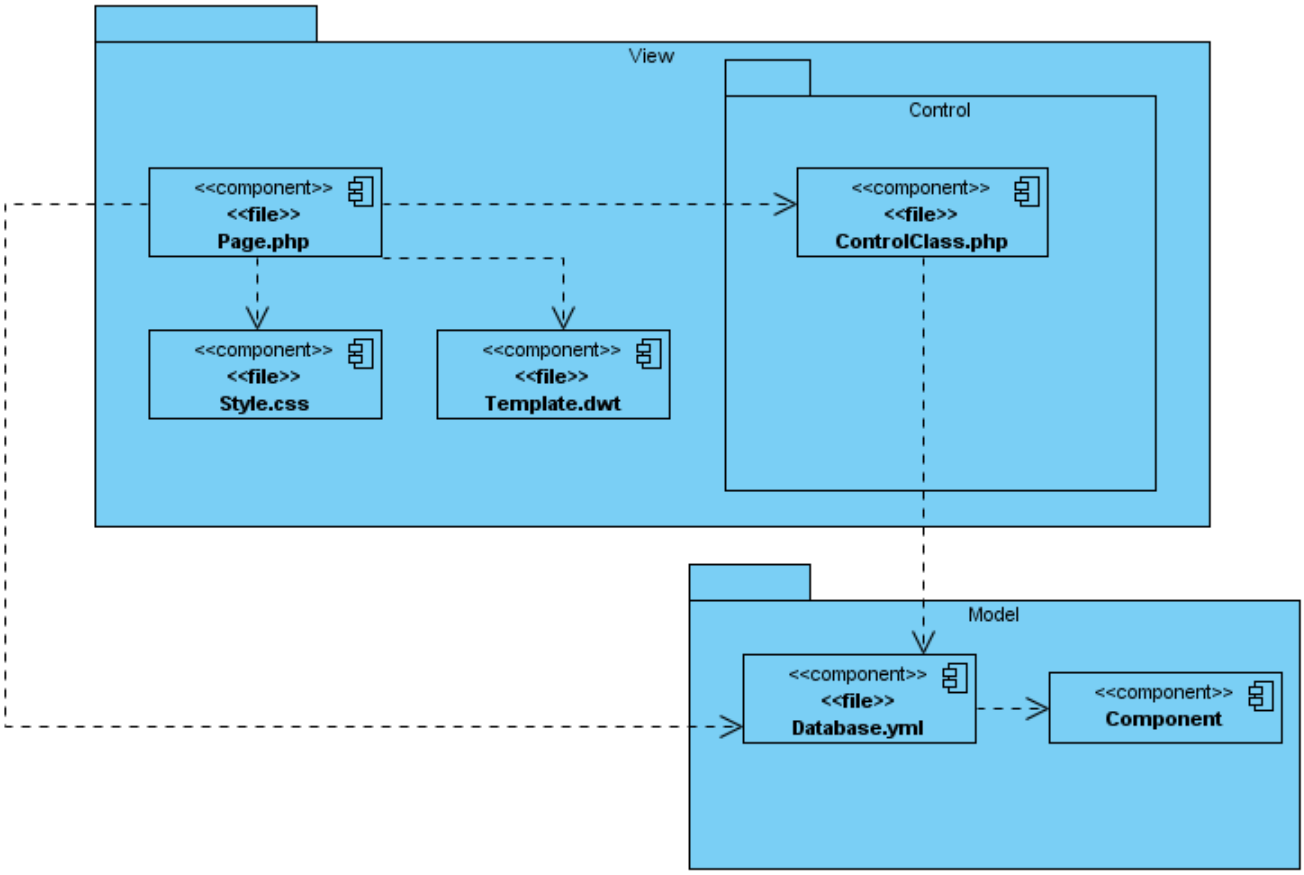
El Modelo de Despliegue es un modelo de objetos compuesto por nodos y sus relaciones, que describen la distribución física de un sistema.



4.3 Diagrama de Componentes

Un diagrama de componentes muestra las organizaciones y las dependencias lógicas de componentes software. Este puede contener paquetes que se utilizan para agrupar elementos del modelo.

Los Diagramas de Componentes han sido agrupados igualmente en paquetes para facilitar su comprensión y poder detallar mejor estos. Los paquetes que se mostrarán le dan seguimiento a los mostrados durante el diseño.



4.4 Prueba

Las pruebas se deben aplicar durante todo el ciclo de vida del software. Se deben planificar correctamente desde el inicio y establecer qué hacer, cómo hacer, quién va a hacer y en qué condiciones hacer las comprobaciones. En el desarrollo del software es normal que se encuentren errores desde que comienza el proceso, es por ello que éste va acompañado de una actividad de pruebas que garantice la calidad y constituye una guía donde se realiza la ejecución, de un sistema o componente, bajo condiciones o requerimientos especificados, los resultados son observados y registrados, y se emite una evaluación de algún aspecto del elemento probado. La prueba de software representa una revisión de las

especificaciones del diseño y la codificación, así como se ha establecido como un elemento crítico para garantizar la calidad del producto. El Flujo de Trabajo de Prueba tiene como objetivos encontrar y documentar los defectos que puedan afectar la calidad del software, validar que este trabaje como fue diseñado; así como probar los requisitos que debe cumplir el sistema y que estos fueron implementados correctamente

4.5 Pruebas de Caja Negra

Las Pruebas de Caja Negra son pruebas que se desarrollan en aplicaciones con interfaces gráficas. Este tipo de pruebas se encarga de verificar que las funciones que debe desempeñar el sistema son operativas. Se centran en los Requisitos Funcionales del software, sin interesarse por el funcionamiento interno del mismo. La realización de estas pruebas permite encontrar:

Funciones incorrectas.

Errores de interfaz.

Errores en las salidas.

Errores en el acceso a datos.

4.6 Resultados de las pruebas

Las pruebas realizadas a lo largo de todo el ciclo de desarrollo de la herramienta, mostraron que se obtuvo una aplicación con un alto grado de funcionalidad y 100% de cumplimiento de los objetivos propuestos.

El desarrollo de estas pruebas sirvió de apoyo a los desarrolladores para realizar su trabajo y localizar los errores que a simple vista no son detectados cuando se está programando. Esto permitió que en el transcurso e implementación de la herramienta fueran corregidos muchos errores antes de dar los toques finales posibilitando que se hiciera una mejor distribución del tiempo de trabajo a la misma. De esta forma, se obtiene una herramienta cuya funcionalidad ha sido comprobada con anterioridad antes de ponerse en manos de los usuarios finales, lo que garantiza la confiabilidad en la aplicación y la obtención de resultados satisfactorios durante su uso.

Conclusiones

En el desarrollo del capítulo, durante la implementación se cumplieron los principios de diseño establecidos, así como los estándares de interfaz e implementación determinados. Se presentó el Modelo de Implementación, donde fue descrita la distribución física del sistema y sus componentes. Finalmente, se realizaron pruebas obteniendo resultados satisfactorios para los clientes.

CONCLUSIONES

Con la definición del proceso anteriormente expuesto se garantiza una excelente gestión de las evaluaciones en el nivel 2 de CMMI. Con el desarrollo de la herramienta informática, implementada a partir de la necesidad de automatizar los procesos de evaluación que se llevan a cabo por los especialistas de Calisoft en la UCI, permitirá elevar la eficiencia en el proceso de revisión de los proyectos, ayudará a diagnosticar el grado de acercamiento de los proyectos productivos al programa de mejora continua en la UCI, permitiendo asegurar la calidad de los proyectos y establecer como deberán funcionar correctamente. La evaluación de variables posibilita conocer el estado real del proyecto evaluado. Para garantizar una mayor calidad se realizó entrevistas al personal capacitado, para lograr mayor conocimiento acerca del programa de mejoras, se definió un método matemático, que permite conocer el resultado y grado de avance, de la evaluación realizada. Después de haber realizado un análisis de las tecnologías, metodologías y herramientas para el desarrollo de Software, se seleccionaron las más adecuadas, permitiendo así que la propuesta de solución fuese validada satisfactoriamente por el cliente cumpliendo con el objetivo trazado.

RECOMENDACIONES

Por la importancia que brinda de la aplicación se recomienda:

- ✓ Realizar actualizaciones a la aplicación para que soporte otros niveles de CMMI.
- ✓ Incorporar un rol que se llame invitado para que cada proyecto realice evaluaciones internas.

BIBLIOGRAFÍA

1. **Peralta, M., Diez, E. y Britos, P. y García Martínez, R.** *Evaluación asistida de CMMI-SW.* Buenos Aires : s.n.
2. **Mary Beth Chrissis, Mike Konrad,Sandy Shrum.** *CMMI Guía para la integración de procesos.* 2009.
3. **Mariana Perez-Vargas, Miguel Serrano.** *Evaluación de procesos de software con SCAMPI.* 2007.
4. **Elisandra Soria Silva, Arianne Prades Roque.** *Elaboración de una aplicación que diagnostique el grado de acercamiento de los proyectos productivos al Programa de Mejora de la UCI.* Ciudad Habana : s.n., 2009.
5. **Dagmay Aveleira Quiñones, Diana Rosa Alfonso Espinosa.** *Propuesta de procedimiento para la aplicación del área de proceso de Medición y Análisis de CMMI.* Habana : s.n., 2008.
6. *Libro de Proceso para la Administración de Requisitos.* Habana : s.n., 2009.
7. *Libro de Proceso para PPQA* . Habana : s.n., 2009.
8. *Libro de Proceso para Monitoreo y Control de Proyecto* . Habana : s.n., 2009.
9. *Libro de Proceso para la Planeación del Proyecto.* 2009.
10. *Libro de Proceso para Medición y Análisis* . 2009.
11. *Libro de Proceso para la Administración de la Configuración.* 2009.
12. *Libro de Proceso para la Administración de Acuerdos con Proveedores.* 2009.
13. **Sagrario, Augusto.** *Mejora de procesos: La evolución natural.* 2007.
14. **Pressman, Roger.** *Ingeniería de Software Un enfoque práctico.*
15. **IEEE.** *Standard Glossary of Software Engineering Terminology.* 1990 .
16. **McFeeley, B.** *A User's Guide for Software Process Improvement.* Pittsburgh, Pennsylvania : s.n., 1996.
17. *CMMI for Development, v1.2.* University, Carnegie Mellon,Pittsburgh : s.n., 2006.
18. **Scalone, Fernanda.** *Estudio Comparativo de los Modelos y Estándares de Calidad del Software.* Buenos Aires : s.n., 2006.
19. **Gil, Pilar Gómez.** *Un Camino Hacia el Éxito Mundial en el Desarrollo del Software Mexicano.* 2007.
20. **I, Garro.** *Calidad de Software: Modelos y estándares.* 2005.
21. **MIGUEL, B.** *El proyecto informático de construcción de software.* . 2004.
22. **Natalia Juristo, A. M. M., Sira Vegas .** *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* (2005).
23. **Gabriel Fernández, J. T.** *Modelo de Mejora de Proceso de Software.* (2008).
24. **Almeraz, E.** *Lecciones aprendidas y estrategias de Implantación de CMMI® en México.* (2006).

REFERENCIAS

Software Engineering Institute, CMMI® for Development, Version 1.2, Software, 2006, Engineering Institute Pittsburgh, PA 15213-3890. Disponible en: http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html.

CMMI, 2002. Capability Maturity Model Integration. Disponible en http://www.sei.cmu. edu/cmmi/cmmi.html.

CMM, 1991. Capability Maturity Model for Software. Disponible en http://www.sei.cmu. du/cmm/.

CMMS, 1991. Capability Maturity Models. Disponibles en http://www.sei.cmu.edu/cmm/ cmms/cmms.html.

Chrissis, M B, Konrad, M, Shrum, S, 2003. CMMI. Guidelines for Process Integration

and Product Improvement. 688 páginas. Editorial Addison-Wesley.

ISBN 0321154967.

CMM-Quest, 2001. Self assessment tool, HM&S IT-Consulting GmbH. Demo disponible en el sitio de la empresa, http://www.cmm-quest.com/.

. IME Toolkit, 2003. Interim Maturity Evaluation Toolkit, Management Information Systems. Disponible en http://www.man-info-systems. com/IMEtoolkit.htm.

. Appraisal Wizard, 2003. Formal or informal appraisal tool, Integrated System Diagnostics Incorporated. Demo disponible en el sitio de la empresa,

http://www.isd-inc.com.

Monográfico Calidad del Software / Software de calidad. Número 137, s.l.: NOVATICA, Enero-Febrero 2004.

Pressman, R.S. Ingeniería del Software: Un enfoque práctico. 2002.

PostgreSQL Global Development Group. Postgresql.org . *Postgresql.org.* [En línea] 2009. [Citado el: 16 de Enero de 2010.] http://www.postgresql.org/.

Visual Paradigm. Visual-Paradigm.com. *Visual-Paradigm.com.* [En línea] 2009. [Citado el: 10 de Enero de 2010.] http://www.visual-paradigm.com/.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. *Lenguaje Unificado de Modelado.Manual de Referencia.* Madrid : Pearson Education S.A, 2000. ISBN.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación.S.A, 2000. ISBN.

PHP.net. Php.net. *Php.net.* [En línea] 2009. [Citado el: 15 de Enero de 2010.] hhtp://php.net.

Standish Group Report: There's Less Development Chaos Today - SD Times On The Web. Software Develoment Times (SDTimes). Disponible en: http://www.sdtimes.com/content/article.aspx?ArticleID=30247 [Accedido Mayo 27, 2009].

Software Engineering Institute, CMMI® for Development,Version 1.2, Software, 2006, Engineering Institute Pittsburgh, PA 15213-3890. Disponible en: http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html.

Krasner, H., Accumulating the Body of Evidence for The Payoff of Software Process Improvement,1997. Disponible en: http://lifelong.engr.utexas.edu/pdf/sqi/sqi.pdf.

GLOSARIO DE TÉRMINOS

APIs: (Application Program Interface). Conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos APIs comunes a los que deben ajustarse todos los desarrolladores de aplicaciones. Herramientas de programación para rutinas, protocolos y software.

ASID: Acrónimo de **A**tomicity, **C**onsistency, **I**solation and **D**urability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español. En bases de datos es un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. Así pues, si un sistema de gestión de bases de datos es *ACID compliant* quiere decir que el mismo cuenta con las funcionalidades necesarias para que sus transacciones tengan las características ACID.

Capacidad: Indica si sólo se ejecuta, o si también se planifica, se encuentra organizativa y formalmente definido, se mide y se mejora de forma sistemática.

Mejora de Procesos: Optimiza la efectividad y la eficiencia mejorando también los controles, reforzando Madurez

Modelo de procesos: Conjunto estructurado de elementos que describen características de procesos efectivos y de calidad.

Procedimiento: Es una serie de pasos, claramente definidos, que permiten trabajar correctamente y ayudan a disminuir la probabilidad de accidentes y fallos.

SaaS: Modelo de distribución del software que proporciona a los clientes el acceso al mismo a través de la red (generalmente Internet), de manera que les libra del mantenimiento de las aplicaciones, de operaciones técnicas y de soporte. Las aplicaciones distribuidas en la modalidad SaaS pueden llegar a cualquier tipo de empresa sin importar su tamaño o su ubicación geográfica. Se trata de un modelo que une el producto (software) al servicio, para dotar a las empresas de una solución completa que permita optimizar sus costes y sus recursos.

SEI: Instituto de Ingeniería de Software (Software Engineering Institute)(SEI) es un instituto federal estadounidense de investigación y desarrollo, fundado por el Congreso de los Estados Unidos en 1984

para desarrollar modelos de evaluación y mejora en el desarrollo de software, que dieran respuesta a los problemas que generaba al ejército estadounidense la programación e integración de los sub-sistemas de software en la construcción de complejos sistemas militares. Financiado por el Departamento de Defensa de los Estados Unidos y administrado por la Universidad Carnegie Mellon. Es un referente en Ingeniería de Software por realizar el desarrollo del modelo SW-CMM (1991) que ha sido el punto de arranque de todos los que han ido formando parte del modelo que ha desarrollado sobre el concepto de capacidad y madurez, hasta el actual CMMI.