Universidad de las Ciencias Informáticas
Facultad 9

Herramienta para automatizar el monitoreo y control de la implantación de CMMI.

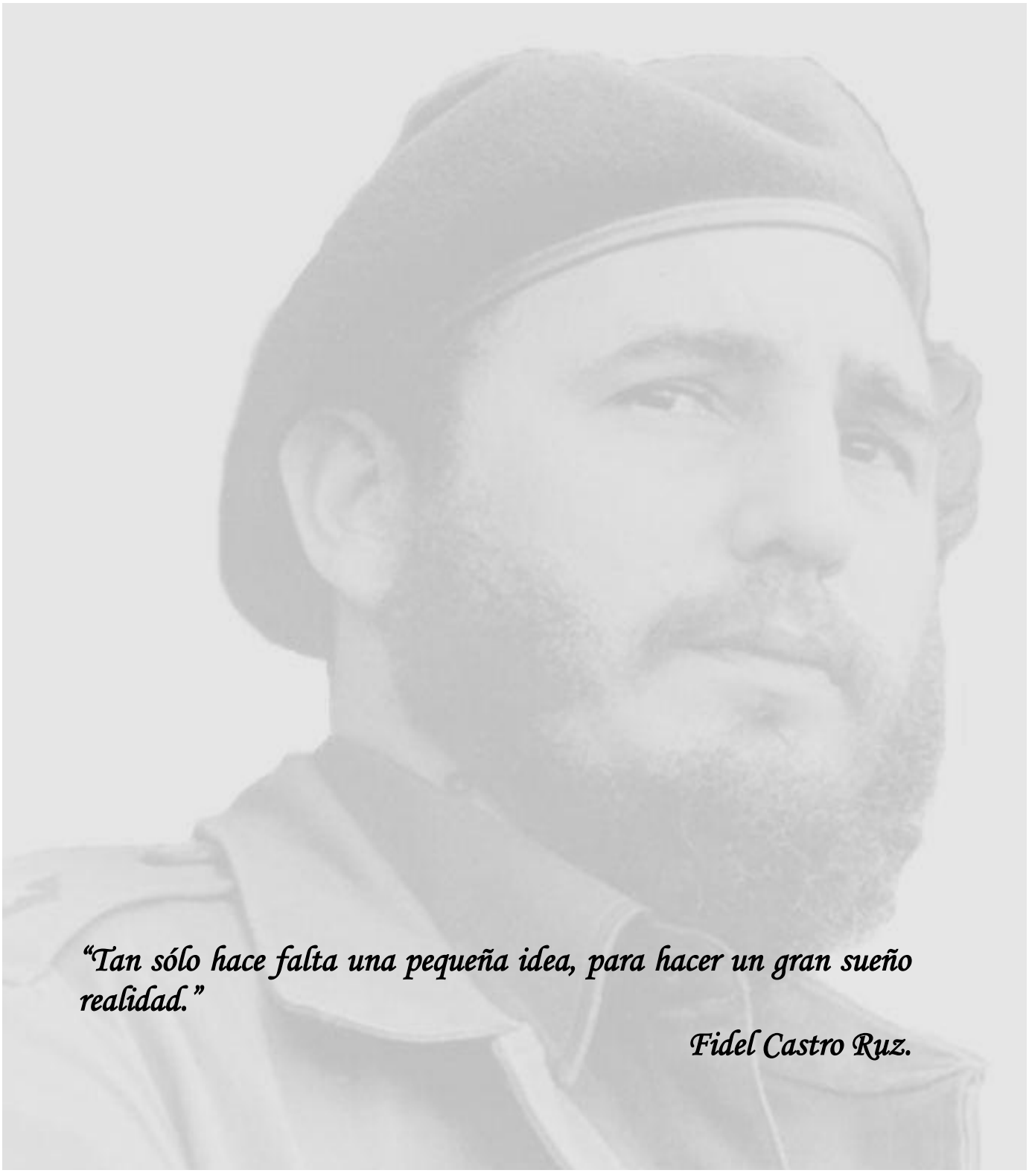
TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Ilenis Fajardo Terrero

Tutor: Ing. Nadia Porro Lugo
Cotutor: Ing. Maurice Cabrejas Martínez

Ciudad de La Habana, junio 2009

"Año 50 de la Revolución"



"Tan sólo hace falta una pequeña idea, para hacer un gran sueño realidad."

Fidel Castro Ruz.

DEDICATORIA

A mi mamá…

A mi papá Rafael...

A mi abuelita Cuchy…

A mi hermana Day…

A toda mi familia…

A mis amigos de toda la vida…

.

AGRADECIMIENTOS

A mi tutora Nadia Porro y a mi cotutor Maurice Cabrejas por estar siempre dispuesto cuando los necesitaba y darme la fuerza suficiente para seguir adelante.

A Michael Jorrín y a Lissette Rodríguez, por toda su ayuda y colaboración.

A Geovanys, por todo su amor, su paciencia y apoyo incondicional que tanto me ayudó para llegar a la meta. Por ser parte de mi vida en estos dos últimos cursos.

A mis amigos de la Caboclada y que forman parte de mi familia Nerelys, Zuzel, Noris, Hilda, Michel y Ángel.

A mis tres grandes amigas, que son como mis hermanas Mariela, Yaro y la Puchy.

A mis amigos de toda mi vida que han estado presente en el momento que lo he necesitado.

A toda mi familia por darme su apoyo y confiar en mí en todo momento.

A mi mamita Sonia , mi abuelita Cuchy, mi papá Rafael y mi hermana Day, que son mi vida y la razón de mi existir. Son las personas que se merecen todo mi esfuerzo para seguir adelante y que ninguna meta sea imposible para mí. Son los que han hecho de mí una persona de bien y a los que le debo todo en mi vida.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Ilenis Fajardo Terrero Ing. Nadia Porro Lugo

OPINIONES Y AVALES

OPINIÓN DEL TUTOR

RESUMEN

El presente trabajo surge de la necesidad de monitorear y controlar el proyecto de mejora de procesos que se lleva a cabo actualmente en la Universidad de las Ciencias Informáticas (UCI) basado en el Modelo Integrado de Madurez de la Capacidad (CMMI) como referencia. Por esta razón, se plantea como principal objetivo desarrollar una herramienta que permita de forma automatizada monitorear y controlar la implantación de dicho modelo.

Se trata de una propuesta de solución informática, que tiene como nombre “SIMPuci”, Sistema de Implantación de un Proyecto de Mejora, con el objetivo de satisfacer las funcionalidades con la que van a interactuar los usuarios que intervienen en el proceso. Para el desarrollo de la misma se tiene como referencia el modelo IDEAL (Iniciar, Diagnosticar, Establecer, Actuar y Aprender) como método oficial de implantación de CMMI y a COBIT (Objetivos de Control para la Información y la Tecnología Relacionada) como modelo de referencia para el control.

Para lograr un mejor entendimiento de la investigación se exponen una serie de conceptos asociados al dominio del problema en cuestión. Además se realiza un estudio del arte y se seleccionan las tecnologías y herramientas con las cuales se construye la solución propuesta. El sistema que se propone se detalla de una manera simple con todos los artefactos generados de la metodología XP y posteriormente se implementan en su totalidad las funcionalidades requeridas con las tecnologías de punta seleccionadas.

Palabras Claves: CMMI, COBIT, control, IDEAL, mejora, monitoreo, procesos.

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURA

Introducción

En la época actual, el éxito de las organizaciones está directamente vinculado con la efectividad del uso de la Tecnología. En este marco, el desarrollo de software es un proceso clave, ya sea ejecutado interna o externamente, con impacto directo en la calidad y disponibilidad de los productos y servicios de las entidades que se dedican a la producción.

El desarrollo y evolución de software se ve permanentemente impactado, por un lado, por la creciente cantidad de requerimientos y, por el otro, por los cambios y variedad de aplicaciones disponibles, ambientes operativos, plataformas, infraestructura de redes, ambientes de desarrollo y lenguajes de programación.

Cada vez más se requiere que el desarrollo y evolución de software logre productos de mayor calidad y en menor tiempo respondiendo a los requerimientos crecientes en ambientes tecnológicos cambiantes, con procesos bien definidos, predecibles y en condiciones de ser permanentemente mejorados.

Por estas razones, muchas organizaciones han decidido invertir en mejorar sus procesos de desarrollo y evolución de software, obteniendo como resultado inmediato un excelente retorno al lograr mayor calidad en sus productos y disminuir sus tiempos y costos de desarrollo.

La situación actual y las perspectivas de la industria cubana de software están caracterizadas por el trabajo que se viene realizando en materia de capacitación y en la inserción tanto en el mercado nacional como internacional.

Para lograr estos objetivos fue creada en el año 2002 la Universidad de las Ciencias Informáticas como parte de la batalla de ideas cuya misión es formar profesionales comprometidos con la Revolución y muy preparados en Ciencias Informáticas. Con su surgimiento, se ha ganado una infraestructura productiva que combina el estudio con la producción de software, donde se desarrollan múltiples proyectos productivos destinados a la informatización de la sociedad cubana, y la comercialización de software en el exterior.

INTRODUCCIÓN

La Universidad tiene una estructura docente productiva organizada en diez facultades, con un segundo perfil productivo definido; cada facultad tiene su proyecto de control de calidad que cuenta con sus propios métodos y estrategias de trabajo acorde a las características de la facultad, siempre regidos y alineados por los objetivos de la Dirección de Calidad de Software de la UCI, siendo este el centro de referencia de calidad que garantiza el crecimiento continuo de una producción de software con calidad en la organización; a través de la definición de procesos siguiendo las especificaciones de metodologías, estándares y modelos de desarrollo de software.

Situación Problemática

En la actualidad el centro está acometiendo un proyecto de mejora de sus procesos basado en el modelo CMMI bajo los servicios del SIE Center (Centro Excelente de Industria de Software) del Tecnológico de Monterrey. El proceso de mejora está encaminado a que la universidad alcance en el 2010 una certificación internacional del nivel 2 del modelo CMMI. Para esto se hace uso de su modelo oficial de implantación IDEAL, quien guíe la planificación e implementación de las mejores prácticas de este modelo.

Para proveer que se asimile este proyecto de mejora de procesos en los proyectos productivos UCI, es necesario un exhaustivo seguimiento y control al progreso del mismo, para asegurar que su organización está basada en procesos y con un programa de mejora continua alineado con sus objetivos de negocio, debido a que no está establecido cómo monitorear y controlar en los proyectos el estado de este proceso, esto puede traer consigo que se tengan falsas nociones del avance del mismo , así como que no se identifiquen y detecten a tiempo los problemas existentes.

La polémica antes expuesta es consecuencia de algunas deficiencias señaladas por algunos Especialistas del Grupo de Mejora de Procesos perteneciente a la Dirección de Calidad de Software en entrevistas efectuadas con el objetivo de conocer la situación actual del proyecto de mejora de procesos, basado en el modelo CMMI.

Según el resultado de las entrevistas, se determinó que debido a que es la primera vez que se realiza este proceso y no hay experiencia del mismo, el personal destinado a esta tarea no se encuentra lo

suficientemente capacitado, y pueden existir fallas durante la ejecución. Por lo que es necesario darle un seguimiento y control a esta actividad.

Precisamente el monitoreo y control de proyectos consiste en controlar cada una de las actividades de un proyecto durante un tiempo determinado. Es el procedimiento por el cual se verifica la eficiencia y eficacia de la ejecución de un proyecto mediante la identificación de sus logros y debilidades, en consecuencia, recomienda medidas correctivas para optimizar los resultados del proyecto.

Teniendo en cuenta lo antes expuesto se plantea como **problema científico**: Inadecuado monitoreo y control de la implantación de las buenas prácticas de CMMI en los proyectos productivos de la UCI.

Se define el **objeto de estudio** como: Proceso de monitoreo y control de la implantación de CMMI en el proceso de desarrollo software.

El **campo de acción** está determinado por: Herramientas de monitoreo y control para la implantación de las buenas prácticas de CMMI en los proyectos productivos UCI.

Este trabajo tiene como **objetivo general**: Desarrollar una herramienta que permita realizar el monitoreo y control de la implantación de CMMI de forma automatizada.

Para cumplir el objetivo general se trazaron los siguientes **objetivos específicos**:

- Determinar las actividades del proceso de monitoreo y control que puedan ser automatizadas.
- Controlar el desempeño de las actividades definidas.
- Desarrollar el análisis y diseño de la solución propuesta.
- Implementar una herramienta para la automatización del proceso de monitoreo y control en la implantación de CMMI.

Para dar cumplimiento de forma exitosa a los objetivos planteados, se definen las siguientes **tareas**:

- Establecer un diagnóstico de las tendencias actuales en el proceso de monitoreo y control de la implantación de CMMI en los proyectos de la UCI y tomar posición al respecto.
- Caracterizar las distintas propuestas y herramientas existentes en el mercado para el proceso de monitoreo y control de la implantación de CMMI.

- Valorar las bases teóricas que sustenta el proceso de monitoreo y control de la implantación de CMMI.
- Establecer los métodos y procedimientos más factibles para la automatización del proceso de monitoreo y control.
- Analizar las metodologías de desarrollo de software más utilizadas y definir cuál es la más apropiada para el desarrollo de la solución propuesta.
- Analizar y definir las herramientas y tecnologías a utilizar en el desarrollo de la solución.
- Definir las actividades a automatizar basándose en modelo IDEAL, como método de implantación de CMMI.
- Documentar los artefactos que son generados durante la presentación y la construcción de la solución propuesta.
- Implementar la herramienta.

Como principal **aporte práctico** se obtendrá una herramienta que automatice el proceso de monitoreo y control de la implantación de CMMI.

El desarrollo de un sistema de monitoreo y control de la implantación de CMMI, posibilitará la correcta realización de este proceso, lo que constituye la **idea a defender** de este trabajo.

Entre los **posibles resultados** se tienen:

- Caracterización del proceso de monitoreo y control de la implantación de CMMI en los proyectos de la UCI.
- Obtención de la documentación generada durante la planificación y construcción de la aplicación.
- Herramienta de monitoreo y control para la implantación de CMMI en los proyectos productivos UCI.

La utilización de una aplicación permitirá un mejor control del proceso y una mayor eficiencia en el desarrollo de las actividades del mismo, pues posibilitaría alertar, monitorear y corregir cada una de las tareas y disminuiría el esfuerzo realizado por los trabajadores en la construcción de la documentación y artefactos que en sentido general son generados durante el flujo de las actividades de monitoreo y control.

INTRODUCCIÓN

En el proceso de desarrollo de la investigación se emplean **métodos científicos** tanto teóricos como empíricos, los primeros posibilitan ir más allá de las particularidades fenomenológicas y triviales de la situación en análisis, además de la situación existente del fenómeno en una etapa determinada; y los segundos posibilitan extraer de los fenómenos examinados las informaciones que se requieren sobre ellos a través de observaciones**.**

Como **métodos teóricos** se utiliza: *Análisis Histórico-Lógico* mediante el cual se desarrolla la investigación centrándose en las tendencias actuales de la mejora de procesos y sistemas enfocados a la implementación de esta mejora , aprovechando de estos algunos elementos teóricos que aportarán para el desarrollo de la aplicación. Por otra parte se emplea método *Analítico - Sintético* con el fin de extraer lo esencial de la bibliografía consultada, realizando una división mental del objeto de estudio en sus múltiples relaciones y componentes y comprender sus características generales tras sintetizar estas como un todo. Y además de los antes mencionados se utiliza el método *Inductivo – Deductivo,* el cual facilita el análisis de los elementos generales a elementos más particulares, o sea, permite deducir cual es la mejor estrategia para el monitoreo y control de la implantación del proyecto de mejora de procesos basado en CMMI, partiendo del resultado de otros estudios e investigaciones.

Dentro de los **métodos empíricos** se emplea la *entrevista,* con el objetivo de obtener información acerca de la problemática existente, así como de las características y peculiaridades de la implantación de un proyecto de mejora en la universidad. Para ello se utiliza como técnica de muestreo no probabilística, el muestreo intencional que permite elegir explícitamente los elementos que son representativos o con posibilidades de brindar mayor información. Se define como población los 8 integrantes del Grupo de Mejora de Procesos, el cual lo constituye el Grupo de Normalización y el Grupo de Métricas perteneciente a la Dirección de Calidad de Software de la UCI, siendo la unidad de estudio los Especialistas perteneciente a éste Grupo de Mejora, y se toma como muestra 3 integrantes del mismo, lo que representa un 37.5%. La muestra se seleccionó teniendo en cuenta que la población seleccionada es homogénea y los integrantes pertenecientes a éste grupo son los que están más familiarizados con el tema de mejora de procesos, por lo que las entrevistas pudo arrojar resultados representativos.

La investigación está estructurada por: Introducción, 3 capítulos, conclusiones, recomendaciones, bibliografías, anexos y glosario.

INTRODUCCIÓN

En el **Capítulo 1 "Fundamentación Teórica"**: se hace un análisis crítico de la bibliografía consultada, realizando un estudio de algunos de los modelos de calidad existentes, así como los principales conceptos asociados al dominio del problema.

En el **Capítulo 2 "Tendencias y tecnologías actuales a desarrollar"**: se hace un análisis y estudio de las principales tecnologías actuales que se adecuan a la solución propuesta.

En el **Capítulo 3 "Solución propuesta"**: se realiza una descripción de la propuesta de solución, definiendo los artefactos de la metodología de desarrollo a utilizar. Para ello se definen los requerimientos que debe cumplir la herramienta en base a historias de usuario y se lleva a cabo un análisis de los mismos describiendo los planes de entrega y las tareas de ingeniería. Además se describe el diseño y se muestra la interfaz gráfica.

CAPÍTULO 1: Fundamentación Teórica

1.1 Introducción

Este capítulo tiene como objetivo fundamental, abordar los aspectos que se utilizan como fundamento teórico que sustenta el problema de esta investigación. Se hace un análisis de algunos de los modelos de calidad, haciendo énfasis en el modelo CMMI como modelo escogido y su modelo oficial IDEAL para la implantación del mismo. De igual manera se describe las características que presenta COBIT como base para la solución propuesta, en monitorear y controlar la implantación de CMMI.

1.2 Conceptos asociados al dominio del problema

1.2.1 Calidad

Calidad es un concepto manejado con bastante frecuencia en la actualidad, pero a su vez, su significado es percibido de distintas maneras. Al hablar de bienes y/o servicios de calidad, las personas se refieren normalmente a bienes de lujo o excelentes con precios elevados. Su significado sigue siendo ambiguo y muchas veces su uso depende de lo que cada uno entiende por calidad, por lo cual es importante comenzar a unificar su definición.

Diversos autores han dado su definición de calidad de manera diferente, por ejemplo:

El Dr. Joseph M. Juran, centró su estudio en dos significados críticos:

“Calidad: Se refiere a la ausencia de deficiencias que adopta la forma de: Retraso en las entregas, fallos durante los servicios, facturas incorrectas, cancelación de contratos de ventas, etc.” (JURAN, 1990)

Calidad es “adecuación al uso”. (JURAN, 1990)

David Hoyle la define como “un grado de excelencia, la conformidad con los requerimientos, la totalidad de funciones del producto o servicio que satisfacen las necesidades especificadas, la actitud para el uso, la ausencia de defectos, imperfecciones o contaminación y el deleite de los clientes”. (HOYLE, 1998)

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

Pressman define calidad como:“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”. (PRESSMAN, 2002)

Organizaciones reconocidas también han ofrecido su definición de la calidad:

De acuerdo a la definición del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) calidad es: “el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”. (IEEE, 1990)

Según la Real Academia Española calidad es: “propiedad o conjunto de propiedades inherentes a una cosa que permiten apreciarla como igual, mejor o peor que las restantes de su especie”. (REA, 2003)

El concepto actual de calidad ha evolucionado hasta convertirse en una forma de gestión que introduce el concepto de mejora continua en cualquier organización y a todos los niveles de la misma, y que afecta a todas las personas y a todos los procesos. Con lo anterior se puede concluir que la calidad se define como:"Un proceso de mejoramiento continuo, en donde todas las áreas de la empresa participan activamente en el desarrollo de productos y servicios, que satisfagan las necesidades del cliente, logrando con ello mayor productividad".

1.2.2 Calidad de Software

Al decir que se va a producir un software con calidad primero es necesario conocer las características del software como producto, y las implicaciones que se desprenden de la manera particular en que es desarrollado. Al final del proceso de desarrollo de software lo que se obtiene es un producto que a diferencia de la mayoría de los productos conocidos “no se gasta con el uso y repararlo no significa una restauración a su estado original sino corregir defectos que estaban desde el momento de su entrega y que deben ser solucionados en la etapa de mantenimiento”.(PIATTINI, 2005)

“Un proceso de poca calidad generalmente producirá un producto de poca calidad “.(HUMPHREY, 1995)

Por tanto para mejorar la calidad en el producto, se necesitan medir las características y los parámetros de la calidad del proceso.

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

La Organización Internacional de Estándares (ISO/IEC-9126) y la Comisión Internacional Electrotécnica (IEC) definen calidad de software como: “la totalidad de características de un producto de software que le confiere la capacidad de satisfacer necesidades explícitas e implícitas”. (ISO/IEC-9126, 1991)

La calidad del software es definida por el cumplimiento con los requisitos para los que fue desarrollado aunque no se puede dejar de tener en cuenta que el objetivo principal al luchar por la calidad es lograr la satisfacción del cliente. Se puede concluir que la obtención de un producto de software con calidad implica mantener relacionados enfoques de proceso, producto y efecto, utilizando modelos y estándares durante su desarrollo y garantizar la completa satisfacción del cliente.

1.2.3 Monitoreo y control de proyectos

La palabra monitoreo se asocia con la acción de controlar o supervisar cuidadosamente una actividad durante un tiempo acordado. Según algunos autores, el término monitoreo se define como:

“Proceso continuo de recolección y análisis de datos cualitativos y cuantitativos, con base en los objetivos planteados en un programa o proyecto, que tiene como propósito descubrir fortalezas y/o debilidades para establecer líneas de acción, permitiendo brindar correcciones y reorientaciones técnicas en la ejecución”. (PAREDES, 2003)

“Toda actividad de gestión aseguradora de que el trabajo real va de acuerdo al plan: compara lo realizado con las metas y planes, revela cuando y donde existen desviaciones, y pone en marcha acciones para correctoras, ayudando a la realización de los planes”. (THAYER, 1988)

“Proceso de hacer que las cosas ocurran de forma ordenada o de acuerdo a lo planificado”. (FAIRLEY, 1994)

El control se concibe como la verificación a posteriori de los resultados conseguidos en el seguimiento de los objetivos planteados, donde la estandarización en términos cuantitativos, forma parte central de la acción de control.

El control es una etapa primordial en la administración, pues, aunque una empresa cuente con magníficos planes, una estructura organizacional adecuada y una dirección eficiente, el ejecutivo no podrá verificar

cuál es la situación real de la organización si no existe un mecanismo que se cerciore e informe si los hechos van de acuerdo con los objetivos.

El concepto de control es muy general y puede ser utilizado en el contexto organizacional para evaluar el desempeño general frente a un plan estratégico. Según los planteamientos de varios autores estudiosos del tema, se define el término control como:

“El control consiste en verificar si todo ocurre de conformidad con el plan adoptado, con las instrucciones emitidas y con los principios establecidos. Tiene como fin señalar las debilidades y errores a fin de rectificarlos e impedir que se produzcan nuevamente”. (FAROL, 1961)

“El control es una función administrativa: es la fase del proceso administrativo que mide y evalúa el desempeño y toma la acción correctiva cuando se necesita. De este modo, el control es un proceso esencialmente regulador”. (CHIAVENATTO, 2000)

En resumen, de acuerdo a las definiciones dadas anteriormente por diferentes autores referente al término monitoreo y control respectivamente, se define monitoreo y control como el proceso administrativo que verifica, mide y evalúa el desempeño de las metas ya planificadas , y pone en marcha acciones correctivas para cuando estas se desvíen del plan .

El monitoreo y control genera beneficios, tales como:

- Proporciona información acerca de la situación de la ejecución de los planes, sirviendo como fundamento al reiniciarse el proceso de planeación.
- Determina y analiza rápidamente las causas que pueden originar desviaciones, para que no se vuelvan a presentar en el futuro.
- Examina continuamente los objetivos del proyecto, determinando así el riesgo de no cumplir con ellos.
- Identifica problemas recurrentes que necesitan atención y ayuda a identificar soluciones de problemas.
- Al evitar errores reduce costos y ahorra tiempo.

El monitoreo y control no solo ayuda a supervisar que se realice el trabajo planificado en el tiempo fijado, sino también a que el personal del proyecto sepa cómo van las cosas y así poder detectar a tiempo posibles dificultades o problemas. Este control garantiza el cumplimiento de lo planificado, y a su vez puede ser usado como documentación para el perfeccionamiento de éste.

1.2.4 Medición en el proceso de monitoreo y control.

La medición es la base de la ingeniería, la ciencia y los negocios. La ingeniería del software ha comenzado con el establecimiento de métricas estándar para el desarrollo de software. La medición de software consiste en la colección significativa y precisa de información que tiene valor práctico para el personal. El objetivo es proporcionar a los profesionales y administradores un conjunto de datos útiles y tangibles para dimensionar, estimar y controlar proyectos de software con rigor y precisión. Entonces, se puede definir como métricas de software o medidas de software a:

"La continua aplicación de técnicas basadas en la medición al proceso de desarrollo de software y a sus productos para proveer información administrativa significativa y oportuna, junto con el uso de esas técnicas para mejorar el proceso y sus productos." (WESTFALL, 1995)

La medición tiene los siguientes fines:

- **Informar**: Transmitir y comunicar la información necesaria para la toma de decisiones.
- **Coordinar**: Encaminar todas las actividades eficazmente a la consecución de los objetivos.
- **Evaluar**: La consecución de las metas (objetivos) se logra gracias a las personas, y su valoración es la que pone de manifiesto la satisfacción del logro.
- **Motivar**: El impulso y la ayuda a todo responsable es de vital importancia para la consecución de los objetivos.

(HORNA, 2004)

Todas las actividades pueden medirse con parámetros que enfocados a la toma de decisiones, son señales para monitorear la gestión. De esta forma se asegura que las actividades vayan en el sentido

correcto y permiten evaluar los resultados de una gestión frente a sus objetivos, metas y responsabilidades.

1.2.4.1 Métricas a aplicar en el proceso de monitoreo y control

El uso de métricas proporciona un dato cuantitativo y cualitativo sobre el proceso de monitoreo y control de proyectos, dando estimaciones sobre el estado de las diferentes actividades y del proceso en general. Da la posibilidad de realizar estudios comparativos entre diferentes proyectos y muestra el estado de la mejora de procesos en la institución. Para dar solución al problema planteado, se apoya en seis de los objetivos de control de importancia alta del modelo COBIT, los cuales los primeros cuatros mencionados a continuación se les dará cumplimiento de forma implícita en la solución propuesta y a los otros dos restantes mediante la definición de un conjunto de métricas con el propósito de evaluar el proceso implementado.

- **PO1: Definir un plan estratégico para TI**

A este objetivo se le da cumplimiento mediante la planificación de las actividades propuestas por el modelo IDEAL, a la hora de haberse creado un proyecto o iniciarse un nuevo ciclo.

- **PO10 Administrar Proyectos**

Este objetivo se enfoca en la administración de proyectos definidos, lo cual facilita la participación de los interesados y el monitoreo de los riesgos y los avances de los proyectos, lográndose un control administrativo en la entrega de resultados dentro de marcos de tiempo, presupuesto y calidad acordados.

- **DS 5:Garantizar la seguridad de los sistemas**

Este objetivo se le da cumplimiento mediante la gestión de cuentas de usuarios, de modo que se garantiza la integridad de la información, y la protección de los activos, logrando minimizar el impacto de las vulnerabilidades e incidentes de seguridad.

- **DS11: Administrar los datos**

Mediante el cumplimiento de este objetivo, se permite la administración de artefactos manteniendo la exactitud, disponibilidad y protección de los mismos, así como también de la información persistente que reside en la base de datos.

- **ME1: Monitorear y evaluar el desempeño de TI**

1. **Métrica**: Eficiencia en el cumplimiento de las actividades

Nombre	Eficiencia en el cumplimiento de las actividades
Definición	Esta métrica se calcula : ECA = NAT/NAC Donde: ECA: Eficiencia en el cumplimiento de las actividades. NAT: Número de actividades terminadas. NAC: Número de actividades creadas
Metas	▪ Alcanzar un valor de 1 o cercano a este. ▪ Evaluar el avance del proyecto de mejora. ▪ Indicar la calidad del proyecto.
Procedimiento de análisis	Los posibles valores de esta métrica oscilan entre [0,1].Donde los valores más cercanos a 1 reflejan una mayor eficiencia en el cumplimiento de las actividades. En caso de que sea menor que uno deben tomarse las acciones correctivas necesarias.
Responsable	Jefe del Proyecto

2. **Métricas:** Por ciento de actividades retrasadas del proyecto

Nombre	Por ciento de actividades retrasadas del proyecto
Definición	Esta métrica se calcula:

	(NAR /NAC)*100 Donde: NAR: Número de actividades retrasadas. NAC: Número de actividades creadas.
Metas	Entre los rangos de : 80 y 100% → Crítico 60 y 79% → Mal 40 y 59% → Regular 20 y 39% → Bien 0 y 19% → Muy Bien
Procedimiento de análisis	En dependencia del resultado que arroje la métrica se tomarán las medidas pertinentes para mejorar los resultados. Sé buscará que la meta esté entre bien o muy bien, siempre tomando medidas para disminuir el valor cuantitativo.
Responsable	Jefe del Proyecto

3. **Métrica :** Magnitud de retraso

Nombre	**Magnitud de retraso**
Definición	Esta métrica se calcula: MR = Tp – Tr MR: Magnitud de retraso. Tp: Tiempo planificado. Tr: Tiempo real.
Metas	Si la diferencia entre el tiempo planificado y el tiempo real de las actividades del proyecto es: (+) → Existe una buena estimación. (-) → Hay una mala estimación.
Procedimiento de análisis	Esta medida se utiliza para seguir el tiempo de duración real de

	las actividades y tener una idea de cuán atrasado está el desarrollo del proyecto. Es el tiempo real de duración de las actividades con respecto al tiempo planificado.
Responsable	Jefe del Proyecto

- **PO 9: Evaluar y administrar los riesgos de TI**

4. **Métrica:** Número de riesgos identificados

Nombre	**Número de riesgos identificados**
Definición	Cantidad de riesgos identificados
Metas	▪ Determinar qué riesgos tienen probabilidad de afectar el proyecto y documentar las características de cada uno.
Procedimiento de análisis	Priorizar los riesgos identificados y hacer el plan de mitigación de los mismos.
Responsable	Jefe del Proyecto
Objetivo	Evaluar y administrar los riesgos de TI

5. **Métrica:** Situación de riesgos

Nombre	**Situación de riesgos**
Definición	Esta métrica se determina: Trc / Tr ≥ Trs/ Tr > Trr / Tr (Crítica) Trc / Tr >Trs/ Tr ≥ Trr / Tr Trc / Tr < Trs/ Tr > Trr / Tr (Menos Crítica) Trc / Tr < Trs/ Tr ≥ Trr / Tr

	Trc / Tr ≤ Trs/ Tr < Trr / Tr (A valorar) Trc / Tr < Trs/ Tr < Trr / Tr Donde: Tr: Total de riesgos identificados Trc: Total de riesgos críticos Trs :Total de riesgos significativos Trr : Total de riesgos rutinarios
Metas	▪ Situación Crítica ▪ Situación Menos Crítica ▪ Situación a Seguir
Procedimiento de análisis	Una vez clasificados los riesgos en críticos, significativos y rutinarios, sé comprobará cuan crítica resulta la situación actual de los riesgos, de forma general
Responsable	Jefe del Proyecto

6. **Métrica:** Por ciento de riesgos críticos identificados

Nombre	**Por ciento de riesgos críticos identificados**
Definición	Esta métrica se calcula: (TRPC/TRI)*100 Donde: TRPC: Total de riesgos con prioridad crítica. TRI: Total de riesgos identificados.
Metas	Entre los rangos de: 80 y 100% → Crítico 60 y 79% → Mal 40 y 59% → Regular 0 y 19% → Muy Bien

Procedimiento de análisis	En dependencia del resultado que arroje la métrica se tomarán las medidas pertinentes para mejorar los resultados. Sé buscará que la meta esté entre bien o muy bien, siempre tomando medidas para disminuir el valor cuantitativo.
Responsable	Jefe del Proyecto

7. **Métrica:** Efectividad de la mitigación de los riesgos críticos

Nombre	**Efectividad de la mitigación de los riesgos críticos**
Definición	Esta métrica se calcula: EMR = RM/TRC Donde: EMRC: Efectividad de la mitigación de los riesgos críticos. RM: Riesgos mitigados. TRC: Total de riesgos críticos.
Metas	A medida que la relación entre los riesgos mitigados y el total de riesgos críticos se acerque a uno, será más efectiva la mitigación de los riesgos críticos.
Procedimiento de análisis	Los posibles valores de esta métrica oscila entre [0,1].Donde los valores más cercanos a 1 reflejan una mayor eficiencia en la mitigación de los riesgos críticos. En caso de que sea menor que uno deben tomarse las acciones correctivas necesarias, ya que según la prioridad de estos riesgos serían los primeros en mitigarse.
Responsable	Jefe del Proyecto

8. **Métrica:** Efectividad de la mitigación de los riesgos

Nombre	Efectividad de la mitigación de los riesgos
Definición	Esta métrica se calcula: EMR = RM/TRI Donde: EMR: Efectividad de la mitigación de los riesgos. RM: Riesgos mitigados. TRI: Total de riesgos identificados.
Metas	A medida que la relación entre los riesgos mitigados y el total de riesgos identificados se acerque a uno, será más efectiva la mitigación de los riesgos.
Procedimiento de análisis	Los posibles valores de esta métrica oscila entre [0,1].Donde los valores más cercanos a 1 reflejan una mayor eficiencia en la mitigación de los riesgos identificados. En caso de que sea menor que uno deben tomarse las acciones correctivas necesarias.
Responsable	Jefe del Proyecto

1.2.4.2 Métricas Primitivas

1. Número de actividades terminadas

Nombre	Número de actividades terminadas
Definición	De las actividades creadas, las actividades terminadas son aquellas que se le da cumplimiento dentro de la fecha planificada.
Procedimiento de recolección	Las actividades a medida que se van completando se va aumentando el porcentaje de culminación hasta que se complete la misma, y cambia su estado a “Terminada” y se almacena en el historial como una tarea completada.
Responsabilidades	Jefe del Proyecto

2. Número de actividades creadas

Nombre	Número de actividades creadas
Definición	Estas son las actividades que se le asignan a una persona o equipo para ser cumplida en un determinado tiempo.
Procedimiento de recolección	Estas actividades son creadas automáticamente en el proyecto, y luego son asignadas, mediante el jefe de proyecto a un usuario determinado, la que a su vez se registran en el historial como una tarea creada.
Responsabilidades	Jefe del Proyecto

3. Número de actividades retrasadas

Nombre	Número de actividades retrasadas
Definición	Estas son las actividades que tienen una fecha de cumplimiento real mayor que la fecha tope de culminación de la misma.
Procedimiento de recolección	Estas actividades luego que pase de la fecha planificada de culminación, se estiman como una actividad retrasada, cambia su estado y se guarda en el historial como una actividad retrasada.
Responsabilidades	Jefe del Proyecto

4. Tiempo planificado de la actividad

Nombre	Tiempo planificado de la actividad
Definición	Rango de tiempo en el que una actividad debe ser cumplida.
Procedimiento de recolección	Este rango de tiempo es verificado por el jefe de proyecto, ya que es el encargado de asignar una actividad determinada.
Responsabilidades	Jefe del Proyecto

5. Tiempo real de la actividad

Nombre	Tiempo real de la actividad
Definición	Rango de tiempo en el que la tarea es realmente cumplida.
Procedimiento de recolección	Este rango de tiempo es verificado por el jefe de proyecto, ya que es el encargado de asignar una actividad determinada.
Responsabilidades	Jefe del Proyecto

6. Total de riesgos identificados

Nombre	Total de riesgos identificados
Definición	Estos son los riesgos identificados a lo largo de la ejecución del proyecto, ya sea durante un ciclo, fase o la realización de alguna actividad, por cualquier persona o individuo involucrado en el mismo
Procedimiento de recolección	Estos riesgos son introducidos por el propio personal involucrado en el proyecto, los cuales quedarán registrados en una plantilla, con sus correspondientes características.
Responsabilidades	Jefe del Proyecto

7. Total de riesgos críticos

Nombre	Total de riesgos críticos
Definición	Estos riesgos son introducidos y clasificados según su prioridad como críticos por el propio personal involucrado en el proyecto.
Procedimiento de recolección	Estos son los riesgos con una prioridad crítica, los cuales estarán registrados en una plantilla con su respectiva clasificación.
Responsabilidades	Jefe del Proyecto

8. Total de riesgos significativos

Nombre	Total de riesgos significativos
Definición	Estos riesgos son introducidos y clasificados según su prioridad como significativos por el propio personal involucrado en el proyecto.
Procedimiento de recolección	Estos son los riesgos con una prioridad significativa, los cuales estarán registrados en una plantilla con su respectiva clasificación.
Responsabilidades	Jefe del Proyecto

9. Total de riesgos rutinarios

Nombre	Total de riesgos rutinarios
Definición	Estos riesgos son introducidos y clasificados según su prioridad como rutinarios por el propio personal involucrado en el proyecto.
Procedimiento de recolección	Estos son los riesgos con una prioridad rutinaria, los cuales estarán registrados en una plantilla con su respectiva clasificación.
Responsabilidades	Jefe del Proyecto

10. Total de riesgos mitigados

Nombre	Número de actividades retrasadas
Definición	Estos son los riesgos identificados a lo largo de la ejecución del proyecto, ya sea durante un ciclo, fase o la realización de alguna actividad, por cualquier persona o individuo involucrado en el mismo que ya hayan sido mitigados.

Procedimiento de recolección	Estos riesgos son aquellos que según su respuesta y el período de mitigación que le fue asignado, hayan sido mitigados y estén registrados como riesgos mitigados en la planilla de riesgos.
Responsabilidades	Jefe del Proyecto

Las métricas definidas proporcionan medidas e información sobre la forma que el personal desarrolla las actividades. Estas fueron concebidas para medir un trabajo de equipo que sigue una definición de procesos consistentes, pudiendo coordinar mejor el trabajo individual de sus miembros y dar continuidad a su proceso. Los procesos definidos identifican y simplifican tareas de rutina y ayudan a pensar con mayor precisión en el trabajo a realizar. Una vez que los procesos están definidos y medidos se pueden cambiar y mejorar. Con la idea de cuantificar el comportamiento de los procesos y caracterizarlos completamente se puede considerar que se han definido métricas del Proceso.

Ofrecen una medida de cuan efectivos han sido los resultados obtenidos, luego de aplicar tareas, actividades y procesos como son las estrategias, el plan de contingencia y el plan de mitigación. Los valores obtenidos al aplicar las métricas son registrados y comparados cada vez que se repita el proceso, brindando una medida de comparación.

1.2.5 Proceso de desarrollo de software

El proceso de desarrollo de software es un concepto más amplio, basado en el de ciclo de vida, y cubre todos los elementos necesarios (tecnologías, personal, artefactos, procedimientos, etc.) relacionados con las actividades involucradas en la vida de un producto de software**.** Define como se organiza, gestiona, mide, soporta y mejora el desarrollo, independientemente de las técnicas y métodos usados.

Según Jacobson “un proceso define **quién** está haciendo **qué**, **cuándo** y **cómo** para alcanzar un determinado objetivo. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad. Captura y presenta las mejores prácticas que el estado actual de la tecnología permite. En consecuencia, reduce el riesgo y hace el proyecto más predecible. (JACOBSON, 2000)

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

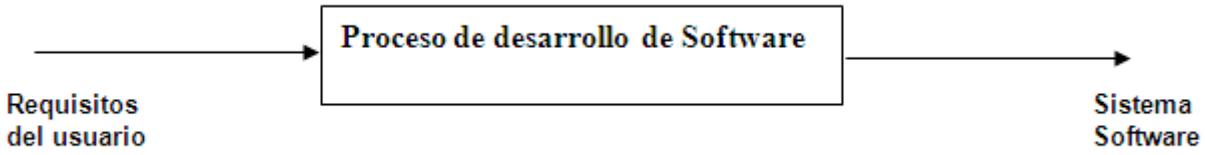


Figura 1: Proceso de desarrollo de software según Jacobson

(JACOBSON, 2000)

Pressman caracterizando el proceso de desarrollo de software plantea que" se establece un marco común del proceso definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos del software, con independencia de su trabajo o complejidad. Un número de conjuntos de tareas que permiten que las actividades del marco de trabajo se adapten a las características del proyecto del software y a los requisitos del equipo del proyecto. Finalmente las actividades de protección abarcan el modelo de procesos. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso." (PRESSMAN, 2002)

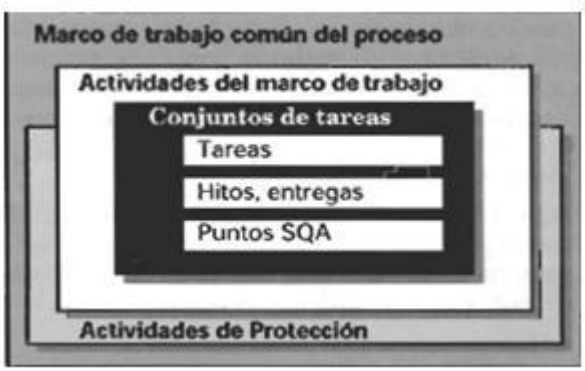


Figura 2: Proceso de desarrollo de software según Pressman

(PRESSMAN, 2002)

Después de analizar los diferentes conceptos, se puede afirmar que un proceso de desarrollo de software es un grupo de actividades llevadas a cabo para desarrollar y gestionar sistemas de software, estas guían los esfuerzos de las personas implicadas en el proyecto; cada proceso tiene clientes y otras partes interesadas que son afectados por el proceso y quienes definen los resultados requeridos de acuerdo con

sus necesidades y expectativas. Todos los procesos deberían estar alineados con los objetivos de la organización y diseñados para aportar valor, teniendo en cuenta el alcance y la complejidad de la organización.

1.2.6 Mejora de Procesos de Software

La mayoría de las organizaciones han tomado conciencia de que tienen que poner en su punto de mira a los procesos de la empresa y se plantean cómo mejorarlos y evitar algunos males habituales como: poco enfoque al cliente, bajo rendimiento de los procesos, barreras departamentales, subprocesos inútiles debido a la falta de visión global del proceso, excesivas inspecciones, reproceso, y otros.

Según Moen "el mejoramiento de un proceso es el esfuerzo continuo para saber acerca del sistema de causas en un proceso y para usar este conocimiento en el cambio y mejora del proceso y de esa manera reducir su variación, complejidad y mejorar la satisfacción del cliente." (MOEN, 2001)

La siguiente figura muestra la importancia del mejoramiento de los procesos de software.

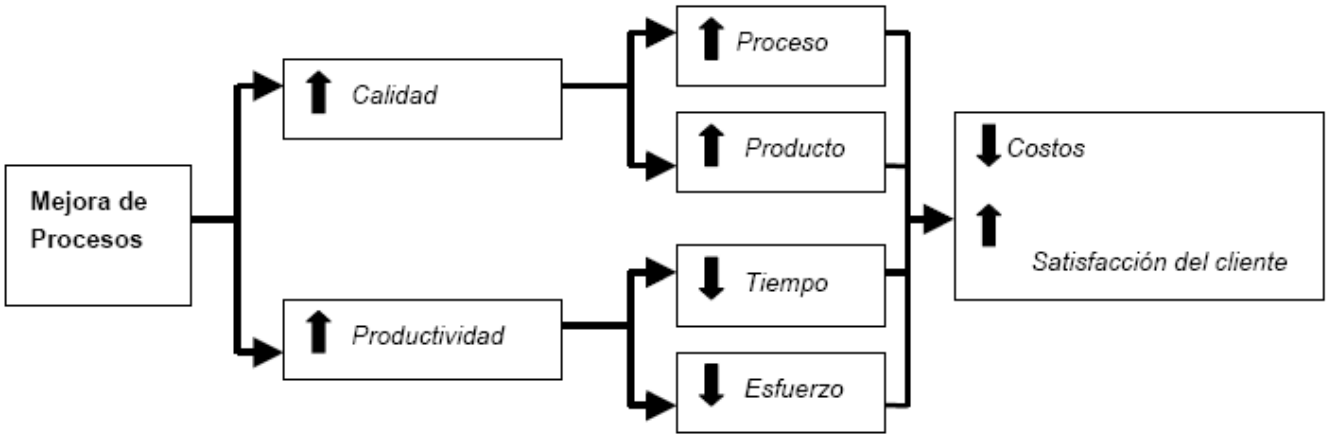


Figura 3: Reducción de costos y aumento en la satisfacción del cliente, como indicadores primarios de la mejora del proceso.

La mejora de los procesos significa optimizar la efectividad y la eficiencia mejorando también los controles, reforzando los mecanismos internos para responder a las contingencias y las demandas de nuevos y futuros clientes.

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

Para mejorar los procesos, se tienen que considerar, entre otros, los siguientes aspectos:

- Análisis de los flujos de trabajo.
- Fijar los objetivos de satisfacción del cliente (tanto internos como externos), para dirigir la ejecución de los procesos.
- Desarrollar las actividades de mejora con los propietarios y actores del proceso.

Mónaco y Vitali plantean que la mejora de procesos tiene como objetivos:

Aumentar la madurez de los procesos: grado en que están definidos, administrados, medidos y aplicados.
Aumentar la capacidad de los procesos: representa los resultados esperados que pueden ser obtenidos a partir de aplicar el proceso.

Para llevar a cabo un proyecto de mejora se requiere:

- **Querer mejorar**: está relacionado con las necesidades de la organización y la actitud del personal, la motivación y la personalidad de cada individuo.
- **Poder mejorar**: implica tener los medios necesarios y suficientes y contar con el conocimiento, experiencia y habilidad del trabajador, no solo para ejecutar bien sus tareas, sino también para ver las oportunidades de mejorarlas.
- **Actuar en consecuencia**: iniciar y llevar a cabo un proyecto de mejora de procesos.

Un proyecto de mejora de procesos exitoso requiere:

- Un patrocinador con poder de decisión: la Dirección de la organización está convencida y dispuesta a cambiar su historia.
- Sensibilización de toda la organización respecto de la Calidad y sus beneficios (todos deben ser y sentirse partícipes).
- Un modelo de referencia elegido, que sirva como hoja de ruta en el proceso.
  (VITALI, 2005)

“Se ha reportado en numerosos estudios, relativos al beneficio obtenido a raíz de procesos de mejora que la implantación de un proceso formal de verificación y validación supone importantes beneficios para las organizaciones software”. (CARLOS-III, 2006)

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

Para ayudar a la empresas en la mejora de sus procesos se crearon los modelos de mejora de procesos, ya que un modelo de mejora de procesos favorece que la organización “ponga sobre la mesa” sus procesos actuales, reflexionen sobre ellos para comprender qué es lo que hace y porqué lo hace, y en base a este estudio los optimicen para que estos sean lo más “eficaces” y “eficientes” como sea posible.

1.2.7 Modelos de mejora de procesos

A lo largo de los años se han definido diferentes modelos que pretenden caracterizar la calidad del software, haciendo una división clara entre modelos de calidad del proceso y modelos de calidad del producto. Los modelos de calidad de procesos influyen en la calidad de los productos y deben incluir procesos de aseguramiento de la calidad, planificación, verificación, validación y otros para cada entregable producido.

Los modelos de mejora de procesos se aplica en las organizaciones, generalmente mediante una inversión significativa, para estandarizar y mejorar continuamente sus procesos, y con el objetivo de obtener por un lado productos y servicios estandarizados, uniformes, estables y confiables que satisfagan en forma continua al cliente para el cual están diseñados, y por otro lado lograr productividad, competitividad, seguridad, replicabilidad y globalización de las actividades, operaciones, productos y servicios, entre otros beneficios. La aplicación de los modelos de calidad en una organización involucra un cambio cultural de la misma, fuertemente influenciado por actividades de sensibilización, capacitación y formación.

Un modelo indica “Qué hacer”, no “Cómo hacer”, ni “Quién lo hace”, proporcionando a las organizaciones que los utilizan:

- Un punto donde comenzar
- El beneficio de las experiencias pasadas de la comunidad participante
- Un lenguaje común y una visión compartida
- Un marco para priorizar acciones
- Una forma de definir lo que significa “mejora” para la organización

“Un modelo de calidad no es más que las técnicas, herramientas y metodologías que le facilitan a las empresas que se encargan de la fabricación de software ,guiar por un camino único el avance de dicho

proceso y así lograr que se cumpla con los requisitos iníciales pedidos por el cliente ya que esa es la base de la calidad de un producto.” (MANSO, 2005)

Aplicar normas y modelos de calidad es la mejor manera de asegurar que se cumpla con los requisitos pedidos por el cliente; reducen la probabilidad o riesgo de ocurrencia de errores en explotación logrando una mayor fiabilidad.

“Al cliente le place mucho más saber que el software que va a adquirir está certificado por alguna metodología que controle la calidad en todas las etapas de vida de un software, esto da una mayor confianza en el resultado del trabajo esperado.” (ALARCON, 2004)

El número de modelos y estándares ha seguido creciendo, haciendo cada vez más difícil la decisión para una empresa que tiene que escoger un modelo para la evaluación y mejora de su proceso de desarrollo. En la siguiente figura se muestran algunos modelos.

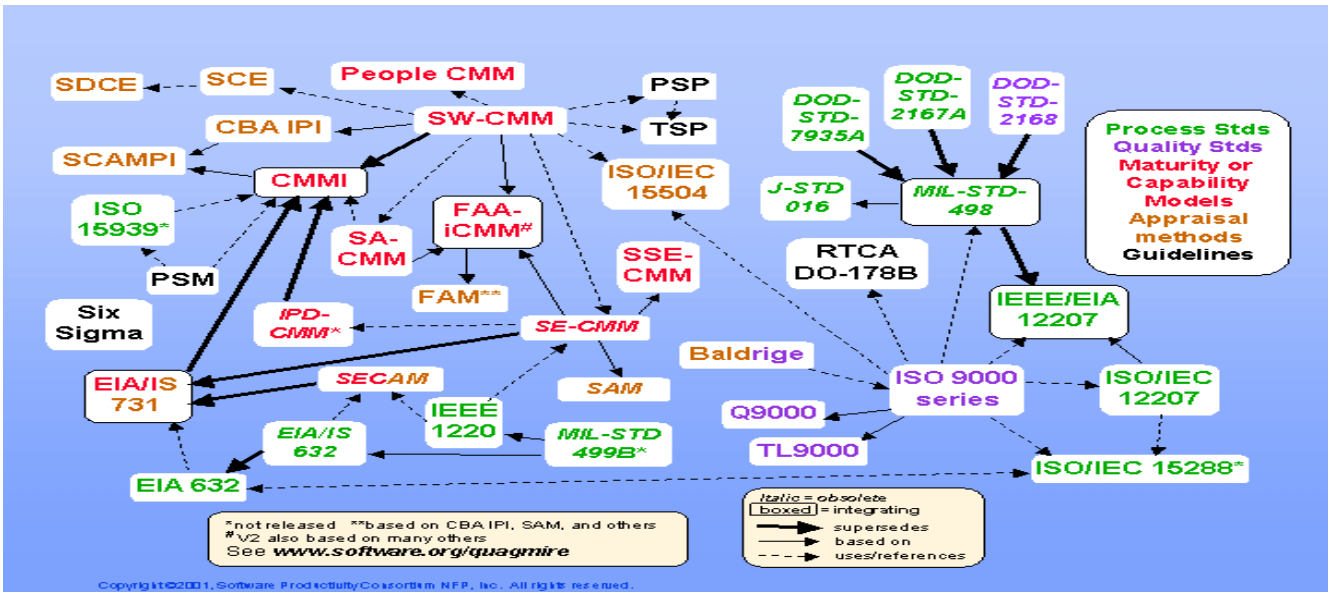


Figura 4: El lodazal de los modelos de procesos

(VILLA, 2004)

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

Los modelos a los que a continuación se hace referencia son los más significativos para la presente investigación, se pretende hacer una comparación entre los más usados con el fin de demostrar cuál es el más eficiente para llevar a cabo una mejora de procesos.

1.2.7.1 ISO 9001:2000

ISO 9000 es un conjunto de estándares internacionales para sistemas de calidad, diseñado para la gestión y aseguramiento de la calidad. Especifica los requisitos básicos para el desarrollo, producción, instalación y servicio a nivel de sistema y a nivel de producto. Las ISO no siempre son específicas para la fabricación de software y pueden ser difíciles de interpretar para aplicarlas.

Características:

- Establecen el QUÉ no el CÓMO.
- Están orientadas al PROCESO no al PRODUCTO:
- La calidad del proceso asegura mayores probabilidades de un producto de calidad.

Aspectos Positivos:

- Abarca la mayoría de las áreas funcionales de una organización, esto es, gestión, recursos humanos, producción, ingeniería y calidad
- Reconocimiento y apariencia internacional, marca de reconocido prestigio.
- Libertad de implementación y de interpretación de los requisitos.
- Incrementa las oportunidades de negocio en ciertos mercados y mejora la satisfacción del cliente.

Aspectos Negativos:

- No proporciona información de cómo aplicarlo a empresas de menor tamaño.
- A causa de la amplia aplicabilidad del estándar ISO, hay pocas directrices para su implementación en algunas industrias o campos específicos.
- No existen directrices para su aplicación en una división o en una sucursal de una gran empresa.
- No es específica para la industria de software.

(VILLA, 2004)

“Esta es una de las normas más populares pero pueden surgir errores a la hora de adaptar esta norma a un proyecto por lo genérica que resulta ser.” (INFORMATICA, 2000.)

1.2.7.2 ISO/IEC 15504 (SPICE)

ISO/IEC 15504 es un estándar internacional de evaluación y determinación de la capacidad y mejora continua de procesos de ingeniería del software, con la filosofía de desarrollar un conjunto de medidas de capacidad estructuradas para todos los procesos del ciclo de vida y para todos los participantes. Es el resultado de un esfuerzo internacional de trabajo y colaboración y tiene la innovación, en comparación con otros modelos, del proceso paralelo de evaluación empírica del resultado.

Aspectos positivos:

- ISO/IEC desarrolla un modelo de 2 dimensiones de evaluación de la capacidad del proceso, donde se valora la organización de desarrollo software en la dimensión del proceso contra los atributos del proceso en la dimensión de capacidad.
- La primera versión estructuraba el modelo en nueve partes, pero en el curso de los debates y votaciones, en aras de reducir el tamaño del estándar, se decide que se divida en cinco partes:
- Parte 1. Conceptos y Vocabulario. Publicada en (7/10/04)
- Parte 2. Realizando una Evaluación (Requisitos, normativa). Publicada en (30/10/03)
- Parte 3. Guía para Realización de Evaluaciones. Publicada (6/1/04)
- Parte 4. Guía para el Uso de Resultados de Evaluaciones. Publicada (6/7/04)
- Parte 5. Un Modelo de Evaluación de Procesos Ejemplar (estimado diciembre de 2004)
- Define un conjunto de criterios de conformidad para permitir la comparación de modelos externos de procesos y encontrar requisitos comunes.

Aspectos negativos:

- En un principio se pensaba que el dominio de procesos debería ser más amplio para abarcar todos los posibles ciclos de vida, siendo difícil que todos los atributos de los procesos fueran universales, aplicables a todos los procesos y prácticas base.
- La dimensión de la capacidad ha alcanzado un alto grado de dificultad y existen solapamientos con la dimensión de los procesos.
- La complejidad de las evaluaciones (y por consiguiente el costo) es significativamente más alta que en otros modelos.

(VILLA, 2004)

"Este modelo ha resultado un tanto popular aunque no define bien las áreas de procesos dentro del proyecto, además es un poco abstracto y también muy genérico lo que trae problemas de interpretación a la hora de adaptar dicho modelo al proyecto". (INFORMATICA, 2000.)

1.2.7.3 CMMI

El CMMI es un modelo de calidad del software que clasifica las empresas en niveles de madurez. Estos niveles sirven para conocer la madurez de los procesos que se realizan para producir software. Constituye un marco de referencia de la capacidad de las organizaciones de desarrollo de software en el desempeño de sus diferentes procesos, proporcionando una base para la evaluación de la madurez de las mismas y una guía para implementar una estrategia para la mejora continua de los procesos.
(VILLA, 2004)

Aspectos positivos:

- Incluye prácticas de institucionalización, que permiten asegurar que los procesos asociados con cada área de proceso sean efectivos, repetibles y duraderos, mediante características comunes como: compromiso de la realización, capacidad de realización, actividades realizadas, mediciones y análisis, y la verificación de la implementación.
- Brinda una guía paso a paso para la mejora, a través de niveles de madurez y capacidad.
- Posibilita la transición del "aprendizaje individual" , al aprendizaje de la organización por mejora continua, lecciones aprendidas y uso de bibliotecas y bases de datos de proyectos mejorados.

Aplicaciones:

Es un modelo que:

- Describe los elementos claves de un proceso de software efectivo.
- Describe el mejoramiento evolutivo que una organización de software debe realizar para ir de un proceso inmaduro a un proceso disciplinado y maduro.
- Cubre las actividades de planificación, administración e ingeniería del proceso de desarrollo y mantención de software.

Implementa un marco ordenado y disciplinado para:

- Mejorar los procesos de desarrollo y mantención de software.
- Aumentar la calidad de los productos.
- Disminuir las crisis en los proyectos.
- Disminuir los costos anormales.
- Aumentar la satisfacción del cliente.
- Mejora la habilidad para alcanzar las metas de costo, planificación, funcionalidad y calidad del producto. (MORAGA, 2004)

Aspectos negativos:

- El CMMI puede llegar a ser excesivamente detallado para algunas organizaciones.
- Puede ser considerado prescriptivo.
- Requiere mayor inversión para ser completamente implementado.

1.2.8 Comparación entre los modelos analizados

Los modelos analizados *ISO 9001:2000, CMMI y SPICE* son los más referenciados para la mejora de procesos, por tal motivo se realiza entre ellos un estudio comparativo.

Aspectos	ISO 9001:2000	CMMI	ISO/IEC15504 (SPICE)
Ámbito de Aplicación	Genérico	Software y Sistemas	Software y Sistemas
En su favor	El más extendido y sencillo	El de mayor prestigio	Más consensuado y probado
En su contra	Simple, general, no guía paso a paso	Difícil de entender, mayor inversión, prescriptivo	Difícil en capacidad, complejo para evaluar
Procesos	Estructura propia	Estructura propia	Delega en ISO 12207, por mayor aplicabilidad
Validación	Encuestas de satisfacción	Encuestas de satisfacción y casos de estudio	´Trials´ y esfuerzo empírico
Objetivo	Cumplimiento de	Mejora del proceso,	Valoración del proceso y

	requisitos de calidad por procesos	determinación capacidad contratista	guía para la mejora
Representación	Plana	Continua y por etapas	Continua (por etapas a nivel de proceso)
Técnicas de análisis	Guías y listas de comprobación	Cuestionarios de evaluación	Varios
Método para mejora de procesos	Ninguno, guía ISO 9004	IDEAL, mapa guiado	SPICE 4taParte

Tabla 1: Comparación ISO, CMMI, SPICE

Conclusiones del estudio comparativo

Se puede decir que CMMI a pesar de requerir una gran inversión para ser completamente implementado, es un modelo de procesos que ayuda a medir y analizar con facilidad los proyectos, ya que posee una línea definida y detallada para la producción de software.

CMMI identifica y detecta a tiempo los problemas existentes en la organización así como las posibles soluciones a llevar a cabo en aras de lograr una mejora de los procesos, proporciona una guía paso a paso que posibilita que se alcancen el nivel de madurez y capacidad necesarias para obtener productos con calidad, a diferencia de la norma ISO 9001:2000 y el modelo SPICE.

La complejidad de las evaluaciones del modelo SPICE es significativamente más alta que en CMMI e ISO 9001:2000, siendo más fácil a la hora de realizar este proceso en CMMI.

CMMI brinda seguridad y confiabilidad a todas aquellas empresas que la aplican ya que incluye prácticas de institucionalización que permiten asegurar que los procesos asociados con cada área de proceso sean efectivos, repetibles y duraderos. Además tiene dos formas de representación, la continua y la escalonada, dando la posibilidad a las empresas de representarlos según las necesidades que estas tengan.

En cuanto a los procesos en CMMI e ISO 9001:2000 poseen una estructura propia, mientras SPICE delega en ISO 12207 por tener una mayor aplicabilidad.

Con el uso de CMMI se alcanzan valores favorables para la organización si de costos, tiempo, productividad, calidad y satisfacción del cliente se trata.

Por todos los aspectos antes planteados, CMMI es el modelo de procesos más conveniente para llevar a cabo el programa de mejora que se está llevando a cabo en la universidad.

1.2.9 Introducción a CMMI

A mediados de la década de los 90´s, el Instituto de Ingeniería de Software - SEI perteneciente a la Universidad Carnegie Mellon, decide unificar todos los modelos que había creado con anterioridad fundamentalmente CMM-SW (Capability Maturity Model for Software Engineering), embarcándose en un esfuerzo que culmina en el año 2002 dando origen a una nueva generación llamada CMMI.

CMMI es un modelo de mejora de procesos y adquisición de software. Proporciona un marco de referencia para evaluar la efectividad de los procesos actuales, facilitando con ello la definición de actividades, prioridades y metas para garantizar la mejora continua.

Ha sido diseñado para integrar y dar cobertura a las disciplinas definidas en su antecesor CMM (Modelo de Madurez de Capacidad).Las disciplinas son las siguientes:

- Ingeniería de Software (CMMI - SW)
- Ingeniería de Software (CMMI - SW) + Ingeniería de Sistemas (CMMI - SE)
- Ingeniería de Software (CMMI - SW) + Ingeniería de Sistemas (CMMI - SE) + Desarrollo Integrados de Procesos y Productos (IPPD).
- Ingeniería de Software (CMMI - SW) + Ingeniería de Sistemas (CMMI - SE) + Desarrollo Integrados de Procesos y Productos (IPPD) + Administración de Proveedores (CMMI - SS).

Estas disciplinas también son conocidas como cuerpos de conocimiento o modelos CMMI.

1.2.9.1 ¿Por qué se aplica CMMI?

Estas son algunas razones por las cuales numerosas empresas aplican CMMI hoy en el mundo:

(HUACOTO, 2005)

- Es un modelo que aplica estándares de calidad.
- Provee un enfoque más efectivo e integrado a Ingeniería de Sistemas y de Software.
- Construye procesos desde un inicio, o sobre inversión previa usada con SW-CMM.
- Provee un enlace o relación más explícita entre la Gestión y la Ingeniería con los objetivos del negocio.
- Provee mayor visibilidad del ciclo de vida del producto; y las actividades de ingeniería ayudan a asegurar que los productos y servicios satisfacen las expectativas de los clientes.
- Incorpora lecciones aprendidas de otras áreas, de mejores prácticas e implanta prácticas de alta madurez más robustas.
- Incluye funciones organizacionales que son críticas para los productos y servicios.
- Soporta integración futura con otros modelos CMMI de disciplinas específicas.
- Muestra el camino a seguir en la gestión de proyectos, integrando de una manera ordenada los procesos y los productos.

Entonces…. ¿Por qué aplicar CMMI en la UCI?

La Dirección de Calidad de Software, como máxima responsable de la calidad del producto de software de la universidad decidió escoger el modelo CMMI como modelo de mejora de proceso a implantar. Durante la investigación para este trabajo se obtuvo más datos que ayudan a comprender el porqué se escogió este modelo; mediante una entrevista realizada a Dennis Neuland Agüero miembro del Grupo de Mejora. Donde este responde: “es un modelo que tiene en cuenta a las empresas que no tienen nada definido y propone un modelo para ir aumentando el nivel de madurez o capacidades de la misma. Otra variante que pudiera ser aplicada son las Normas ISO pero esta tiene una única opción, o certificas ISO o no lo certificas. CMMI propone varios niveles que permiten progresar paulatinamente. CMMI está orientado directamente al software y se complementa con otros procesos como pueden ser, PSP y TSP que son del

mismo SEI (Software Engineering Institute). Además el SEI propone un conjunto completo de herramientas para el trabajo con CMMI, como: CMMI- Modelo de Procesos, IDEAL- programa de mejora de procesos basados en CMMI, SCAMPI- Método de Evaluación. Otra razón es que este modelo es gratis, lo que cuesta son los cursos, entrenamiento y la evaluación, este último si la empresa lo cree necesario. Cuando se utiliza ISO todo hay que pagarlo. Otra razón por la que se propone CMMI es que la empresa misma si cuenta con personal preparado puede implantar este Modelo de Madurez, mientras que con ISO no es tan simple, ya que es más necesaria la asesoría".

1.2.9.2 Representación de CMMI

CMMI soporta dos caminos de mejoras. Un camino le permite a las organizaciones un incremento en mejoras de procesos correspondiente a un área de proceso individual que la organización seleccione. El otro camino le permite a las organizaciones mejorar un grupo de procesos relacionados, guiándolas en grupos sucesivos de áreas de procesos propuestos por el modelo.(SEI, 2006)

Estos dos caminos de mejoras son asociados con los dos tipos de niveles que concuerdan con las dos representaciones:

La **representación continua** ofrece máxima flexibilidad al usar CMMI para la mejora de proceso. Un organismo puede elegir mejorar la función de un proceso solo en un lugar conflictivo relacionado, o puede dedicarse a varias áreas que están estrechamente relacionadas para los objetivos comerciales de la organización.

Esta representación es utilizada cuando *se conocen,* que procesos deben ser mejorados en su organización. (SEI, 2006)

La **representación escalonada** ofrece una forma ordenada, dispuesta a afrontar mejoras basadas en modelo de proceso una etapa a la vez (es decir, no se comienza en la etapa inmediata superior sin antes haber logrado la etapa que la antecede). Lograr cada etapa asegura una adecuada infraestructura de proceso siendo colocada como una base para la siguiente etapa.

Esta representación es utilizada cuando *no se conocen,* que procesos deben ser mejorados en su organización, y el modelo propone que se avance por niveles de una forma escalonada. (SEI, 2006)

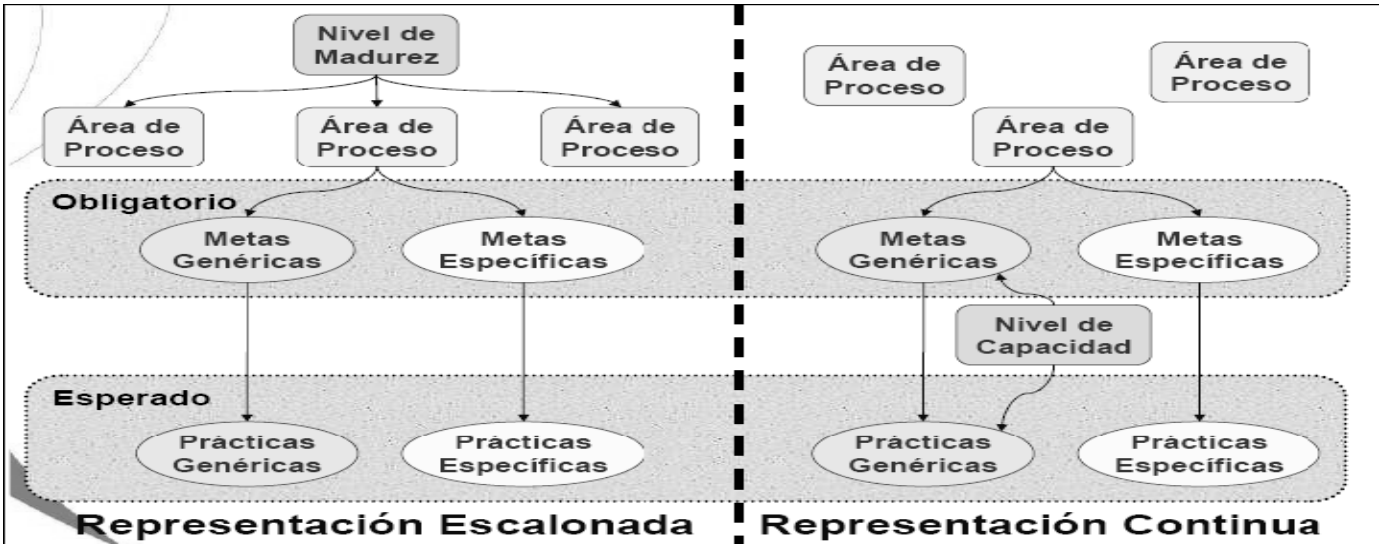


Figura 5: Representación de CMMI

(PARRO, 2007)

Estos dos caminos de mejora son asociados con los dos tipos de niveles que concuerdan con las dos representaciones. Para la *representación continua*, se usó el término "nivel de capacidad". Para la *representación escalonada*, se usó el término "nivel de madurez". Al final con los dos caminos se llega al mismo objetivo, que es una mejor calidad del producto o servicio. Teniendo como referencia estos dos términos propuesto por el modelo CMMI, surge la interrogante, cual representación es conveniente escoger.

¿Por qué se escoge la representación Escalonada?

Debido a la propia estrategia de la Dirección de Calidad de Software; esta, igualmente decidió escoger la Representación Escalonada. Gracias a la entrevista que se realizó a Dennis Neuland Aguero miembro del Grupo de Mejora, se obtuvo más datos del porqué seleccionar dicha representación. Este comenta: “hoy en la UCI por tener poca experiencia en procesos de mejoras y no tener antecedentes en Modelos CMM, se ha escogido la representación Escalonada, debido a el centro no tiene todos sus procesos definidos

que le permita escoger el área de proceso, independientemente del nivel que sea, por lo que se decide iniciar con la implementación de las áreas de procesos del nivel 2, y así alcanzar la certificación del modelo CMMI en este nivel."

El modelo CMMI define 5 niveles mediante los cuales se describen los distintos grados de madurez de una organización. Para que una organización se encuentre en un determinado nivel es necesario cumplir con todas las actividades definidas para ese nivel y para los niveles anteriores.

- *Nivel 1 – Inicial:* Procesos impredecibles, pobremente controlados y reactivos.
- *Nivel 2 – Gestionado:* Procesos caracterizados por los proyectos y su gestión.
- *Nivel 3 – Definido:* Procesos caracterizados en la organización, y con acciones proactivas.
- *Nivel 4 – Gestionado Cuantitativamente:* Procesos cuantitativamente medidos y controlados.
- *Nivel 5 – Optimizado:* Focalizado en la mejora continua de proceso.

Cada nivel de madurez –con excepción del inicial- queda caracterizado por un conjunto de áreas de proceso las que a su vez quedan definidas por uno o varios objetivos específicos y un objetivo genérico. Cada uno de ellos tiene vinculado un conjunto de prácticas, llamadas específicas y genéricas respectivamente que al ser ejecutadas colectivamente, permiten cumplir con algún objetivo que es considerado importante para el modelo.

Los *objetivos y prácticas genéricas* tienen que ver con el grado de institucionalización de los procesos. Son llamados así porque son los mismos en todas las áreas de proceso. Cumplir con un objetivo genérico de un área de proceso determinada implica tener un mayor control de la planificación e implementación de los procesos vinculados a esa área de proceso.

Los *objetivos y prácticas específicas* están vinculados a un área de proceso determinada. Son considerados elementos que deben ser satisfechos para implementar exitosamente los procesos relacionados con un área de proceso en particular.

Este trabajo estará centrado en las áreas de procesos que propone CMMI para el nivel 2, las cuales son siete en total:

1. *Administración de Requerimientos (REQM)*
2. *Planificación del Proyecto (PP)*
3. *Monitoreo y Control del Proyecto (PMC)*
4. *Medición y Análisis (MA)*
5. *Aseguramiento de la Calidad de Productos y Procesos (PPQA)*
6. *Administración de la Configuración (CM)*
7. *Administración de Acuerdos con Proveedores (SAM)*

En una organización que haya alcanzado este nivel de madurez encontraremos que hay una disciplina para la gestión de proyectos que se mantiene aún en periodos de estrés. Los recursos estarán capacitados para hacer su trabajo, y habrá políticas organizacionales formalmente establecidas, cuyo cumplimiento será monitoreado. Habrá visibilidad de las actividades realizadas, y los proyectos se ejecutarán siguiendo un plan y de acuerdo a un proceso formalmente establecido.

Beneficios que aporta:

CMMI como todo modelo de mejora de procesos proporciona unos beneficios a las organizaciones que los utilizan, y en este caso en particular dichos beneficios los podemos agrupar en los siguientes puntos:

Una reducción de costos por:

- Una mayor fiabilidad de las planificaciones (estimaciones basadas en hechos).
- Reducción de reproceso.
- Acuerdos claros sobre el servicio y la funcionalidad del producto a entregar.

Un aumento en la confiabilidad por:

- Reducción consistente de errores (reduciendo el número de defectos y detección en las fases más tempranas del ciclo de vida).
- Cumplimiento de fechas.

Una mayor efectividad por:

- Visibilidad sobre el proceso y sobre el producto.
- Operar con estándares documentados.
- Personal formado.

Una mejora en la imagen de marca por:

- Una mayor calidad de los productos entregados

Una mejora en la gestión de la subcontratación por:

- Una mejor gestión de los acuerdos y contratos con proveedores, para mejorar la eficacia de los productos y servicios comprometidos.

Evaluación:

En paralelo con el desarrollo de CMMI, el SEI elaboró un método para la evaluación formal del modelo denominado SCAMPI (*Standard CMMI Appraisal Method for Process Improvement*). El método define una serie de reglas para la evaluación del modelo, las cuales deben utilizarse para valorar las distintas partes del mismo durante una evaluación formal. Estas reglas hacen que sea necesario utilizar herramientas, ya que el método de evaluación deja de ser una simple encuesta para convertirse en una evaluación detallada y casi matemática.

En la actualidad el modelo CMMI incluyendo por supuesto su modelo IDEAL y su método de evaluación SCAMPI, se ha convertido en un requisito para la contratación de servicios, y para el desarrollo de software. Incluso es reconocido como estándar para la certificación de procesos software en determinados sectores y países.

1.2.10 Modelo IDEAL y su aplicación en un programa de mejora

El modelo IDEAL es utilizado por muchas organizaciones en el mundo para emprender sus programas de mejoras de procesos. Dicho modelo proporciona un conjunto de actividades coherentes para sustentar la adopción de las prácticas recomendadas por CMMI.

“El modelo IDEAL provee un enfoque disciplinado de ingeniería para la mejora del proceso de software, focaliza la gerencia del programa de mejoras y establece los fundamentos para una estrategia de largo plazo”. (HISTA, 2007)

Las cinco fases del modelo IDEAL contienen un sistema de tareas de las cuales se especifica su propósito, objetivo, criterios para la entrada y salidas y subtareas, mediante las cuales se ejecutan la puesta en práctica de un programa de mejora. Este modelo se basa en recrear un panorama ideal, por

supuesto habrá siempre acontecimientos del mundo real que eviten que las organizaciones sigan la secuencia de tareas en la mejora de proceso. Los responsables de esta tarea deben adaptar la guía a su situación particular. La secuencia propuesta por el modelo se recomienda porque las acciones están documentadas y ayudan a definir planes, acciones y resultados.

Fases y Actividades del modelo IDEAL:

El modelo presenta cinco fases, que proporcionan un lazo continuo con los pasos necesarios para desarrollar un programa de mejora, tal y como se muestra en la siguiente figura.

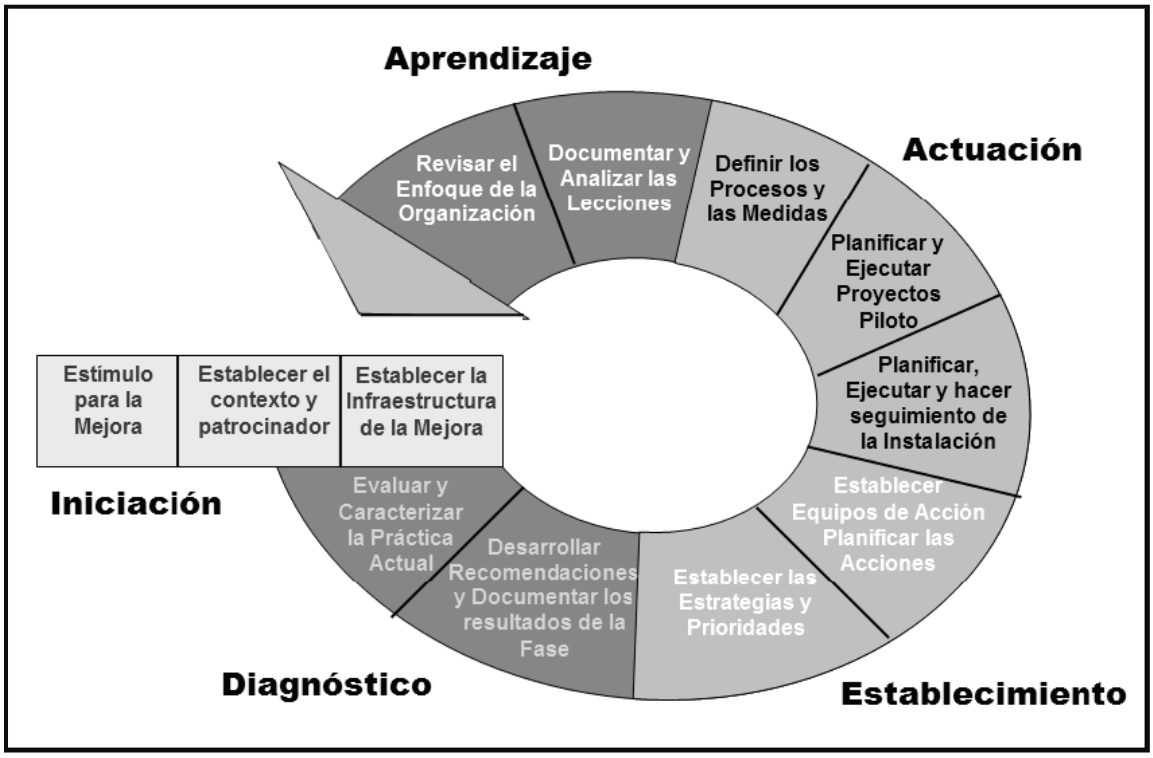


Figura 6: Ciclo del modelo IDEAL

(MCFEELEY, 2003)

En el cuadro siguiente se explica que se hace en cada fase.

I	Iniciar	Definir la base para un proceso de mejora, se garantiza la disponibilidad de recursos, y la infraestructura.
D	Diagnosticar	Identificar dónde está posicionada la Organización y a dónde quiere llegar. Establece los niveles actuales de madurez de procesos, las descripciones, métricas, etc. Inicia el plan de acción de desarrollo.
E	Establecer	Planificar las acciones a ejecutar para alcanzar el estado deseado. Establece las metas y prioridades, completa el plan de acción.
A	Actuar	Investigar y desarrollar las soluciones a los problemas de los procesos. Extender las mejoras exitosas del proceso a toda la organización.
L	Aprender	Aprender de la experiencia realizada y visualizar oportunidades de mejora.

Tabla 2: Fases del modelo IDEAL

Estas fases del Modelo IDEAL contienen un conjunto de tareas que se realizan durante la ejecución de un programa de mejoras de procesos.

- **Fase "Inicio"**

Su propósito es establecer los fundamentos básicos para garantizar y dar soporte a la iniciativa de mejoras de procesos. En la misma, el grupo de administración establece cuáles son los objetivos de la organización y de la mejora de procesos. El apoyo de la alta dirección y de los gerentes en general es fundamental para el éxito del programa de mejoramiento. En esta etapa se garantiza la disponibilidad de recursos, la infraestructura y la priorización del proyecto de mejoramiento. Es importante destacar que las actividades de esta fase determinan el éxito o el fracaso del programa.

Actividades:

- **Establecer el contexto**: es el detonante de la iniciativa. Es importante conocer cuáles son las razones para mejorar e identificar los aspectos comerciales u organizacionales que se pretende asegurar.
- **Establecer el patrocinio de la alta dirección:** el apoyo de los distintos niveles de dirección es crítico, su ausencia o debilidad es una receta para fracasar. El apoyo debe ser claro, efectivo y constante.
- **Establecer la infraestructura adecuada**: contar con un mecanismo para dirigir e implementar el proyecto de mejoras. Se debe capacitar a los distintos gerentes y personal de proyecto. Conocer CMMI es fundamental.

Los elementos de la infraestructura deben tener claramente definidos sus deberes, roles y responsabilidades para asegurar el éxito del programa SPI (Software Process Improvement). El propósito principal para establecer una infraestructura para un programa de SPI es construir los mecanismos necesarios para ayudar a la organización a institucionalizar la mejora continua de procesos. La infraestructura establecida de cualquier programa de SPI es fundamental para el éxito de ese programa. Una sólida y eficaz infraestructura puede sostener un programa de desarrollo SPI hasta que ésta comience a producir resultados visibles y puede significar la diferencia entre el éxito de una SPI programa y un fracaso.

- **Fase "Diagnóstico"**

Su propósito es evaluar mediante un método formal las fortalezas y debilidades del proceso actual utilizado en los proyectos. Los objetivos del programa se relacionan con las prácticas existentes y se determinan aquellas que no están suficientemente desarrolladas. Generalmente esta fase es desarrollada con el asesoramiento de expertos en el modelo de referencia.

Actividades:

- **Determinar el estado actual y el esperado**: implica una evaluación de los proyectos de la organización. Es equivalente a identificar el punto de partida y el punto de destino antes de hacer un viaje. CMMI sirve como un modelo de referencia para determinar el estado deseado que se pretende alcanzar.

- **Plantear recomendaciones y documentar los resultados de la fase**: un equipo experto identifica las debilidades y fortalezas de las prácticas actuales, en base a la información analizada durante la evaluación. Sus recomendaciones sirven como entrada al plan de acción para la mejora. La salida es generalmente un informe de resultados.

- **Fase "Establecimiento"**

Su propósito es realizar la planificación específica de las mejoras que se desea alcanzar. En la misma se desarrolla un plan de proyecto, se establece la estrategia y se eligen prioridades para la acción, en base a recursos, necesidades urgentes, efectividad de la acción, impacto y otros.

Actividades:

- **Establecer las prioridades:** la mejor comprensión de las necesidades que se han ido identificando en los pasos previos, permite establecer la estrategia y los recursos necesarios para completar el trabajo. Se identifican a los recursos competentes que participarán en el proyecto de mejoras.

- **Elaboración del Plan de Acción**: las recomendaciones de la evaluación se trasforman en un plan concreto que satisface las prioridades y necesidades de la organización.

Habitualmente se consideran acciones de corto, mediano y largo plazo. El plan incluye calendarios de proyecto, tareas, hitos, puntos de decisión, recursos, responsabilidades, métricas, mecanismos de seguimiento, riesgos con sus respectivas estrategias de mitigación.

- **Fase "Actuación"**

El propósito es implementar la mejora de procesos llevando a cabo el plan de acción. Aquí se introducen o mejoran los procesos, se entrena a los respectivos niveles de personal, se miden los avances y beneficios logrados, se realizan proyectos pilotos, se implantan los procesos mejorados en los proyectos nuevos o existentes, se hacen mini evaluaciones para constatar la evolución del plan y otros.

Actividades:

- **Desarrollar la solución:** implica la definición e integración de los procesos, herramientas, información, conocimiento y habilidades, tanto existentes como nuevas.

- **Probar la solución:** una vez que las soluciones han sido diseñadas, se necesita probarlas en proyectos pilotos antes de decidir institucionalizarlas en el resto de los proyectos.

- **Refinar la solución:** cuando la solución propuesta ha sido aplicada en un proyecto piloto, se puede refinar para reflejar el conocimiento, la experiencia y las lecciones aprendidas en el ensayo.

- **Implementar la solución:** una vez que se ha decidido que se tiene una solución aceptable, se procede a aplicarla a lo largo de la organización.

- **Fase "Aprendizaje"**

El propósito es aprender de la experiencia del ciclo recién realizado y aumentar la habilidad de la organización para mejorar los procesos en forma continua. Se determinan los logros, el esfuerzo invertido, la manera en que las metas fueron satisfechas y la forma más adecuada de implementar cambios en el futuro. Se utilizan las mediciones y registros acumulados durante la aplicación de las etapas anteriores del ciclo.

Actividades:

- **Analizar y validar los resultados:** identificar el grado en que el esfuerzo invertido logró los propósitos deseados. Las lecciones se recolectan, se analizan, se resumen y se documentan. Se reexaminan las necesidades de la empresa identificadas en la fase Inicio para ver si fueron satisfechas.

- **Revisar el enfoque seguido y proponer acciones futuras:** se plantean y documentan recomendaciones que resultan del análisis y la validación. Se proponen pautas y acciones para el siguiente plan de mejoras.

Generalmente el final del primer ciclo coincide con las primeras etapas del ciclo siguiente. Se recomienda efectuar una nueva evaluación para determinar las nuevas necesidades y fortalezas que servirán de entrada al nuevo plan de acción.

1.2.11 COBIT como método para monitorear y controlar el proceso de mejora con IDEAL

Una correcta planificación de un programa de mejoras es el cimiento para proceder a la confección de un minucioso seguimiento, verificando que dicha planificación se cumpla. Para darle un acertado monitoreo y control a esta planificación hace falta el uso de alguna herramienta que propicie y facilite este trabajo.

Los Modelos de Control Interno son una herramienta eficaz que entre sus características esta el proceso de monitoreo y control. El Control Interno no es más que el proceso integrado a las operaciones, efectuado por la dirección y el resto del personal de una entidad para proporcionar una seguridad razonable. La utilización de los modelos de control Interno brindan los siguientes beneficios:

- Confiabilidad de la información.
- Eficiencia y eficacia en las operaciones.
- Cumplimiento y seguimiento de la planificación establecida.
- Control de los recursos de todo tipo, a disposición de la entidad.

COBIT(Objetivos de Control para la Información y Tecnologías Relacionadas) es el modelo para el Gobierno de las Tecnologías de la Información (TI) desarrollado por la Asociación para la Auditoría y Control de Sistemas de Información (ISACA) y el Instituto de Gobierno de las Tecnologías de la Información (ITGI). Es una metodología aceptada mundialmente para el adecuado control de proyectos de tecnología, los flujos de información y los riesgos que éstas implican. Se utiliza para planear, implementar, controlar y evaluar el gobierno sobre TIC; incorporando objetivos de control, directivas de auditoría, medidas de rendimiento y resultados, factores críticos de éxito y modelos de madurez.

COBIT brinda un conjunto de buenas prácticas a través de un marco de trabajo de dominios y procesos, y presenta las actividades en una estructura manejable y lógica. Las buenas prácticas de COBIT representan el consenso de los expertos. Están enfocadas fuertemente en el control y menos en la ejecución. Estas prácticas ayudarán a optimizar las inversiones facilitadas por la TI, asegurarán la entrega del servicio y brindarán una medida contra la cual juzgar cuando las cosas no vayan bien.

Independientemente de la realidad tecnológica de cada caso concreto, COBIT determina, con el respaldo de las principales normas técnicas internacionales, un conjunto de mejores prácticas para la seguridad, la

calidad, la eficacia y la eficiencia en TI que son necesarias para alinear TI con el negocio, identificar riesgos, entregar valor al negocio, gestionar recursos y medir el desempeño, el cumplimiento de metas y el nivel de madurez de los procesos de la organización.

Este marco de referencia define claramente siete características que deben estar presentes en el manejo de la información:

- Efectividad
- Eficiencia
- Confidencialidad
- Integridad
- Disponibilidad
- Confiabilidad

El enfoque hacia procesos de COBIT se ilustra con un modelo de procesos, el cual subdivide TI en 34 procesos de acuerdo a sus 4 dominios o áreas de responsabilidad de planear, construir, ejecutar y monitorear, ofreciendo una visión de punta a punta de la TI. Los conceptos de arquitectura empresarial ayudan a identificar aquellos recursos esenciales para el éxito de los procesos, es decir, aplicaciones, información, infraestructura y personas.

Dominios:

1. **Planear y Organizar (FABIEN)PO)**

Conduce la estrategia y las tácticas y corresponde a la identificación de la forma en que la información tecnológica puede contribuir mejor a alcanzar los objetivos de gestión.

2. **Adquirir e Implementar (AI)**

Para llevar a cabo la estrategia es necesario identificar, desarrollar y adquirir soluciones de TI apropiadas, así como implementarlas e integrarlas en los procesos de gestión.

3. **Entregar y Dar Soporte (DS)**

Corresponde con la distribución normal de los servicios requeridos, que van desde las tradicionales operaciones sobre seguridad y continuidad hasta la formación.

4. Monitorear y Evaluar (ME)

Todos los procesos de TI deben evaluarse regularmente en el tiempo para comprobar su calidad.

Asociados a los 4 dominios existen 34 procesos de alto nivel, desagregados en 302 procesos y/o actividades de menor jerarquía. A su vez cada proceso y/o actividad de menor jerarquía involucra una sumatoria de buenas prácticas necesarias de considerar para el cumplimiento de los requisitos de control exigidos para cada caso, estableciendo los resultados deseados o propósitos a ser alcanzados mediante su implementación. Además existen medidas de control de diversa naturaleza: de indicadores, y de modelos de madurez.

Algunos de los beneficios que se obtienen al implementar COBIT son los siguientes:

- Enfocarse en objetivos y necesidades del negocio mejorando la cooperación y comunicación entre los administradores del negocio y los auditores.
- Ayuda a los administradores a entender como los asuntos de seguridad y control benefician sus áreas de operación
- Ayuda a las organizaciones a compararse con la competencia e implementar mejores prácticas de objetivos de control y la tecnología relacionada.
- Las organizaciones generan confianza y credibilidad hacia sus clientes.
- Permite a las organizaciones cumplir con requerimientos regulatorios.

La evaluación de la capacidad de los procesos basada en los modelos de madurez de COBIT es una parte clave de la implementación del gobierno de TI. Después de identificar los procesos y controles críticos de TI, el modelado de la madurez permite identificar y demostrar a la dirección las brechas en la capacidad. Entonces se pueden crear planes de acción para llevar estos procesos hasta el nivel objetivo de capacidad deseado.

1.3 Análisis de otras soluciones existentes

1.3.1 Ámbito internacional

En el ámbito internacional son muchas las organizaciones que hacen uso del modelo CMMI. En el último informe del SEI según la consultora española TQS (Tecnología y Calidad de Software, S.A) sobre la

situación actual de CMMI, que data de marzo de 2005, constata un incremento en la utilización del modelo, que es utilizado en un 60 por ciento por empresas comerciales, y en un 40 por ciento por agencias gubernamentales, militares y subcontratistas. Este informe también señala que se trata de un sistema que se aplica con independencia del tamaño de la empresa. Las empresas que han reportado datos para esta evaluación, un total de 630 repartidas por 36 países, las que ya han podido comprobar los beneficios de la implantación de CMMI, que se resumen en mejoras en la ejecución de proyectos y sus costes de desarrollo y mejoras en la calidad de los productos. Concretando más, el SEI, en su informe Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results, divide en cinco categorías los beneficios de CMMI, a las que TQS ha añadido dos más, derivados de su experiencia como consultora.(PÉREZ, 2008)

En el caso de COBIT, es utilizado en los más de 100 países miembros de la ISACA, por una gran variedad de empresas. En el sitio Web de la asociación, se puede acceder a 38 casos de estudio de empresas que han utilizado el estándar, agrupados en los dominios Consultoría de TI, Educación, Servicios Financieros, Aseguradoras, Gobierno, Salud, Manufactureras, Transporte, e Industrias Básicas. Esto da la idea de la variedad de implementaciones que ha tenido, habiendo sido adaptado a las necesidades de cada una.

En todos los casos que aparecen en la web como casos de estudio se trata de grandes empresas que poseen gran cantidad de recursos tecnológicos y que se han visto en la necesidad de evaluar y optimizar su gobernabilidad sobre las TI, entre las más conocidas están Sun Microsystems (Estados Unidos), Universidad Tecnológica Curtin (Australia Occidental), Allstate (Estados Unidos), Harley-Davidson (Estados Unidos), Provincia de Mendoza (Argentina), la Society for Worldwide Interbank Financial Telecommunication (SWIFT, Bélgica), Organización Gubernamental Australiana (Canberra), Departamento de Defensa de los Estados Unidos, Royal Philips Electronics (Holanda), y el gobierno de Uruguay, entre otros.

A continuación se ofrece algunas de las herramientas existentes, que toman como guía de referencia a CMMI y COBIT.

SIMPLe

El Sistema de Implementación de Mejora de Procesos SIMPLe es una herramienta que está orientada a dar soporte al jefe de proyecto de calidad en la gestión de un proyecto de mejora de procesos. Representa una bitácora documental del desarrollo del proyecto de mejora de la calidad de procesos, en donde se pueden registrar fácilmente mediante actividades, comentarios y artefactos, cómo se va produciendo el desarrollo del proyecto de mejora de procesos.

SIMPLe hace más accesible el proceso de mejora y más fácil la transición hacia CMMI permite gestionar de manera centralizada el proyecto de calidad, el cual es dividido en varios ciclos de mejora y varias fases de acuerdo al modelo IDEAL. Otra gran fortaleza de la herramienta es que provee una planificación y administración guiada y estructurada mediante la creación de actividades en base a plantillas, que son un conjunto de actividades, ordenadas secuencialmente y con una duración estimada en días. (SIMPLe-Sistema de Implementación de Mejora de Proceso, 2008)

MEYCOR COBIT MG

El software MEYCOR COBIT MG fue desarrollado por DATASEC. Además de incluir el Modelo de Maduración de COBIT 4.0, contiene otras funcionalidades que facilitan la tarea del gerenciamiento de los recursos de TI, como ser la evaluación de múltiples centros de análisis o el seguimiento de varios períodos de evaluación.

Esto constituye un elemento diferenciador, ya que MEYCOR COBIT MG no solo sirve para realizar un diagnóstico de la situación actual y generar recomendaciones y proyectos, sino que también permite realizar evaluaciones periódicas, las cuales pueden ser comparadas entre sí para medir los avances alcanzados de un período a otro. (Meycor COBIT Guías de Gerenciamiento - Management Guidelines (MG). 2008)

Cobit Audit Advisor

El software Cobit Audit Advisor de Methodware, aplica la estructura COBIT con un proceso abarcativo y consistente, permitiendo a los auditores de sistema de información conducir las auditorías, consolidar y analizar los resultados de auditoría con una sola herramienta dinámica. Esta solución es una aplicación basada en el software Windows de Microsoft que automatiza la estructura COBIT. Todo lo que se

necesita, desde formatos de cuestionario hasta generación de reportes en un poderoso paquete de simple utilización.(CobiT Audit Advisor, 2008)

Las herramientas anteriormente mencionadas SIMPle, MEYCOR COBIT MG, Cobit Audit Advisor de forma general son muy buenas para las necesidades por la cual han sido implementadas .En el caso de Cobit Audit Advisor solo se permite descargar el demo y se enfoca en la auditorías de sistema de información. Mientras que en el caso de MEYCOR COBIT MG solo se puede acceder al folleto con las características del mismo. En SIMPLe, se permite trabajar desde internet y de todas las antes mencionadas es la más completa y la que más se ajusta al trabajo que se quiere realizar, solo le falta entre otros detalles la aplicación de las buenas prácticas que propone COBIT para el control de sus procesos, por la cual se convierte en la herramienta de apoyo para el desarrollo de la aplicación que se desea.

1.3.2 Ámbito nacional

Aunque no existe aún ninguna empresa cubana certificada con CMMI, si son muchas las investigaciones que existen respecto al tema, proporcionando propuesta para su implantación como un programa de mejoras de procesos.

Anterior a este trabajo se han realizado investigaciones en las se ha tratado algunos de los temas en lo que aquí se hace referencia, tal es el caso de la tesis de Maurice Cabrejas y Yosbel Ruiz que presenta como tema: “Guía para la planificación y seguimiento de programas de mejoras en proyectos productivos UCI”.(CABREJAS, 2008) Sé hace énfasis en la planificación, monitoreo y control de un programa de mejoras de procesos para proyectos productivos de la universidad. Enfocan o toman como base el modelo CMMI, y su modelo oficial IDEAL para la planificación, así como también el modelo de control interno COBIT para el seguimiento, por lo cual se toma como guía de apoyo para el presente trabajo.

En Cuba, las experiencias en COBIT son pocas, comenzándose a utilizar en las Auditorías a las Tecnologías de la Información, por el Ministerio de Auditoría y Control (MAC), el Ministerio de la Informática y las Comunicaciones (MIC) y la Corporación CIMEX S.A. La implementación más profunda de COBIT se realizó en la Empresa de Telecomunicaciones de Cuba S.A. (ETECSA), en el ejercicio de las Auditorías Informáticas, desarrollándose soluciones para esta actividad; y más recientemente fue utilizado como marco de referencia en la revisión de las medidas de control para el uso de las redes de datos de dicha empresa.

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

En la UCI Ariesky Sotolongo y Sandro Cruz realizaron una investigación referente a la “Implantación del modelo COBIT en el proceso productivo de la Casa Autoría DVD de la UCI”.(CRUZ, 2007).En el mismo se realiza una propuesta de la implantación de COBIT a las necesidades de la Casa, estableciendo un buen gobierno de TI, que permita a la dirección asegurarse de que las TI agregan valor al negocio y sostienen las estrategias y objetivos de la Casa.

1.4 Desarrollo de una Aplicación Web

En los inicios de la computación, solo existía aplicaciones del tipo consola, posteriormente aparecieron las aplicaciones de escritorio basadas en GUI[1], y como toda evolución, en los años 90 con el nacimiento de Internet fueron surgiendo lo que hoy conocemos como aplicaciones web, que en ese entonces se limitaban a ser simples páginas de texto estático, pero que con el tiempo han ido tomando fuerza hasta llegar a lo que se conoce hoy en día como un sistema informático que los usuarios utilizan accediendo a un servidor web a través de Internet o de una intranet. Cabe mencionar que la principal característica de las aplicaciones web es que son ejecutadas sobre aplicaciones de escritorio que son conocidas como navegadores web, de los cuales los más conocidos son Internet Explorer, Mozilla Firefox y Safari.(RODRÍGUEZ, 2009)

Una de las principales ventajas que presentan las aplicaciones web ante las aplicaciones de escritorio o consola es el hecho de que no dependen de ningún sistema operativo ni configuración de hardware específica; para su ejecución simplemente basta con teclear su dirección URL[2] en cualquier navegador web. De igual manera sus actualizaciones se hacen de una manera muy sencilla, sin necesidad de hacer descargas, instalaciones o comprar físicamente el producto. Esta flexibilidad ha sido uno de los principales motivos por los que cada vez son más utilizadas para una creciente diversidad de tareas.

---

[1] **GUI *(Graphical User Interface,*** *en español,* ***Interfaz Gráfica de Usuario):*** es un tipo de interfaz de usuario que utiliza un conjunto imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

[2] **URL *(Uniform Resource Locator,*** *es decir,* ***Localizador Uniforme de Recursos):*** se refiere a la dirección única que identifica a una página web en Internet.

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

Debido a las ventajas que ofrecen las aplicaciones Web, la solución del problema planteado en esta investigación y el cumplimiento del objetivo general, será resuelto mediante el desarrollo de una aplicación web.

1.5 Conclusiones

En este capítulo se abordaron los elementos teóricos que sustentan el problema científico y los objetivos del trabajo. Se reafirmó que el modelo CMMI es el más eficaz para llevar a cabo un programa de mejora de procesos, conjunto con su método de implantación IDEAL. Además se determinó la necesidad de monitorear y controlar mediante COBIT el proceso de implantación. Y por último se reflejaron algunas de las soluciones existentes de acuerdo a la base del presente trabajo.

Lo que conlleva a la realización de una aplicación web, que permita monitorear y controlar mediante el marco de referencia COBIT el proceso guía de IDEAL como método para la implantación de las buenas prácticas propuestas por el modelo base CMMI.

CAPÍTULO 2: Tendencias y tecnologías actuales a desarrollar

2.1 Introducción

En este capítulo se realiza un análisis detallado de las principales tendencias y tecnologías que se ajustan a la solución propuesta. Se hace un estudio de los lenguajes de programación más apropiados para el desarrollo de la aplicación, el framework y Sistema de Gestión de Bases de Datos (SGBD). Además se considera la metodología a utilizar para el análisis y diseño del sistema teniendo en cuenta las facilidades que puede aportar al trabajo.

2.2 Metodologías de desarrollo de Software

Actualmente a nivel mundial, en dependencia del tiempo de vida y la complejidad del proyecto que se vaya a desarrollar se proponen diferentes metodologías. Una metodología es el conjunto de técnicas y procedimientos que permiten conocer los elementos necesarios para definir un proyecto de software, es la base para la edificación de un producto de este tipo. Esencialmente, sirve para aumentar la "calidad" del software y controlar de manera transparente todo el proceso de desarrollo. Si se quiere que un proyecto sea escalable y flexible a los cambios es lógico pensar que para lograr ese propósito se necesite tomar en cuenta una de las muchas metodologías para el proceso de desarrollo de software que existen.

Las metodologías dadas sus características, se enmarcan en dos grandes grupos, los llamados "métodos pesados o tradicionales " y los "métodos ligeros o ágiles". La diferencia más notable entre estos dos grupos es que mientras los métodos pesados intentan obtener los resultados apoyándose principalmente en la documentación ordenada, los métodos ágiles tienen como base de sus resultados la comunicación e interacción directa con todos los usuarios involucrados en el proceso. Ver comparación entre estos dos grandes grupo. (Ver_Anexo II: Comparación entre las metodologías “ágiles” y “no ágiles”)

2.2.1 Metodologías Ágiles o Ligeras

Programación Extrema (XP)

CAPÍTULO 2
TENDENCIAS Y TECNOLOGÍAS

La metodología XP según Kent Beck, es ligera, flexible, predecible y para equipos de desarrollo pequeños y medianos. Se basa en cuatro valores fundamentales que son la simplicidad, la comunicación, el reciclado continuo de código o retroalimentación (refactoring) y la tenacidad que debe tener el programador. También sigue principios de soluciones rápidas, simples, con cambios incrementales, flexibilidad para adoptar cambios y trabajo con calidad. (BECK, 1999)

Esta metodología se rige además por doce prácticas entre las que se destacan: el juego de planificación, entregas pequeñas, diseños simples, probar constantemente, retroalimentación, programación en parejas, integración continua y cliente como parte del equipo. Lo que indica que es una metodología orientada al desarrollo de un software, no realiza la gestión, ni el modelado. (BECK, 1999)

El ciclo de desarrollo consiste en los siguientes pasos:
1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto, las cuales se resumen en las siguientes cuatro: Planificación, Diseño, Implementación y Prueba.

Scrum

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos y se basa en el principio ágil de desarrollo iterativo e

incremental. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

Crystal

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).

2.2.2 Metodologías Tradicionales o Robustas

Proceso Unificado de Desarrollo (RUP)

RUP como metodología ,es un proceso de desarrollo de software que define *quién* está haciendo *qué*, *cuándo* y *cómo* alcanzar un determinado objetivo, además es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto .

Para la realización del producto software la metodología RUP divide el trabajo en 9 flujos de trabajo y 4 fases. Cada una de estas fases es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Tiene como características fundamentales ser iterativo e incremental, dirigido por casos de usos y centrado en la arquitectura.

El Proceso Unificado de Desarrollo además contiene las mejores prácticas de la ingeniería de software que están siendo utilizadas actualmente para el desarrollo exitoso de un proceso de desarrollo de

software, desplegando las mismas ofrece al equipo de desarrollo un número significante de ventajas pues lo provee de guías, plantillas y herramientas para una mejor forma de trabajo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Microsoft Solution Framework (MSF)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.
MSF tiene las siguientes características:

- **Adaptable**: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- **Escalable**: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- **Flexible**: es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica**: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

Las metodologías robustas RUP y MSF no se ajustan al presente trabajo ,debido que para el desarrollo del mismo se estima de muy poco tiempo para dar cumplimiento al objetivo general propuesto, y el equipo de desarrollo se encuentra integrado por una sola persona, además de ser un proyecto pequeño, en el cual el entorno de desarrollo es muy cambiante, por lo que es necesario el uso de una metodología ágil cuya desarrollo incremental del software se realice mediante iteraciones muy cortas, contenga pocos

artefactos y mantenga la calidad en desarrollo. Son por estas características que de las metodologías ágiles analizadas se escoge XP para el desarrollo de la solución propuesta.

2.2.3 Fundamentación de la metodología de desarrollo de software escogida

Analizando las características de los métodos anteriormente expuestos se llegó a la conclusión de que la más aceptada para la aplicación que se desea obtener es la metodología ágil XP, debido a las siguientes características:

- Es una metodología de desarrollo ágil.
- Está pensada para equipos de desarrollos pequeños.
- Se especifican pocos artefactos eliminando el "papeleo" innecesario y dedicando más tiempo a la implementación.
- Permite la entrega pequeña de un producto funcional al finalizar cada iteración.
- Da la posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente.
- Flexibilidad para adoptar cambios.
- Tiene un alcance acotado y viable.
- Se aplica en equipos integrados y comprometidos con el proyecto y que se auto administran.

2.3 Herramientas CASE para la modelación UML

UML[3] es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

[3] **UML** ***(Unified Modeling Language,*** *en español,* ***Lenguaje Unificado de Modelado):*** es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

CAPÍTULO 2
TENDENCIAS Y TECNOLOGÍAS

Las herramientas CASE[4] tienen como objetivo automatizar los aspectos claves del proceso de desarrollo de sistemas desde su inicio hasta el final. Las consecuencias de esta automatización se observan en el aumento de la productividad por parte de los desarrolladores de sistema, en la mejora de la comunicación entre usuarios y especialistas, el sostenimiento de una estandarización en las actividades de un proyecto y en las facilidades que brindan para el mantenimiento del sistema. Dentro de las Herramientas CASE multiplataforma utilizadas hoy en día para la modelación de los diagramas UML se encuentran ArgoUML y Visual Paradigm. De estas dos herramientas se brinda una pequeña descripción a continuación.

2.3.1 ArgoUML

ArgoUML es una herramienta para el modelado de proyectos por medio de UML creado por Jason Robbins y su equipo de colaboradores de la Universidad de California. ArgoUML está codificado en Java, por lo que basta tener instalado el JRE[5] para ejecutarlo. Esto lo hace independiente de la plataforma. Es una herramienta de código abierto muy fácil de usar, la licencia que presenta es BSD[6]. Soporta diversos tipos de diagramas de UML:

- Diagrama de Casos de Uso.
- Diagrama de Clases.
- Diagrama de Secuencia.
- Diagrama de Colaboración.
- Diagrama de Estados.
- Diagrama de Actividades.

---

[4] **CASE *(Computer Aided Software Engineering,*** *en español,* ***Ingeniería de Software Asistida por Ordenador):*** son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software. Ayudan en todos los aspectos del ciclo de vida de desarrollo del software.

[5] **JRE *(Java Runtime Environment,*** *en español,* ***Entorno en Tiempo de Ejecución Java):*** corresponde con un conjunto de utilidades que permite la ejecución de programas java sobre todas las plataformas soportadas.

[6] **BSD *(Berkeley Software Distribution,*** *en español,* ***Distribución de Software Berkeley):*** es una licencia de software libre permisiva. Permite el uso del código fuente en software libre.

- Diagrama de Despliegue.

2.3.2 Visual Paradigm

Esta herramienta acelera el desarrollo de aplicaciones, ya que sirve de puente visual entre arquitectos, analistas y diseñadores de sistema, haciendo el trabajo más fácil y dinámico. La herramienta también automatiza tareas tediosas que pueden distraer a diseñadores del desarrollo. Ofrece los siguientes beneficios:

- Varios idiomas.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- La navegación intuitiva entre el código y el modelo visual.
- Poderoso generador de informes PDF/HTML.
- Tiempo real en la demanda.
- Un ambiente modelador visual superior.
- Sofisticado diseño de diagramas.

2.3.3 Fundamentación de la herramienta CASE para la modelación UML escogido

Entre las herramientas anteriormente expuestas fue escogido para utilizar Visual Paradigm al contar con una mayor aceptación entre los desarrolladores de aplicaciones informáticas del mundo. Brinda gran cantidad de funcionalidades en la construcción de los Diagramas, así como un desarrollo más completo.

2.4 Sistema de Gestión de Bases de Datos (SGBD)

Una base de datos es una colección de datos interrelacionados y almacenados en un soporte informático. Se les llama Sistema Gestor de Bases de Datos (SGBD) a los programas que posibilitan acceder a esas bases de datos y manipularlas, permitiendo concurrencia y persistencia. La concurrencia se refiere a la capacidad de los SGBD para gestionar a múltiples usuarios interactuando a la vez desde diferentes puntos de vista sobre los mismos. La persistencia está dada por la posibilidad de conservación de los datos después de terminado el proceso que los creó.

CAPÍTULO 2
TENDENCIAS Y TECNOLOGÍAS

Stanley Su en su libro *"Database Computers: Principles, Architectures and Techniques"* define un Sistema de Gestión de Bases de Datos como: "paquete de software que proporciona todas las facilidades para la creación, recuperación, manipulación y mantenimiento de bases de datos, asegurando su integridad, confidencialidad y seguridad ".(SU, 1998)

Entre los SGBD más populares para los sistemas operativos GNU/Linux[7] se encuentran MySQL y PostgreSQL, de los cuales se brinda una pequeña descripción a continuación.

2.4.1 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL[8] de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. Aunque MySQL es software libre, posee una versión comercial, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL, a la hora de no tener la libertad de compartir y modificar dicho software. Las principales características de este gestor de bases de datos son las siguientes.

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de APIs[9] en gran cantidad de lenguajes (C, C++, Java, PHP, etc.).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.

---

[7] **GNU/Linux *(distribuciones Linux):*** Es una implementación de libre distribución UNIX para computadoras personales (PC), servidores y estaciones de trabajo. Es, a simple vista, un Sistema Operativo.

[8] **GPL *(General Public License,*** *en español,* ***Licencia Pública General):*** Es una licencia creada por la Fundación para el Software Libre, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

[9] **APIs *(Application Programming Interface,*** *en español,* ***Interfaz de Programación de Aplicaciones):*** es el conjunto de funciones y procedimientos(o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado otro software como una capa de abstracción.

- Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

2.4.2 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. PostgreSQL es una derivación libre (código abierto) de este proyecto. Es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos. A continuación se enumeran las principales características de este gestor de bases de datos:

- Implementación del estándar SQL92/SQL99.
- Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs[10] y la cantidad de RAM[11].
- Tiene mejor soporte para triggers y procedimientos en el servidor.
- Incluye herencia entre tablas, por lo que se le incluye entre los gestores objeto-relacionales.
- Implementa el uso de rollback's, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz.
- Numerosos tipos de datos, posibilidades de definir nuevos tipos
- Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido, …)
- Replicación asíncrona.
- Copias de seguridad en caliente.
- APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP y muchos otros lenguajes.

---

[10] **CPUs *(Central Processing Unit,*** *en español,* ***Unidad Central de Procesamiento),*** **o, simplemente, el procesador**: es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de la computadora.

[11] **RAM *(Random Access Memory,*** *en español,* ***Memoria de Acceso Aleatorio):*** es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados.

- Completa documentación.

2.4.3 Fundamentación del SGBD escogido

Se decide utilizar PostgreSQL en la aplicación que se quiere obtener, pues sus ventajas y prestaciones son amplias y ajustables al objetivo propuesto. Es un sistema potente, multiplataforma y de código abierto lo que permite el constante desarrollo y perfeccionamiento de sus funcionalidades; está publicado bajo licencia BSD, que pertenece al grupo de licencias de software libre. Además se tiene en cuenta que MySQL puede tener un futuro incierto con su valor de código abierto, al ser comprado por SUN Microsystems[12], la cual ya ha enseñado su visión futurista con el producto, al plantear que algunas partes de MySQL sólo serían ofrecidas a clientes corporativos que paguen una suscripción por el sistema de base de datos, y que éstos componentes sólo estarían disponibles como código abierto cerrado.

2.5 Lenguajes de programación

Un lenguaje de programación es un lenguaje que es utilizado para controlar el comportamiento físico y lógico de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.
Una característica relevante de los lenguajes de programación es precisamente que permite a más de un programador especificar sobre qué datos debe operar una computadora.

Para la realización de la herramienta serán utilizados varios de los lenguajes de programación existentes, de acuerdo al fin para el cual fueron construidos. A continuación se brinda una breve descripción de cada uno de ellos.

2.5.1 Lenguaje de programación del lado del cliente

Los lenguajes de programación del lado cliente son totalmente independiente del servidor, están insertados en la página Web y son interpretados y ejecutados por el navegador, de modo que cuando el navegador recibe una página web, interpreta y da formato al contenido de la página y entra el código de los scripts al programa intérprete correspondiente, que deberá haber sido instalado en el navegador como

[12] **Sun Microsystems:** es una empresa informática de Silicon Valley, fabricante de semiconductores y software.

un plugin. Para la implementación de la aplicación se deben tener en cuenta alguna de ellos, los cuales son referenciados a continuación.

2.5.1.1 HTML

Marcas de *Hipertexto* (HTML, siglas en inglés) es el lenguaje que todos los programas navegadores usan para presentar información en la *World Wide Web* (WWW). HTML no es más que una serie de etiquetas que se utilizan para definir la forma o estilo que queremos aplicar a nuestro documento.

Entre otras cosas, el manejo de estas etiquetas permitirá:

- Definir la estructura lógica del documento HTML.
- Aplicar distintos estilos al texto.
- La inclusión de imágenes y ficheros multimedia (gráficos, vídeo, audio).

Todas estas características ofrecen importantes ventajas para el desarrollo de la aplicación en la visualización de los contenidos.

2.5.1.2 Java Script

Se conoce a Java Script como un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas, que no es más que pequeños programitas encargados de realizar acciones dentro del ámbito de una página web.

Las características más importantes del lenguaje Java Script son:

- Es un lenguaje interpretado, es decir, no requiere compilación. El navegador del cliente es el encargado de interpretar las instrucciones Java Script y ejecutarlas.
- Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez.
- Permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.
- Permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc.

Al ser un lenguaje de programación que se ejecuta en el lado cliente, posibilita la reducción de carga del servidor, siendo esta una característica muy importante para la aplicación que se quiere desarrollar. Además la vinculación estrecha que existe entre este lenguaje y AJAX (JavaScript asíncrono y XML) hace necesario su utilización.

2.5.2 Lenguaje de programación del lado del servidor

Los lenguajes del lado servidor son independientes del navegador utilizado, estos son reconocidos, ejecutados e interpretados por el propio servidor y se envían al cliente en un formato comprensible para él. Entre los más usados actualmente se encuentran Java mediante las Páginas de Servidor Java (JSP, siglas en inglés) y PHP de los que se brinda una breve descripción a continuación.

2.5.2.1 Java

Java es un lenguaje de programación orientado a objetos desarrollado por la empresa SUN Microsystems en 1995. Este lenguaje define una JVM[13] independiente de la plataforma donde se ejecuta, que procesa programas llamados Applets, descargados desde el servidor Web. En la actualidad existen varias formas de usar Java dentro de un servidor Web, mediante Servlets o JSPs (Java Server Pages). Los Servlets son componentes del servidor. Estos componentes pueden ser ejecutados en cualquier plataforma o en cualquier servidor debido a la tecnología Java que se usa para implementarlos. Los Servlets incrementan la funcionalidad de una aplicación Web y cargan de forma dinámica por el entorno de ejecución Java del servidor cuando se necesitan.

[13] **JVM *(Java Virtual Machine,*** *en español,* ***Máquina Virtual Java):*** es un programa ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

JSP es una tecnología similar a los Servlets que ofrece una conveniente forma de agregar contenido dinámico a una página Web por utilizar código escrito en Java dentro de la página utilizando tags[14] especiales que son procesados por el servidor Web antes de enviarlos al cliente.

Actualmente Java se utiliza en un amplio abanico de posibilidades y casi cualquier cosa que se puede hacer en cualquier lenguaje se puede hacer también en Java. Con Java podemos programar páginas web dinámicas, con accesos a bases de datos, utilizando XML[15], con cualquier tipo de conexión de red entre cualquier sistema. En general, cualquier aplicación que deseemos hacer con acceso a través de la web se puede hacer utilizando Java.

2.5.2.2 PHP

PHP es el acrónimo de Hipertext Preprocesor, es un lenguaje interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, embebido en páginas HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. Este lenguaje dispone de gran cantidad de características que lo convierten en el lenguaje de programación del lado del servidor ideal para usar.

- Es independiente de plataforma, puesto que existe un módulo de PHP para casi cualquier servidor web.
- Está basado en software libre, lo que permite distribuirlo libremente y agregar nuevas funcionalidades si se necesita. Además no hay que pagar licencias por su uso.
- Soporta programación orientada a objetos, sobre todo con las mejoras introducidas en PHP 5, y también la herencia, aunque no múltiple.
- Está dotado de extensa librería de funciones, por lo que está preparado para realizar muchos tipos de aplicaciones web.

---

[14] **Tags (etiqueta o baliza):** es una marca con tipo que delimita una región en los lenguajes basados en XML.

[15] **XML *(Extensibe Markup Languaje,*** *en español,* ***Lenguaje Marcado Extensible):*** es un lenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

- PHP ofrece soporte para un amplio abanico de bases de datos: dBase, DB2, mSQL, Oracle, PostgreSQL, MySQL, etc.

En resumen PHP es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.

2.5.2.3 Fundamentación del lenguaje de programación del lado del servidor escogido.

Como lenguaje de programación para el lado del servidor se escogió PHP, al garantizar con sus características, obtener un producto que se pueda montar en cualquier sistema, ya que desarrollado el sitio en este lenguaje permite portar el sitio de un sistema a otro sin prácticamente ningún trabajo. Agilidad en la elaboración del producto al contar con librerías donde están implementadas las funciones necesarias para dar solución a gran cantidad de funcionalidades del sistema, compatibilidad con PostgreSQL sistema de gestión de bases de datos que se utilizará, los cuales constituyen requisitos imprescindibles para obtener la aplicación.

2.6 Frameworks

En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, librerías y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Entre los *Frameworks* que más se usan para las aplicaciones Web donde se utilice el lenguaje de programación PHP se encuentran CakePHP, Zend Framework y Symfony de los cuales se brinda a continuación una pequeña descripción.

2.6.1 CakePHP

CakePHP es un framework de desarrollo rápido de aplicaciones de código abierto en PHP, inspirado en Ruby on Rails[16]. Es un framework para la construcción de sitios Web que utilizan una base de datos como fuente de recursos, posee una infraestructura que tiene como finalidad permitir el desarrollo de manera ágil y estructurada, sin perder flexibilidad. Entre las características más destacables de CakePHP se incluyen:

- Incorpora el patrón Modelo Vista Controlador (MVC) y orientada a objetos.
- Compatible con PHP4 y PHP5.
- Operaciones básicas en base de datos (creación, obtención, actualización y borrado).
- Generación de plantillas de manera rápida y flexible usando la sintaxis de PHP y con asistentes o *helpers*.
- Incorporación de asistentes de construcción de vistas: para la automatización de la generación de código en AJAX, Java Script, formularios HTML, entre otros.
- Estructura de aplicaciones (Application Scaffolding): permite al programador hacer uso de un conjunto de convenciones aplicables a la estructura de la base de datos de la aplicación y el framework se encarga de generar el código para la interacción a lo largo de todas las capas de la aplicación.

2.6.2 Zend Framework

El Zend Framework es un intento dirigido por la compañía responsable del desarrollo del lenguaje PHP, y mantenido por una comunidad de voluntarios. Se trata de un framework para desarrollo de aplicaciones Web y servicios Web con PHP, te brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. Dentro de sus principales características se encuentran:

- Trabaja con MVC (Modelo Vista Controlador)
- Cuenta con módulos para manejar archivos PDF, canales RSS, Servicios Web, etc.

[16] **Ruby on Rails (ROR):** es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC).

- Incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Completa documentación y test de alta calidad.
- Robustas clases para autenticación y filtrado de entrada.

A pesar de que este framework está dirigido por la compañía responsable del desarrollo de PHP, aún no consigue niveles de eficacia y adopción similares a los Frameworks en PHP.

2.6.3 Symfony

Symfony es diseñado con el objetivo de optimizar la creación de las aplicaciones web, con el uso de sus características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Posee una librería de clases que permiten reducir el tiempo de desarrollo. Symfony está desarrollado en PHP5, se puede utilizar en plataforma Unix, Linux y Windows. Requiere de una instalación, configuración y líneas de comando, e incorpora el patrón MVC. Soporta AJAX, plantillas y un gran número de bases de datos .Su creador, Fabien Potencier, toma prestadas las mejores ideas de cualquier *framework* (da igual si está escrito en *Perl*, *Python* o *Ruby*) y las adapta para Symfony, además de añadir las suyas propias.

Al centralizar las características de otros Frameworks en uno solo, hacen que el uso de este framework posibilite la implementación de una mayor cantidad de funcionalidades de forma sencilla, en las aplicaciones que se desarrollen.

Entre las características generales del Symfony se pueden citar:

- Fácil de instalar y configurar: ha sido probado con éxito en plataformas Windows y derivadas de Unix.
- Independiente del manejador de base de datos: utiliza Propel, una capa de abstracción que le permite interactuar con varias bases de datos.
- Simple de usar: y al mismo tiempo lo suficientemente flexible para adaptarse a escenarios complejos.

- Basado en la premisa de "convención sobre configuración": el desarrollador sólo necesita configurar aquellos aspectos sobre los cuales no hay una tendencia definida.
- Cumple con la mayoría de las mejores prácticas en diseño Web y patrones de diseño.
- Fácil de extender: permitiendo la integración con otras librerías.
- Incorpora herramientas que facilitan la prueba y depuración de aplicaciones: como unidades de generación de código, pruebas del funcionamiento del *framework*, panel de depuración, interfaz por línea de comandos y configuración en tiempo real.
- Uso de plantillas: las cuales pueden ser elaboradas por diseñadores de páginas Web que desconocen el resto de detalles técnicos del *framework*.
- Validación y regeneración automática de formularios: lo que asegura una buena calidad de los datos en la base de datos y una mejor experiencia de usuario.
- Verificación de la salida enviada por la aplicación: ofrece una protección frente a ataques por datos corruptos.

2.6.4 Fundamentación del Framework escogido

Los *Frameworks* anteriormente mencionados poseen de forma general grandes mejoras para poder optimizar el trabajo de los desarrolladores en el lenguaje PHP. A la hora de escoger uno de ellos es necesario ver cual tiene mayor cantidad de características que se ajusten específicamente, a la aplicación que se quiere obtener. Teniendo en cuenta lo anteriormente planteado se decide utilizar Symfony, al constituir un *framework* muy maduro, desarrollado completamente con PHP5, que permitiría lograr migrar de *SGBD* sin hacer cambios en el código fuente de la aplicación, ya que es compatible con la mayoría de los gestores de bases de datos. Además se puede ejecutar tanto en plataformas Unix como en plataforma Windows. Por último y no menos importante es la maravillosa documentación que posee (sobre todo en español).

2.7 Entorno Integrado de Desarrollo (IDE)

Un Entrono de Desarrollo Integrado (IDE) se refiere a las interfaces visuales de programación que se usan para interactuar con el compilador y/o debugger de un lenguaje de programación determinado de forma amigable para el desarrollador. Por lo general consta de un editor de texto, con opciones para tratar el

código que se edita y para mandar comandos al compilador y al debugger sobre acciones a realizar sobre el código que se está editando.

Para el desarrollo del sistema fue escogido como entorno integrado de desarrollo el **Eclipse**. Esta plataforma de desarrollo fue desarrollada originalmente por IBM[17], actualmente es desarrollada por la Fundación Eclipse, una organización independiente sin ánimos de lucro que fomenta una comunidad de código abierto. Entre las principales características que se tuvieron en cuenta para escoger la plataforma en la que se va a desarrollar están:

- Presenta un excelente completamiento de código, lo que permite al programador no tener que memorizar todas las funciones que utilice el lenguaje de programación que esté seleccionado o el framework escogido para el desarrollo de una aplicación.
- Es una plataforma de desarrollo muy estable.
- Permite que se incorporen un número considerable de plugin en dependencia del tipo de aplicación que se quiera implementar.
- Es completamente libre, lo cual es indispensable para la realización del sistema debido a solicitud del cliente.
- Presenta la opción de ver qué tipo de objeto o variable devuelve un método.
- Ayuda la escritura de los ficheros HTML y CSS[18].
- Permite la integración con el framework de desarrollo escogido para la implementación de la aplicación.

Todas las características mencionadas fueron obtenidas a partir de la experiencia de desarrolladores con este sistema. De esta manera quedan explicadas las principales características del entorno de desarrollo integrado en el que se va a desarrollar.

---

[17] **IBM *(International Business Machines, conocida coloquialmente como el Gigante Azul):*** es una empresa que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

[18] **CSS *(Cascading Style Sheets, en** *español, Hojas* ***de Estilo en Cascada):*** son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

CAPÍTULO 2
TENDENCIAS Y TECNOLOGÍAS

2.8 AJAX

El término AJAX se acuñó por primera vez en el artículo “Ajax: A New Approach to Web Applications” publicado por Jesse James Garrett el 18 de Febrero de 2005. Hasta ese momento, no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación web que estaba apareciendo.

El término AJAX es un acrónimo de Asynchronous JavaScript + XML, que se puede traducir como “JavaScript asíncrono + XML”.

El artículo define AJAX de la siguiente forma:

> “Ajax no es una tecnología en sí mismo. En realidad, se trata de la unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes.” (PÉREZ, 2007)

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM[19], para la interacción y manipulación dinámica de la presentación.
- XML, XSLT[20] y JSON[21], para el intercambio y la manipulación de información.

---

[19] **DOM (*Document Object Model*, en español, *Modelo en Objetos para la representación de Documentos*):** es una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.

[20] **XSLT (*Extensible Stylesheet Language Transformations* o *Transformaciones XSL*)**: es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML.

[21] **JSON (acrónimo de *"JavaScript Object Notation"*):** es un formato ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

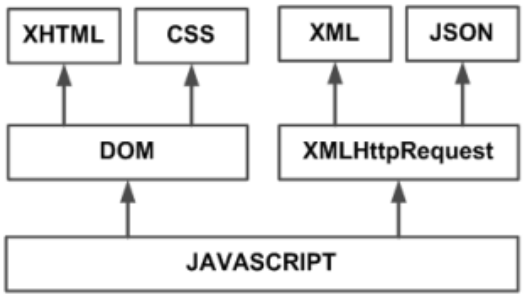


Figura 7: Tecnologías agrupadas bajo el concepto de AJAX

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

2.9 Conclusiones

En este capítulo se ha realizado un análisis de las tecnologías y tendencias actuales sobre las que se trabajará para obtener la aplicación que se desea, profundizándose en el conocimiento de conceptos necesarios para la comprensión del trabajo. Se dio a conocer cómo influyen todos estos elementos en la confección del mismo y las características que hacen que sean escogidos, llegando a la conclusión de que:

- Él sistema será implementado sobre el lenguaje de programación PHP, lo que resulta provechoso a la hora de realizar una migración de Sistemas Operativos.

- El framework utilizado para el desarrollo será Symfony, uno de los más utilizados para el desarrollo en PHP en el mundo, visto así en el estudio del arte realizado. Notable son sus funcionalidades dedicadas a la seguridad y a la aplicación del patrón Modelo-Vista-Controlador (MVC).

- El gestor de bases de datos a utilizar PostgresSQL es hoy por hoy una herramienta muy renombrada en el área de Software Libre para el tratamiento de grandes volúmenes de datos.

- El entorno de desarrollo Eclipse tiene una gran acogida entre los desarrolladores de todo el mundo y entre sus características más representativas está su portabilidad e integración con plugins y frameworks potentes.

- La metodología de desarrollo XP, brinda los mejores elementos para dar solución al problema planteado, y cumplimiento al objetivo general, siendo esta la metodología ágil más popular en la actualidad .El Visual Paradigm es una herramienta modelado con un entorno de trabajo amigable y robusto.

CAPÍTULO 3: Solución propuesta

3.1 Introducción

En el presente capítulo se describe la solución propuesta y se planifica su desarrollo, haciendo hincapié en las historias de usuario, planificación de las iteraciones y el plan de entregas. Además se describe la arquitectura base de la aplicación y por último se ilustra la interfaz gráfica del sistema.

3.2 Descripción de la solución propuesta

Para lograr una optimización en la implantación de CMMI como modelo propuesto para la mejora de procesos mediante IDEAL como su método de implantación, se propone una herramienta que se encargue de automatizar el monitoreo y control de la implantación de este proyecto de mejora de procesos que sea capaz de medir cuán bien se está llevando a cabo este proyecto en la universidad, la cual sería la propuesta de solución para este trabajo.

El Sistema de Implantación de un Proyecto de Mejora (SIMPUci) se desarrollará de manera que permita gestionar de forma centralizada un proyecto de mejora en donde se puede registrar fácilmente mediante actividades y artefactos, cómo se va produciendo el desarrollo del mismo.

Se divide el proyecto en varios ciclos de mejora y varias fases de acuerdo al modelo IDEAL y se provee una planificación y administración guiada de las actividades que se realizan de acuerdo a este modelo. Además se documentan y activan las áreas de procesos por ciclo y se monitorea y controla el avance del proyecto.

Los **objetivos específicos** de esta herramienta son:

- Automatizar la gestión de tareas y procesos específicos dentro del proyecto de mejora.
- Proporcionar una planificación y administración guiada y centralizada del proyecto de mejora.

- Proporcionar visibilidad durante todo el proyecto para los diferentes involucrados, permitiendo el seguimiento del proyecto.
- Gestionar y centralizar artefactos generados a través del proceso de mejora.

Esta herramienta permitirá:

- Gestionar de manera centralizada el proyecto de calidad, sus fases, actividades e involucrados.
- Correlacionar las etapas identificadas en el proyecto con las fases del modelo guía IDEAL.
- Dar soporte a la documentación de los procesos de la organización, relacionándolos con las áreas de proceso del modelo CMMI.
- Almacenar los artefactos generados de manera centralizada.
- El sistema permitirá acceder a toda la información tanto del ciclo actual que será identificado como activo y abierto, y la de los ciclos pasados que serán identificados como inactivos y cerrados. La información de los ciclos inactivos no podrá ser modificada, sólo visualizada.
- Monitorear y controlar el avance del proyecto, mediante el cálculo de métricas que permitan reflejar el estado actual del proyecto.

3.2.1 Modelo Conceptual

Un modelo conceptual explica los conceptos más significativos en un dominio del problema, identificando atributos y asociaciones.

En la siguiente figura se muestran los conceptos identificados para el sistema, y sus atributos.

CAPÍTULO 3
SOLUCIÓN PROPUESTA

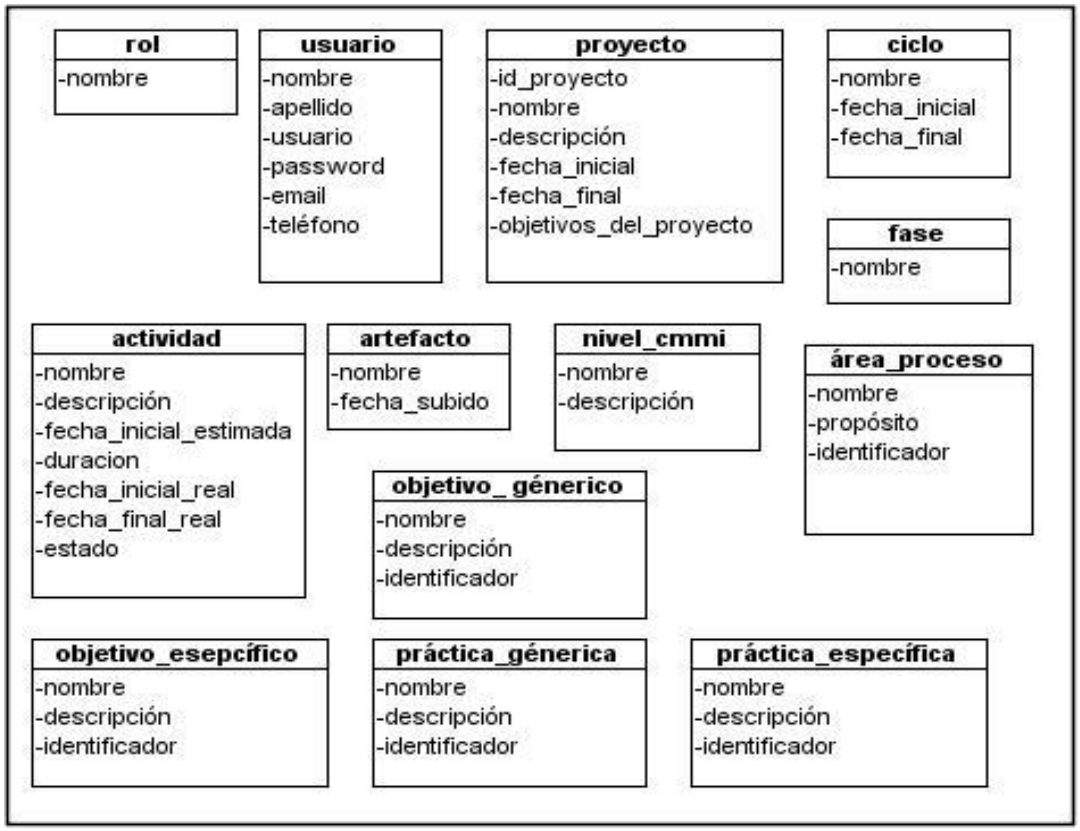


Figura 8: Conceptos y atributos del sistema

Un usuario del sistema cumple uno o más roles y gestiona uno o más proyectos, y viceversa. Cada proyecto está compuesto de uno o varios ciclos, puede tener riesgos y artefactos. El ciclo tiene 5 fases que son: iniciar, diagnosticar, establecer, actuar y aprender, y contienen una o varias actividades. Una o más áreas de proceso son gestionadas en uno o más ciclos. Un área de proceso se encuentra en un nivel de CMMI y debe cumplir un objetivo genérico y varios objetivos específicos, los cuales se cumplen mediante prácticas genéricas y prácticas específicas, respectivamente.

En la siguiente figura se muestra el modelo conceptual obtenido en base a los requisitos, conceptos y definiciones del sistema.

CAPÍTULO 3
SOLUCIÓN PROPUESTA

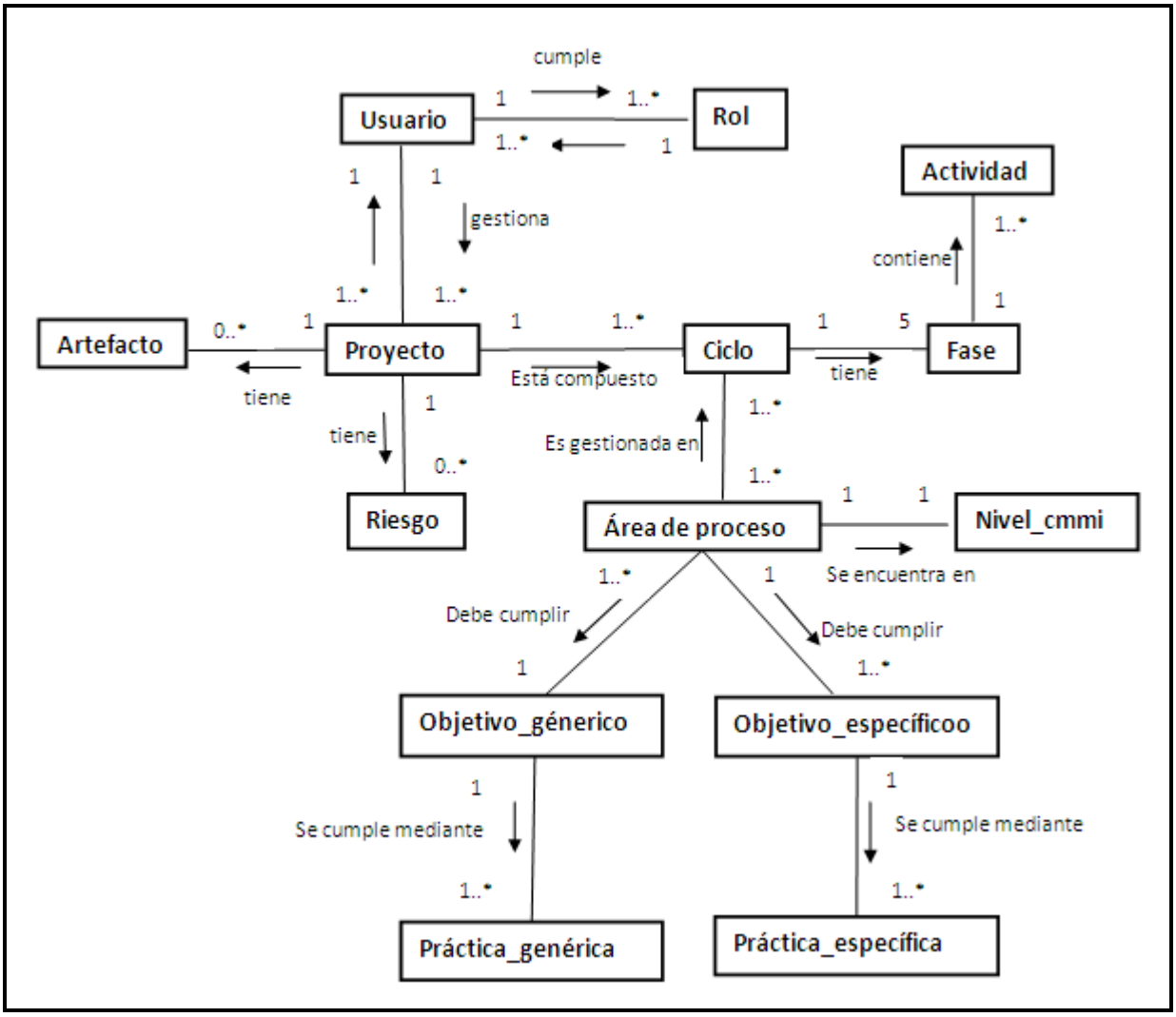


Figura 9: Modelo Conceptual del Sistema

3.2.2 Actores del Sistema

Los actores identificados que interactúan con el sistema son:

Actor	**Responsabilidad**
Jefe de Proyecto	El jefe de proyecto administrará y coordinará el proyecto de mejora de calidad de procesos. Maneja toda la información relacionada y generada, así como coordina con los diferentes involucrados del

	proyecto. El jefe de proyecto será el que administre toda la aplicación y tenga todos los permisos sobre el sistema.
TWG's (Grupo de Trabajo Técnico)	Implementa las actividades de proyecto, específicamente lo referente al área de proceso asignada a su equipo de trabajo. Podrá administrar las actividades y artefactos relacionados al área de proceso designada.
SEPG (Grupo de Proceso de Ingeniería de Software)	Facilita y coordina el proyecto, guía las actividades mas no implementa. Podrá consultar y visualizar toda la documentación; además podrá añadir y visualizar lecciones aprendidas del proyecto.
MSG (Grupo de Gestión de Manejo)	Provee recursos, monitorea avance del proyecto, provee guía y acciones correctivas cuando es necesario. Podrá consultar y visualizar toda la documentación; además podrá añadir y visualizar lecciones aprendidas del proyecto.

En la aplicación el jefe de proyecto, es el máximo líder y será el encargado de administrar todo el proyecto, es el único que podrá modificar, mientras que los restantes actores (TWG´s, SEPG, y MSG) luego de que sean agregados por el líder, solo accederán al sistema para saber cuáles son las tareas o actividades que les corresponde desempeñar.

3.2.3 Funcionalidades del sistema

Las funcionalidades son aquellos requisitos que, desde el punto de vista de las necesidades del usuario, debe cumplir el sistema. A continuación se relacionan las que se establecieron para la aplicación.

F1 Ingresar al sistema

1.1 Autenticar usuario

1.1.1 Usuario

1.1.2 Contraseña

F2 Crear proyecto de mejora

F3 Gestionar proyecto de mejora de calidad de procesos

CAPÍTULO 3
SOLUCIÓN PROPUESTA

3.1 Gestionar usuarios

3.1.1 Crear usuario del sistema

3.1.1.1 Usuario

3.1.1.2 Contraseña

3.1.1.3 Rol

3.1.2 Eliminar usuario del sistema

3.1.3 Asignar roles a los usuarios

3.1.3.1 TWG´s

3.1.3.1.1 Visualizar las actividades que le corresponde desempeñar

3.1.3.1.2 Consultar y visualizar toda la documentación

3.1.3.2 SEPG

3.1.3.2.1 Visualizar las actividades que le corresponde desempeñar

3.1.3.2.2 Consultar y visualizar toda la documentación

3.1.3.3 MSEG

3.1.3.3.1 Visualizar las actividades que le corresponde desempeñar

3.1.3.3.2 Consultar y visualizar toda la documentación

3.2 Gestionar artefactos

3.2.1 Subir artefactos

3.2.2 Abrir artefactos

3.2.3 Eliminar artefactos

3.3 Cerrar ciclo actual

3.4 Editar proyecto

3.5 Eliminar proyecto

F4 Gestionar actividades

4.1 Crear actividad

4.2 Editar actividad

4.3 Eliminar actividad

4.4 Ver detalles

F5 Activar y documentar área de procesos

5.1 Activar área

5.2 Documentación del área

F6 Registrar Riesgos

6.1 Editar

6.2 Eliminar

6.3 Ver detalles

6.3.1 Mitigar

F7 Controlar avance del proyecto

7.1 Métricas

3.2.3.1 Diagrama de funcionalidades del jefe de proyecto en el sistema

Se muestra a continuación un diagrama de las funcionalidades que debe ejercer el jefe de proyecto en el sistema.

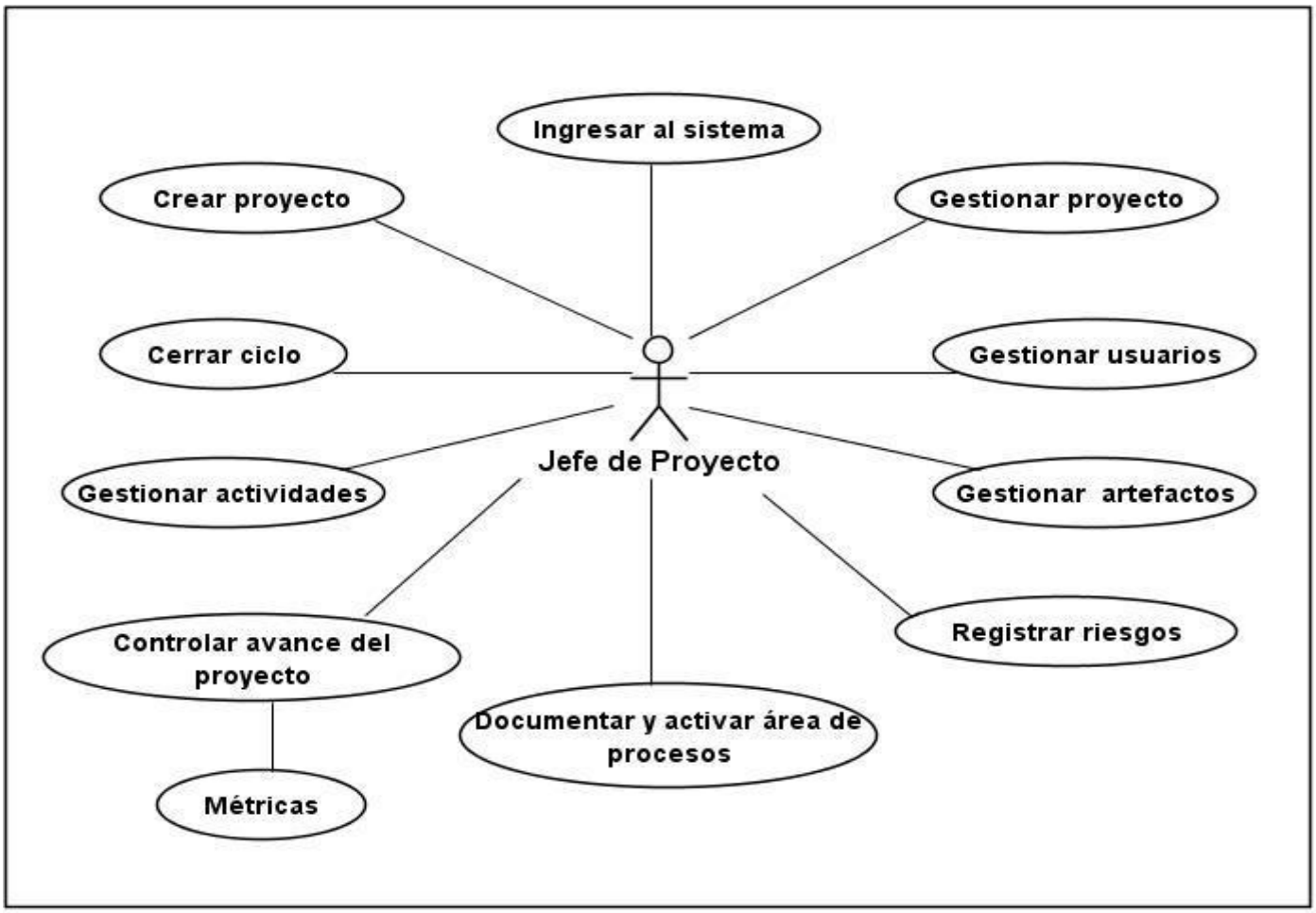


Figura 10: Diagrama de las funcionalidades del jefe de proyecto

Este diagrama resume todas las funciones disponibles en el sistema, ya que el rol de jefe de proyecto puede realizarlas todas.

3.2.4 Definición de requisitos no funcionales

Los requerimientos no funcionales son características que describen alguna forma o restricción para la realización de alguna funcionalidad o conjunto de ellas. Se consideran los atributos del sistema, propiedades que debe tener el producto. A continuación se relacionan los requerimientos no funcionales que debe cumplir el sistema clasificados según el conjunto de aspectos que representan.

- Apariencia o interfaz externa

El sistema deberá poseer una interfaz web sencilla, amigable, lo más atractiva y clara posible para el usuario, además su funcionamiento debe ser fácil de comprender.

- Usabilidad

El sistema solo podrá ser utilizado por aquellas personas que de una u otra forma se encuentren relacionadas con el control y aseguramiento de la calidad en los proyectos productivos de la Universidad de las Ciencias Informáticas.

- Rendimiento

Debe ser eficiente, con capacidad adecuada de procesamiento y cálculo, así como requiere de un tiempo de respuesta relativamente pequeño.

- Portabilidad

El sistema debe ser multiplataforma.

- Seguridad

Para poder acceder al sistema el usuario deberá estar registrado en la aplicación. Los usuarios serán creados en dependencia del rol que desempeñen. Se debe garantizar que solo posean acceso a la información con derecho a visualizar.

- Software

En el lado del Cliente debe existir un navegador que soporte Java Script. En el lado del Servidor debe estar instalado: PHP 5.0 o superior y gestor de base de datos Postgre SQL.

- Hardware

Para el desarrollo y ejecución de esta aplicación se necesita conexión a la red local, por lo que se requiere tarjeta de red. Pentium IV, 239 GHz, 249 MB de RAM.

3.2.5 Navegación del Sistema

Al ingresar al sistema se presentan dos opciones: hacer login en el sistema o crear proyecto de mejora de calidad de procesos, en caso de que el usuario nunca se haya registrado, debe de crear un proyecto de mejora. Al finalizar cualquiera de estas dos acciones se muestra la página principal y mediante el menú, se pueden acceder a las diversas funcionalidades que presenta el sistema.

La herramienta está construida de manera que pueda existir más de un proyecto de mejora de procesos de manera simultánea.

El siguiente diagrama ilustra la estructura de navegación de la herramienta:

CAPÍTULO 3
SOLUCIÓN PROPUESTA

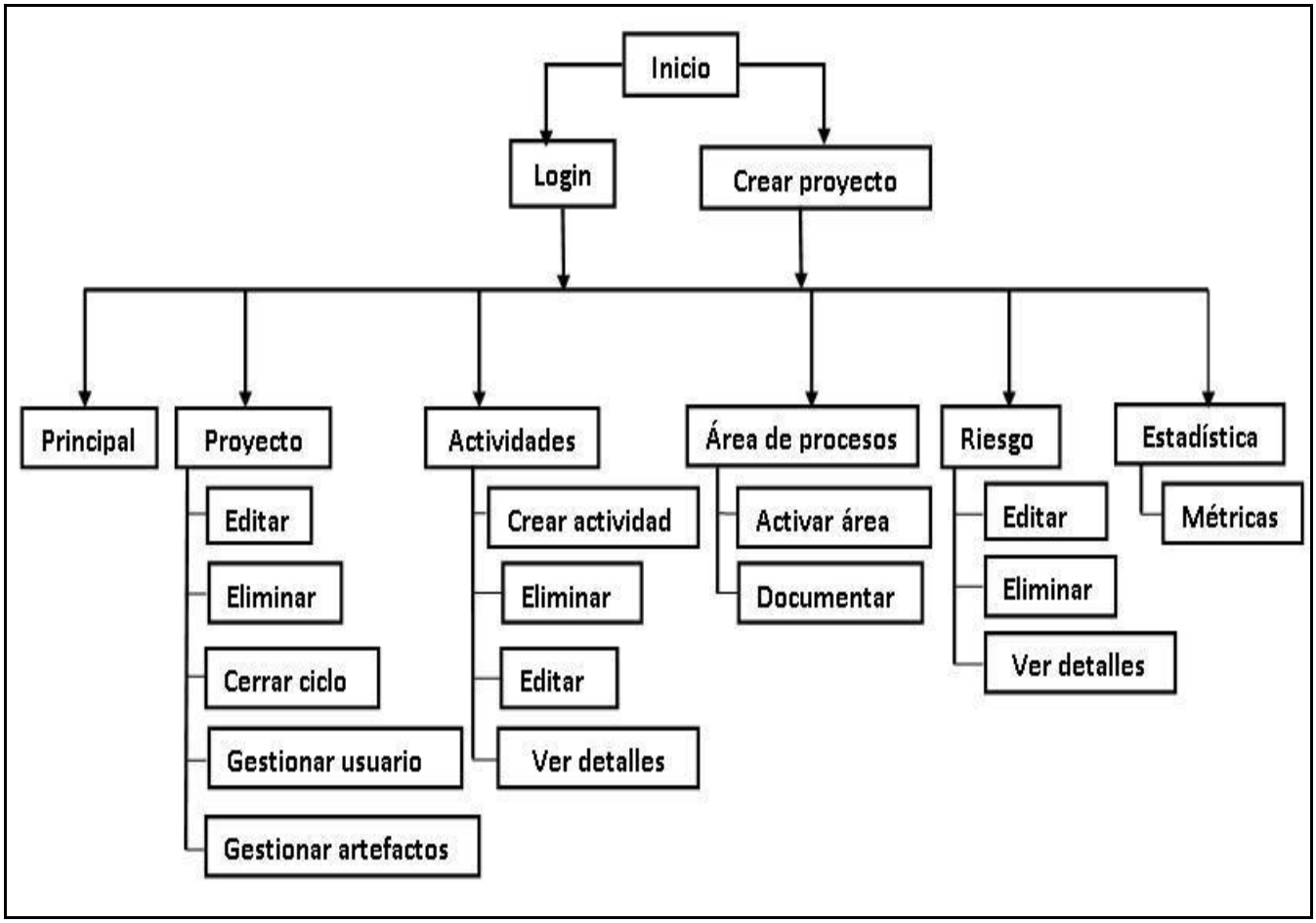


Figura 11: Mapa de navegación del Sistema

Esta figura representa el conjunto de páginas y sus relaciones navegacionales que son proporcionadas dentro de la herramienta.

3.3 Planificación del desarrollo de la propuesta

La metodología de desarrollo XP plantea la planificación como un diálogo entre el desarrollador y el cliente los cuales llegarán a un consenso con respecto al alcance del proyecto, la prioridad a la hora de implementar las funcionalidades requeridas, la composición de las versiones y la fecha de entrega de las mismas. La metodología plantea una serie de liberaciones parciales al terminar cada iteración para retroalimentar el proceso de desarrollo y probar si se están cumpliendo los requisitos del cliente.

3.3.1 Historias de usuario

Las historias de usuarios es uno de los artefactos más importante de esta fase, las cuales sirven de guía durante el proceso de desarrollo de esta metodología. Son elaboradas por el propio cliente describiendo los requisitos que deberá cumplir el sistema tal y como ellos lo entienden.

Tabla 3: Historia de usuario #1

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario**: Autenticar usuarios
Modificación de Historia de Usuario Número:	
Usuario: Usuarios	**Iteración Asignada**:1
Prioridad en Negocio: Alta (Alta/ Media/ Baja)	**Puntos Estimados**: 3/5
Riesgo en Desarrollo: Bajo (Alto / Medio / Bajo)	**Puntos Reales**:
Descripción: Al ingresar al sistema los usuarios ingresarán su usuario y password, en caso de no estar registrado se le da la opción de crear un proyecto de mejora.	
Observaciones:	

Tabla 4: Historia de usuario #2

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Crear proyecto de mejora
Modificación de Historia de Usuario Número:	
Usuario: Jefe de proyecto	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta/ Media/ Baja)	**Puntos Estimados:** 3/5
Riesgo en Desarrollo: Bajo (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: Crear el proyecto de mejora de calidad de procesos y además al usuario jefe de proyecto.	

Una vez creado el Proyecto de mejora de calidad de procesos, el jefe de proyecto podrá ingresar al sistema por medio de la página de login con su usuario y password, el mismo administrará toda la aplicación y tendrá todos los permisos sobre el sistema. El sistema creará automáticamente el primer Ciclo del proyecto, sus Fases siguiendo el modelo IDEAL y las actividades predefinidas por una plantilla.
Observaciones:

Tabla 5: Historia de usuario #3

	Historia de Usuario
Número: 3	**Nombre Historia de Usuario:** Gestionar proyecto de mejora
Modificación de Historia de Usuario Número:	
Usuario: Jefe de proyecto	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta/ Media/ Baja)	**Puntos Estimados:** 5/5
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: El usuario podrá gestionar el proyecto, es decir editarlo o eliminarlo. Se les puede agregar artefactos. Los artefactos son productos tangibles del proyecto, las cosas que el proyecto produce o usa para componer el producto final (modelos, documentos, código, ejecutables, etc.).Una vez que todas las actividades han sido completadas, el sistema permite cerrar el ciclo actual, luego de lo cual se abrirá automáticamente un nuevo ciclo, con sus fases y actividades.	
Observaciones:	

Tabla 6: Historia de usuario #4

	Historia de Usuario
Número: 4	**Nombre Historia de Usuario:** Gestionar artefactos
Modificación de Historia de Usuario Número:	
Usuario: Jefe de proyecto	**Iteración Asignada:** 1

Prioridad en Negocio: Alta (Alta/ Media/ Baja)	**Puntos Estimados:** 5/5
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: El usuario podrá agregar artefactos en base a proyectos, los que podrán ser vistos o eliminados. Los artefactos son productos tangibles del proyecto, las cosas que el proyecto produce o usa para componer el producto final (modelos, documentos, código, ejecutables, etc.).	
Observaciones:	

Tabla 7: Historia de usuario #5

		Historia de Usuario
Número: 5	**Nombre Historia de Usuario:** Cerrar ciclo	
Modificación de Historia de Usuario Número:		
Usuario: Jefe de proyecto		**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta/ Media/ Baja)		**Puntos Estimados:** 5/5
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)		**Puntos Reales:**
Descripción: Una vez que todas las actividades han sido completadas, el sistema permite cerrar el ciclo actual, luego de lo cual se abrirá automáticamente un nuevo ciclo, con sus fases y actividades.		
Observaciones:		

Tabla 8: Historia de usuario #6

		Historia de Usuario
Número: 6	**Nombre historia:** Gestionar actividades	
Modificación de Historia de Usuario Número:		
Usuario: Jefe de proyecto ,TWG's		**Iteración Asignada:** 2
Prioridad en Negocio: Alta (Alta/ Media/ Baja)		**Puntos Estimados:** 5/5

Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: El usuario podrá gestionar actividades, es decir visualizar una lista de actividades, crear, editar o eliminar actividades y además ver detalles de estas.	
Observaciones:	

Tabla 9: Historia de usuario # 7

		Historia de Usuario
Número: 7	**Nombre historia:** Activar y documentar área de proceso	
Modificación de Historia de Usuario Número:		
Usuario: Jefe de proyecto		**Iteración Asignada:** 2
Prioridad en Negocio: Alta (Alta/ Media/ Baja)		**Puntos Estimados:** 4/5
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)		**Puntos Reales:**
Descripción: Identificar las áreas de proceso que serán gestionadas en el ciclo actual. Para esto debe contar con una opción para activar áreas de proceso para el ciclo. Además de un link con la documentación de la misma.		
Observaciones:		

Tabla 10: Historia de usuario #8

		Historia de Usuario
Número: 8	**Nombre historia:** Gestionar usuarios	
Modificación de Historia de Usuario Número:		
Usuario: Jefe de proyecto		**Iteración Asignada:** 2
Prioridad en Negocio: Alta (Alta/ Media/ Baja)		**Puntos Estimados:** 4/5
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)		**Puntos Reales:**
Descripción: El jefe de proyecto es el encargado de asignar responsabilidades, mediante la administración (crear, modificar, eliminar) de usuario y asignación de roles.		

Observaciones:

Tabla 11: Historia de usuario #9

	Historia de Usuario
Número: 9	**Nombre historia:** Registrar riesgos
Modificación de Historia de Usuario Número:	
Usuario: Jefe de proyecto	**Iteración Asignada:** 2
Prioridad en Negocio: Alta (Alta/ Media/ Baja)	**Puntos Estimados:** 5/5
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: Se registrarán los riesgos identificados en el proyecto, los cuáles se podrán editar o eliminar y además se da la opción de mitigación del mismo cuando accede a sus detalles.	
Observaciones:	

Tabla 12: Historia de usuario #10

	Historia de Usuario
Número: 10	**Nombre historia:** Controlar y medir el avance del proyecto.
Modificación de Historia de Usuario Número:	
Usuario: Jefe de proyecto,SEPG,MSG	**Iteración Asignada:** 2
Prioridad en Negocio: Alta (Alta/ Media/ Baja)	**Puntos Estimados:** 5/5
Riesgo en Desarrollo: Alto (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción: El jefe de proyecto controla y mide el avance del proyecto mediante métricas, las cuales representan la visualización del avance y estado del proyecto.	
Observaciones:	

3.3.1.1 Planificación de las historias de usuarios

CAPÍTULO 3
SOLUCIÓN PROPUESTA

Con las historias de usuario elaboradas se comienzan las reuniones con el equipo de desarrolladores, para elaborar el plan de liberaciones. Esta planificación se basa en las estimaciones de los programadores de cuánto tiempo requerirá la implementación de cada historia. Se estima que el desarrollo de cada HU requiere entre 3 o 5 días – persona, por cada iteración, dependiendo de la prioridad que tengan las mismas.

Tabla 13: Plan de Liberaciones

No	Nombre de HU	Prioridad	Esfuerzo (Días)	Iteración
1	Autenticar usuarios	Alta	3	1
2	Crear proyecto de mejora	Alta	3	1
3	Gestionar proyecto de mejora	Alta	5	1
4	Gestionar artefactos	Ata	5	1
5	Cerrar ciclo	Alta	5	1
6	Administrar actividades	Alta	5	2
7	Activar y documentar área de proceso para el ciclo	Alta	4	2
8	Gestionar usuarios	Alta	4	2
9	Registrar riesgos	Alta	5	2
10	Controlar y medir el avance del proyecto.	Alta	5	2

La estimación elaborada en esta etapa de la planificación permite definir claramente la importancia que tiene cada historia de usuario para el cliente. Con esto se define la iteración en la que se debe desarrollar la misma para así elaborar el “plan de entregas” estimado, según el tiempo ideal establecido por los propios desarrolladores.

3.3.2 Plan de Entregas Estimado

En el *Plan de Entregas* deben quedar claramente definidos los objetivos que serán cumplidos en cada iteración. Además de las historias de usuario que se desarrollarán en cada una, el tiempo que se demorará y cómo se evaluará la calidad de la misma, que será mediante las pruebas de aceptación. Para el desarrollo de la herramienta, queda el siguiente plan de entregas:

Tabla 14: Plan de Entregas

Iteración	Objetivo de la Iteración	Orden de la HU a implementar	Duración total (Días)
1	Entregar la herramienta con las tres primeras funcionalidades, estipulada en las tres primeras historias de usuario. Éstas serán evaluadas por el usuario al finalizar la iteración, para la obtención de una herramienta con la calidad requerida por el cliente.	▪ HU_1:Autenticación de usuarios ▪ HU_2:Crear proyecto de mejora ▪ HU_3 Gestionar proyecto ▪ HU_4:Gestionar artefactos ▪ HU_5 Cerrar ciclo	21
2	Refinar el sistema con funcionalidades esenciales, como las expuestas en las tres últimas historias de usuario, y al igual serán evaluadas por el cliente.	▪ HU_6:Gestionar actividades ▪ HU_7:Activar y documentar área de procesos ▪ HU_8: Gestionar usuario ▪ HU_9: Registrar riesgos ▪ HU_10: Controlar y medir el avance del proyecto.	23

3.3.3 Planificación por iteraciones

El desarrollo de las iteraciones comienza desde la fase de planificación, con la elaboración de las tareas de ingeniería y las pruebas de aceptación. Las *tareas de ingeniería* son funcionalidades del sistema que se irán desarrollando por separado y luego serán integradas. Estas se elaboran a partir de las historias de usuario y también requieren de una planificación. Deben ser más descriptivas y sencillas de manera que su desarrollo no tome más de 3 días.(BECK, 1999)

Las *pruebas de aceptación* son las encargadas de probar al final de la iteración que funcionen correctamente los requisitos definidos en la misma. Estas deben ser diseñadas por el cliente cuidando que verifiquen la mayor cantidad de funcionalidades incluidas en las historias de usuario para la mayor cantidad de variantes posibles .(GUTIÉRREZ, 2008)

Luego al final de cada iteración el cliente llevará a cabo las pruebas y dará o no su aprobación sobre la calidad del sistema.

3.3.3.1 Iteración 1

En la primera iteración para poder liberar una primera versión o prototipo funcional al cliente, se implementarán las cinco primeras historias de usuario las cuales ofrecen características principales al sistema. Estas historias quedan distribuidas en tres tareas de ingeniería fundamentales.

Tabla 15: Tarea de ingeniería #1

	Tarea de Ingeniería
Número Tarea:1	**Número Historia de Usuario:**1
Nombre Tarea: Diseño de la interfaz primaria: “autenticarse”	
Tipo Tarea: Desarrollo	**Puntos Estimados:**3/5
Fecha Inicio: 15/04/09	**Fecha Fin:** 18/04/09
Programador Responsable: Ilenis Fajardo Terrero	
Descripción: Se desarrollará una ventana de la presentación inicial de la aplicación y el punto de partida para el inicio de la configuración de la autenticación, teniendo en cuenta que debe ser posible ir agregando progresivamente el resto de los componentes visuales que permitirán el inicio de las demás configuraciones, sin que sea necesario un cambio en lo ya diseñado	

Tabla 16: Tarea de ingeniería #2

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Diseño de la interfaz secundaria: “Crear Proyecto de Mejora de Procesos”	
Tipo Tarea: Desarrollo	**Puntos Estimados:**3/5
Fecha Inicio: 19/04/09	**Fecha Fin:**22/04/09
Programador Responsable: Ilenis Fajardo Terrero	
Descripción: Se desarrollará una ventana que contendrá todos los campos necesarios a llenar para crear el proyecto de mejora de procesos, incluyendo el jefe de este proyecto.	

Tabla 17: Tarea de ingeniería #3

	Tarea de Ingeniería
Número Tarea: 3	**Número Historia de Usuario:** 3 , 4, 5
Nombre Tarea: Diseño de la interfaz secundaria: “Proyecto”	
Tipo Tarea: Desarrollo	**Puntos Estimados:**10/5
Fecha Inicio: 22/04/09	**Fecha Fin:** 02/05/09
Programador Responsable: Ilenis Fajardo Terrero	
Descripción: Se desarrollará una ventana que contendrá toda la información referente al proyecto, incluyendo la opción de cerrar ciclo y la de subir artefactos.	

Pruebas de aceptación

Las pruebas de aceptación diseñadas por el cliente para verificar las funcionalidades implementadas en esta iteración se concentran en las cinco primeras historias de usuario. Se chequeará cada acción implicada con acciones válidas e incorrectas, para asegurar la calidad de la herramienta.

Se diseñaron pruebas de aceptación por cada historia de usuarios en las cuales se analizan las diferentes opciones posibles a seleccionar y que pudieran generar un error de funcionamiento de la aplicación. (Ver_ Anexo III: Casos de pruebas de aceptación de las historias de usuario de la iteración 1)

3.3.3.2 Iteración 2

En la segunda iteración se implementarán las cinco últimas historias de usuarios. Estas proveerán otras de las funcionalidades importantes para el sistema, y quedan distribuidas en siete tareas de ingeniería, incluyendo las dos últimas también para la primera iteración.

Tabla 18: Tarea de ingeniería #4

	Tarea de Ingeniería
Número Tarea: 4	**Número Historia de Usuario:** 6
Nombre Tarea: Diseño de la interfaz secundaria: “Actividades”	
Tipo Tarea: Desarrollo	**Puntos Estimados:** 5/5

Fecha Inicio: 03/05/09	**Fecha Fin:** 08/05/09
Programador Responsable: Ilenis Fajardo Terrero	
Descripción: Se desarrollará una ventana que muestre el listado de todas las actividades a desarrollar de acuerdo a los criterios de filtro que hayan sido seleccionados.	

Tabla 19: Tarea de ingeniería #5

	Tarea de Ingeniería
Número Tarea: 5	**Número Historia de Usuario: 7**
Nombre Tarea: Diseño de la interfaz secundaria: “Procesos”	
Tipo Tarea: Desarrollo	**Puntos Estimados:** 4/5
Fecha Inicio: 09/05/09	**Fecha Fin:** 13/05/09
Programador Responsable: Ilenis Fajardo Terrero	
Descripción: Esta pantalla mostrará las áreas de proceso de los cinco niveles que conforman el modelo CMMI, con el link “documentar” y “activar”, y en la próxima pantalla se crearán las actividades que se realizarán para la mejora de esta área de proceso. Se debe especificar el responsable del área de proceso, y la fecha de inicio con la que serán creadas todas las actividades.	

Tabla 20: Tarea de ingeniería #6

	Tarea de Ingeniería
Número Tarea: 6	**Número Historia de Usuario:** 8
Nombre Tarea: Diseño de la interfaz secundaria: “Crear usuario”	
Tipo Tarea: Desarrollo	**Puntos estimados:** 4/5
Fecha Inicio: 14/05/09	**Fecha Fin:** 18/05/09
Programador Responsable: Ilenis Fajardo Terrero	
Descripción: Esta ventana mostrará los campos a llenar por el jefe de proyecto para la creación de un nuevo usuario, al que se le asigna un rol, y luego podrá autenticarse en el sistema.	

Tabla 21: Tarea de ingeniería #7

	Tarea de Ingeniería
Número Tarea: 7	**Número Historia de Usuario:** 9
Nombre Tarea: Diseño de la interfaz secundaria: “Riesgo	
Tipo Tarea: Desarrollo	**Puntos estimados:** 5/5
Fecha Inicio: 19/05/09	**Fecha Fin:** 23/05/09
Programador Responsable: Ilenis Fajardo Terrero	
Descripción: Esta ventana mostrará el registro de los riesgos identificados en el proyecto.	

Tabla 22: Tarea de ingeniería #8

	Tarea de Ingeniería
Número Tarea: 8	**Número Historia de Usuario:** 10
Nombre Tarea: Diseño de la interfaz secundaria: “Avance”	
Tipo Tarea: Desarrollo	**Puntos estimados:** 5/5
Fecha Inicio: 25/05/09	**Fecha Fin:** 29/05/09
Programador Responsable: Ilenis Fajardo Terrero	
Descripción: Se desarrollará una ventana donde se muestre el avance y estado del proyecto, mediante el cálculo de métricas, así como su representación gráfica.	

Tabla 23: Tarea de ingeniería #9

	Tarea de Ingeniería
Número Tarea: 9	**Número Historia de Usuario:** 1,2,3,4,5,6,7,8,9,10
Nombre Tarea: Diseño de la BD	
Tipo Tarea: Desarrollo	**Puntos Estimados:**5/5
Fecha Inicio: 15/04/09	**Fecha Fin:** 31/05/09
Programador Responsable: Ilenis	

Descripción: Se diseñará una base de datos que permita el almacenamiento de toda la información del sistema para la realización de consultas.

Tabla 24: Tarea de ingeniería #10

	Tarea de Ingeniería
Número Tarea: 10	**Número Historia de Usuario:** 1,2,3,4,5,6,7,8,9,10
Nombre Tarea: Implementación	
Tipo Tarea: Desarrollo	**Puntos Estimados:**15
Fecha Inicio: 15/04/09	**Fecha Fin:** 31/05/09
Programador Responsable: Ilenis	
Descripción: Se implementan todas las clases necesarias para permitir el funcionamiento de toda la aplicación.	

Pruebas de aceptación

En esta iteración se elaboraron pruebas de aceptación para las cinco últimas historias de usuario. (Ver_Anexo IV: Casos de pruebas de aceptación de las historias de usuario de la iteración 2)

3.4 Diseño del Sistema

Durante el ciclo de vida del desarrollo de un software utilizando XP la fase de diseño del software se realiza justo después que se termina la fase de planificación. Durante esta fase se comienza a definir el diseño arquitectónico del software partiendo de las historias de usuario de la primera iteración y de la metáfora, un elemento que según Kent Beck el creador de la metodología, guía al equipo de proyecto durante todo el desarrollo del programa y que además sustituye gran parte de lo que sería la arquitectura del sistema en modelos de desarrollo más pesados o tradicionales.(BECK, 1999)

3.4.1 Metáfora del sistema propuesto

La metáfora creada por Kent Beck es una historia simple de cómo funciona todo el sistema compartido por todo el equipo de desarrollo. Su misión es describir el software de manera que los desarrolladores y clientes que trabajan en conjunto puedan comunicarse fácilmente a la hora de referirse al sistema, o a una parte de éste. Cuando el sistema es pequeño la metáfora es extremadamente sencilla, pero cuando es un sistema complejo el que se desea describir, puede llevar una explicación que clarifique el funcionamiento del mismo. (BECK, 1999)

Teniendo en cuenta los parámetros anteriormente expuestos, la metáfora del sistema sería: *El Sistema automatiza el control de la implantación de un proyecto de mejora de procesos.*

3.4.2 Arquitectura del sistema

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema. Consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. La arquitectura de software establece los fundamentos para que analistas, diseñadores, programadores trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. La Arquitectura define un conjunto de elementos, conectores, restricciones y un sistema de control que caracterizan a un sistema o a una familia de sistemas.

La arquitectura de la herramienta o de cualquier proyecto guiado por XP debe ser sencilla y responder al principio de "*cuál es la solución más simple capaz de funcionar*", debe ser una estructura que organiza la lógica del sistema de manera que al realizar cambios en partes no provoca cambios en otros componentes a menos que sea absolutamente necesario. Debe enfocarse en cómo resolver los problemas identificados en el momento en que se diseña, sin tener en cuenta posibles variaciones o requerimientos futuros, los cuales serán enfrentados y agregados al diseño en el momento en que sean detectados.

CAPÍTULO 3
SOLUCIÓN PROPUESTA

Entre los estilos arquitectónicos más conocidos se encuentra arquitectura en capas y Modelo-Vista-Controlador (MVC), este último fue el seleccionado para guiar el diseño del Sistema de Implantación de un Proyecto de Mejora.

3.4.2.1 Patrón Modelo – Vista – Controlador

SIMPuci está concebido según el patrón de arquitectura: Modelo-Vista-Controlador. El patrón fue seleccionado atendiendo al hecho de que permite cambios en el modelado de la aplicación sin la necesidad de modificar el código de la interfaz, con el objetivo de hacer el software compatible con las variaciones de dichos archivos. Los tres niveles que integran este patrón arquitectónico, están definidos de la siguiente manera:

- 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.

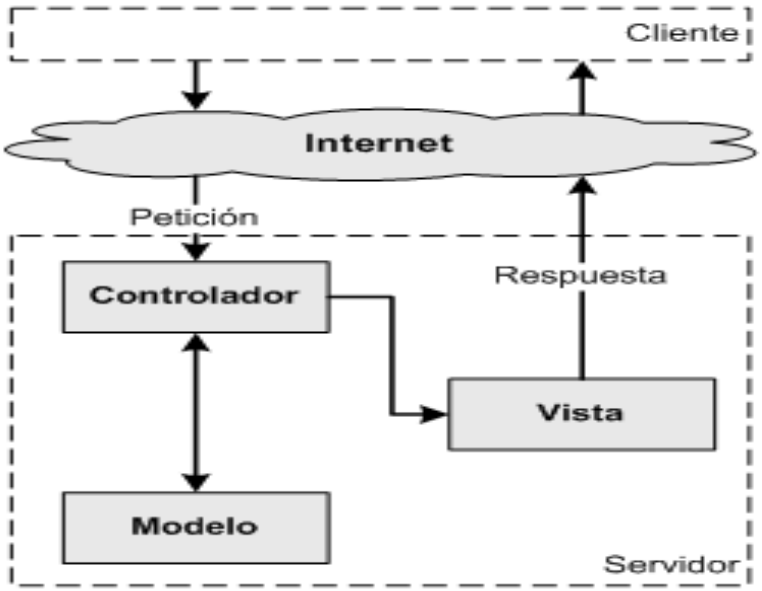


Figura 12: Representación del patrón MVC

Este modelo de arquitectura presenta las siguientes ventajas:

- Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.
- Es posible tener diferentes vistas para un mismo modelo (representación de un conjunto de datos como una tabla o como un diagrama de barras).
- Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.
- Proporciona un mecanismo de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción).

El patrón arquitectónico Modelo-Vista-Controlador usa los patrones de diseño Observer, Strategy y Composite: el patrón Observer conecta la vista (View) al modelo (Model), el patrón Strategy permite al controlador (Controller) manejar los eventos que ocurren sobre la vista, y el patrón Composite provee una jerarquía de vistas.
En el estilo arquitectónico MVC se fomenta la escalabilidad del sistema, la seguridad y la reutilización de código.

3.4.2.2 Representación de la arquitectura en su Vista Lógica

Esta vista representa un subconjunto del diseño, la cual representa los elementos de diseño más importantes para la arquitectura del sistema. Este describe las clases más importantes, y su organización en paquetes.

CAPÍTULO 3
SOLUCIÓN PROPUESTA

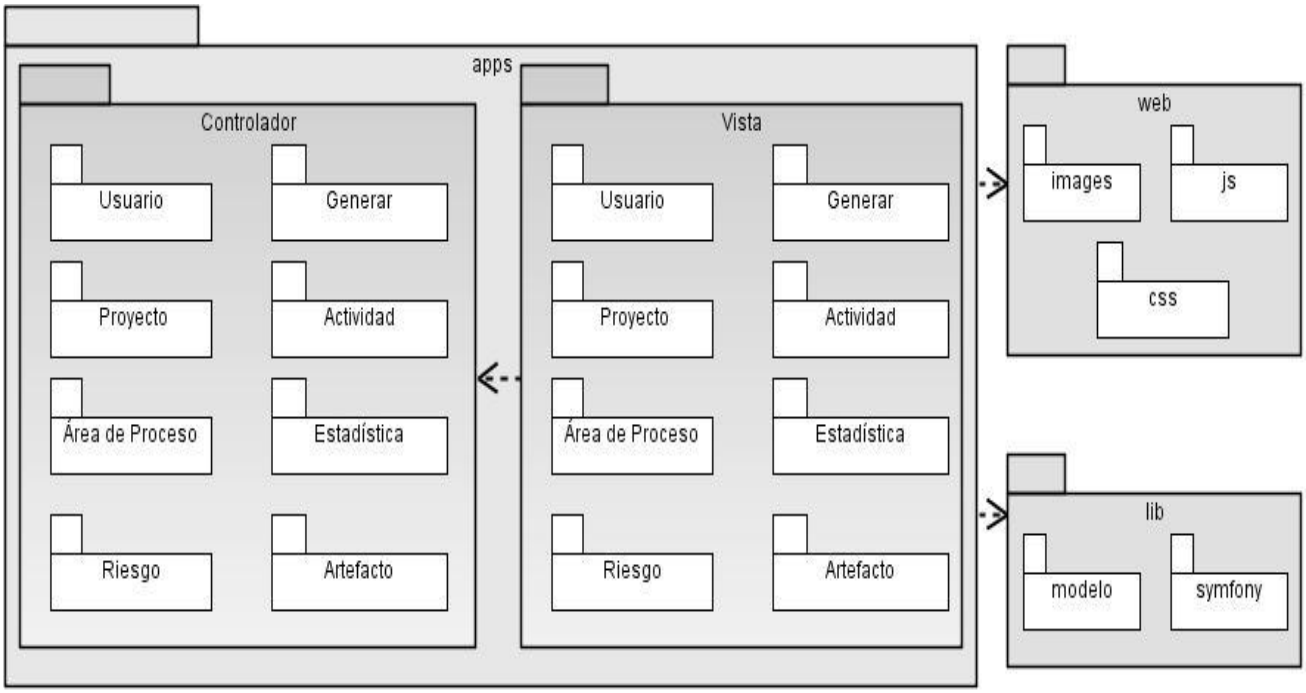


Figura 13: Diagrama de paquetes

Dentro del paquete "symfony" que se muestra en la figura anterior están situadas las clases del núcleo del framework de las cuales las principales son estas que mencionamos a continuación.

- **sfController** es la clase del controlador. Se encarga de decodificar la petición y transferirla a la acción correspondiente.
- **sfRequest** almacena todos los elementos que forman la petición (parámetros, cookies, cabeceras, etc.)
- **sfResponse** contiene las cabeceras de la respuesta y los contenidos. El contenido de este objeto se transforma en la respuesta HTML que se envía al usuario.

Paquete	Descripción
apps	Contiene un directorio por cada aplicación del proyecto
Vista	Contiene las clases Interfaz del sistema, separadas por paquetes.
Controlador	Contiene las clases Control del sistema, separadas por paquetes.

web	La raíz del servidor web. Los únicos archivos accesibles desde Internet son los que se encuentran en este directorio
images	Contiene todas las imágenes que se utilizan en las interfaces del sistema.
css	Contiene los estilos que se aplican en las interfaces del sistema.
js	Contiene las clases desarrolladas en Java Script que se utilizan en las interfaces del sistema.
lib	Almacena las clases y librerías externas. Se suele guardar todo el código común a todas las aplicaciones del proyecto.
modelo	Guarda el modelo de objetos del proyecto.
symfony	Contiene los principales archivos el framework. Ejemplo clases y helper.

Tabla 25: Descripción de los paquetes del sistema

3.4.3 Diseño de la Base de Datos

La base de datos es el sistema utilizado para el almacenamiento de información y acceso controlado. El modelo que se presenta a continuación muestra las relaciones que existen entre todas las entidades de la base de datos, con sus atributos y su tipo.

CAPÍTULO 3
SOLUCIÓN PROPUESTA

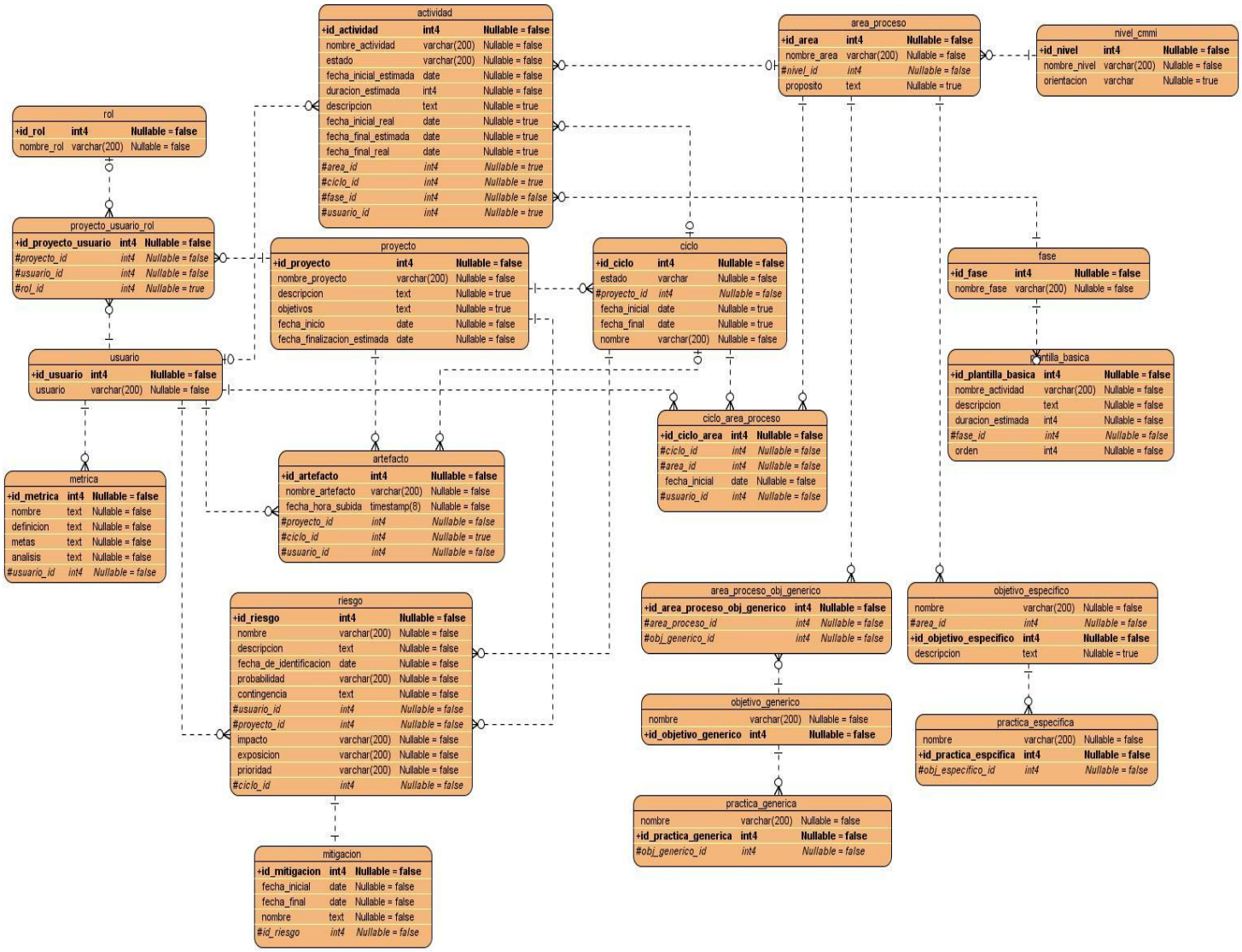


Figura 14: Modelo Entidad Relación

3.5 Modelo de Despliegue

El modelo de despliegue describe la distribución física del sistema, muestra cómo están repartidos los componentes de software entre los distintos nodos de cómputo. Este modelo lo forma una colección de nodos y arcos; donde cada nodo representa un recurso de cómputo, normalmente un procesador o un

dispositivo de hardware similar. La obtención de este modelo permite comprender la correspondencia entre la arquitectura software y la arquitectura hardware.

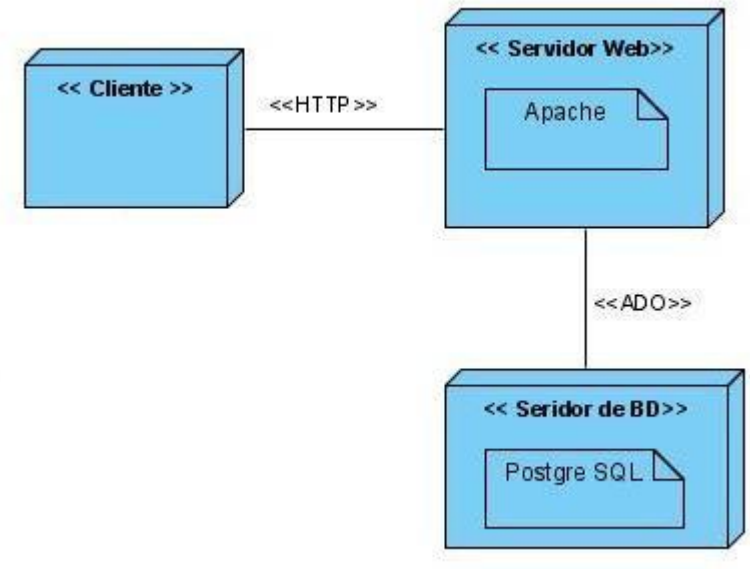


Figura 15: Diagrama de Despliegue

3.6 Modelo de Implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y paquetes que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios.

Un diagrama de componentes representa la separación de un sistema de software en componentes físicos y muestra las dependencias entre estos. Cada subsistema corresponde a un paquete físico y cada componente a un módulo, fichero o librería existente en la memoria de almacenado. Seguidamente se expone el diagrama de componente del sistema.

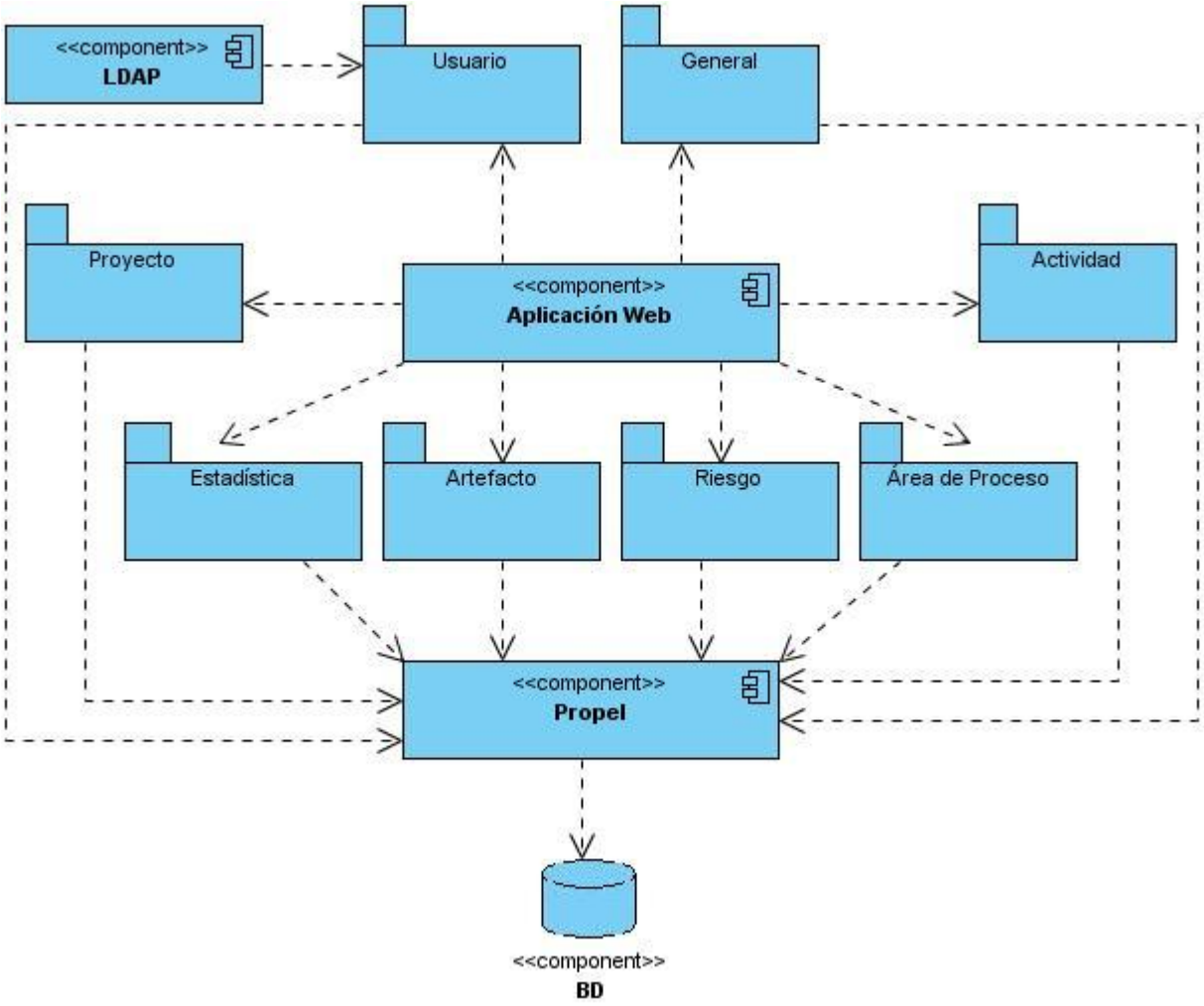


Figura 16: Diagrama de Componentes del modelo de implementación

3.7 Estándares de la interfaz de la aplicación

Para la construcción de este sistema se tendrá en cuenta los estándares de diseño de aplicaciones web. Este contiene una página inicial en el sitio donde los usuarios deben introducir sus datos para ser autenticados, una vez validados sus datos tendrán acceso a diferentes páginas y menú del sitio Este diseño se mantiene en toda la aplicación.

En la parte superior de las páginas aparecerá una imagen de banner donde se muestra el nombre de la aplicación y su menú. El color predominante en el sistema será el azul.

Figura 17: Prototipo de Interfaz de Usuario

3.7.1 Tratamiento de Excepciones

El sistema tratará de minimizar al máximo los posibles errores que puedan existir. En el caso de los datos introducidos por los usuarios en los formularios, se evita que el mismo tenga que teclear la información, brindando la posibilidad de que la pueda seleccionar de una lista y de esta manera siempre serán válidos los datos de entrada.

En caso contrario se hace una validación de datos utilizando las facilidades que brinda Java Script y en la mayoría de los casos la validación se realiza del lado del cliente, ya que son estos precisamente los que introducen los datos al sistema.

CAPÍTULO 3
SOLUCIÓN PROPUESTA

3.8 Conclusiones

Durante el transcurso del presente capítulo se realizó una descripción de la solución general propuesta, se definieron y redactaron las historias de usuarios así como las tareas de la ingeniería correspondientes a cada una de ellas, estimándose un tiempo ideal de 44 días necesarios para implementar la herramienta. Además se obtuvo la metáfora que va a describir el sistema y fue diseña y descrita la base de datos, así como también el prototipo de la interfaz. Como culminación al diseño se presentó el modelo de implementación describiendo la distribución física del sistema y sus componentes.

CONCLUSIONES

Con la realización de este trabajo de diploma se obtuvo una propuesta que da cumplimiento al objetivo general planteado, al lograr desarrollar un sistema que permita de forma automatizada monitorear y controlar las buenas prácticas de la implantación de CMMI, lo que facilita el control eficiente del avance de un proyecto de mejora de calidad de procesos. Para llegar a estos resultados:

- Se caracterizó el modelo CMMI, el cual propone a su vez el modelo IDEAL como guía para la mejora continua de procesos.
- Se caracterizó el modelo COBIT, como método para el monitoreo y control de dicha implantación.
- Se utilizaron las metodologías, tendencias y herramientas propuestas para el desarrollo de la aplicación.
- Se generaron los artefactos propuestos por la metodología usada.

Por medio de la herramienta se logra dar soporte y seguimiento a la implantación de CMMI basándose en el modelo IDEAL, proporciona visibilidad durante todo el proyecto, estado actual y avance, permitiendo dar un correcto seguimiento y obtener claridad con respecto al progreso del proyecto. Además, se abordan las diferentes áreas de proceso identificadas por el modelo CMMI, las cuales pueden ser gestionadas en uno o varios ciclos, dependiendo de la planificación de la organización.

RECOMENDACIONES

Se recomienda:

- Identificar nuevas funcionalidades para el sistema que responden a las necesidades de la implantación de un proyecto de mejora para un óptimo proceso.
- Valorar la integración de la aplicación en el programa de mejora que se está llevando en la universidad.

REFERENCIAS BIBLIOGRÁFICAS

- ALARCON, S. Modelos de Calidad. 2004, nº
- BECK, K. *Extreme Programming Explained.* 1999.
- CABREJAS, M. *Guía para la planificación y seguimiento de programas de mejoras en proyectos productivos UCI.* Universidad de las Ciencias Informáticas, 2008.
- CARLOS-III. Taller sobre Pruebas en Ingeniería del Software. 2006, nº [Consultado el: 20 de enero de 2009]. Disponible en: http://in2test.isi.uniovi.es/pris2006.
- CobiT Audit Advisor. 2008, nº [Consultado el: 9 de mayo de 2009]. Disponible en: http://www.netconsul.com/software/index.php?ver=CobitAdvisor.
- CRUZ, S. *Implementación del modelo COBIT en el proceso productivo de la Casa de Autoría DVD de la UCI.* Universidad de las Ciencias Informáticas, 2007.
- CHIAVENATTO, I. *Administración: Proceso Administrativo.* Editado por: Mcgraw-Hill. 3ra ed. Colombia: 2000. ISBN 9584101617.
- FABIEN. *Symfony, la guía definitiva.*[Consultado el: 6 de Febrero de 2009]. Disponible en: http://www.librosweb.es/symfony/index.html.
- FAIRLEY, R. Risk Management for software projects, IEEE Software 1994, nº
- FAROL, H. *Administración industrial y general.* Buenos Aires: 1961.
- GUTIÉRREZ, J. Pruebas del sistema en Programación Extrema. 2008, nº

REFERENCIAS BIBLIOGRÁFICAS

- HISTA. Consultoría en Metodologías de Desarrollo de Software. 2007, nº [Consultado el: 19 de Enero de 2009]. Disponible en: http://www.histaintl.com/servicios/consulting/ideal.php.

- HORNA, A. 2004, nº [Consultado el: 20 de Abril de 2009]. Disponible en: http://www.aristidesvara.com.

- HOYLE, D. *Manual de Valoración de Calidad ISO 9000.* Editado por: Paraninfo, T. 1ra ed. Madrid: 1998. ISBN 9788428324625

- HUACOTO, N. *Propuesta para implantar CMMI en una empresa con múltiples unidades desarrolladoras de software.* 2005.

- HUMPHREY, W. Introducción al Proceso Software Personal. 1995, nº

- IEEE. *Diccionario de computación.* 1990.

- INFORMATICA. Control de Calidad en los Sistemas. 2000., nº

- ISACA. *COBIT Objetivos de Control.* 3ra ed. 2000. ISBN 1-893209-99-7.

- ISO/IEC-9126. Norma internacional ISO/IEC-9126. 1991, nº

- ITGI. *COBIT* 4ta ed. 2005. ISBN 1-933284-37-4

- JACOBSON, B., RUMBAUGH. . *El Proceso Unificado de Desarrollo de Software.* Editado por: Editorial Addison Wesley. Madrid: 2000.

- JURAN, J. *Juran y la Planificación para la Calidad.* Editado por: Santos, D. D. Madrid,España: 1990.

REFERENCIAS BIBLIOGRÁFICAS

- MANSO, E. Calidad del Software. 2005, nº
- MCFEELEY, B. SEI: Software Engineering Institute.IDEAL: Guía de usuario para la mejora de proceso de software. 2003, nº [Consultado el: 15 de Enero de 2009]. Disponible en: http://www.sei.cmu.edu.
- Meycor COBIT Guías de Gerenciamiento - Management Guidelines (MG). 2008, nº Disponible en: http://www.datasec-soft.com/sp/content/view/8/12/.
- MOEN. *"A generation perspective on small firm internationalisation: from traditional exporters and flexible specialists to born globals".* Reassesing the internationalisation of the firm, 2001. vol. 11, 195-223 p.
- MORAGA, H. Normativas de Calidad:CMMI e ISO/IEC 9126. 2004, nº
- PAREDES, E. Homogeneización de Conceptos de Servicios de Desarrollo Empresarial (SDE)2003, nº
- PARRO. Uso de la representación continua de CMMI® para la Mejora de Negocio. 2007, nº
- PÉREZ, M. TQS analiza la situación actual de CMMI. 2008, nº Disponible en: www.tqs-es.com.
- PIATTINI, M. Classifying web metrics using the web quality model. 2005, nº
- PRESSMAN, R. *Ingenieria del Sofware.Un Enfoque Práctico.* 5ta Edición ed. Madrid, España: McGraw-Hill, 2002. 640 Páginas p. ISBN 8448132149
- REA. *Diccionario de la REA.* 2003, vol. 22a, Disponible en: http://buscon.rae.es.

REFERENCIAS BIBLIOGRÁFICAS

- RODRÍGUEZ, A. Aplicaciones Web vs Aplicaciones de Escritorio. 2009, nº [Consultado el: 25 de Mayo de 2009]. Disponible en: http://blog.develoft.com/?p=18.

- SEI. Software Engineering Institute. CMMI-DEV, V1.2. 2006, nº [Consultado el: 20 de Marzo de 2009]. Disponible en: http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr008.pdf.

- *SIMPLe-Sistema de Implementación de Mejora de Proceso.* 2008, Disponible en: http://www.alturasoluciones.com/intro.html.

- SU, S. *Database Computers:Principles,Architectures and Techniques.* Editado por: Graw-Hil, M. New York 1998.

- THAYER, R. Tutorial: Software Engineering Project Mangement. IEEE Computer Press. 1988, nº

- VILLA, M. Modelos de Evaluación y Mejora de Procesos: Análisis comparativo. . 2004, nº [Consultado el: 15 de Febrero de 2009]. Disponible en: http://ftp.informatik.rwth-aachen.de/Publication/CEUR-ws/vol-120/paper4.pdf.

- VITALI, S. El modelo IDEAL para implantar CMMI. 2005, nº

- WESTFALL, L. “Software metrics that meet your information needs”. 1995, nº [Consultado el: 20 de Abril de 2009]. Disponible en: http://www.asq-software.org/metrics/aqc95.html.

BIBLIOGRAFÍA CONSULTADA

- Cobit 4.0 en español. nº [Consultado el: 20 de Enero de 2009]. Disponible en: http://www.isaca.com.
- PEÑA, Y. Propuesta de proceso macro de Monitoreo y Control para nivel 2 de CMMI. Universidad de las Ciencias Informáticas, 2008.
- NIEVES, M. Herramienta Informática para automatizar los procesos en el Laboratorio de Calidad de Software: Módulo Gestión de las No Conformidades. Universidad de las Ciencias Informáticas, 2007.
- L.PERALTA, M. Asistente para la Evaluación de CMMI-SW.
- LETELIER, J. Metodologías Ágiles en el desarrollo de software: eXtreme Programming (XP). 2008, nº
- GARCIA, R. Factores Clave para la mejora de procesos, La diferencia para el éxito. 2007, nº [Consultado el: 6 de Marzo de 2009]. Disponible en: http://www.software.net.mx/desarrolladores/prosoft/Estudios/factoresprocesos.htm.
- GÓMEZ, R. Control y monitoreo de proyectos. nº [Consultado el: 25 de Enero de 2009]. Disponible en: http://www.ictnet.es/2007/control-y-monitoreo-de-proyectos.
- ALEXANDRA, J. Herramienta para soporte al proyecto de mejora de calidad de procesos con calidad con modelos CMMI e IDEAL. Desarrollo, Universidad de Chile, 2008.
- Revista de Procesos y Métricas de las tecnologías de la información 2007, vol. 4, nº Disponible en: http://www.aemes.org
- Introduction to CMMI. The Carnagie Mellon University, 2006. vol. Versión 1.2

BIBLIOGRAFÍA CONSULTADA

- ISO/SPICE. nº [Consultado el: 15 de Enero de 2009]. Disponible en: http://www.usm.edu.ec/abedini/spice.
- GÓNGORA, M. Herramienta Informática para automatizar los procesos en el Laboratorio de Calidad de Software: Módulo Gestión de las No Conformidades. Universidad de las Ciencias Informáticas, 2007.

ANEXOS

Anexo I

Guía de la Entrevista

Guía para entrevistar a:

Introducción:

- **Saludo**

La presente entrevista está dirigida a especialistas de la Dirección de Calidad involucrados en el proyecto de mejora corriendo.

- **Período que recoge la entrevista:** Curso 2008 – 2009, período de inicio del programa de mejora en la UCI.
- **Objetivo General:** Investigar sobre la problemática existente en la implantación del proyecto de mejora de procesos basado en CMMI, además de las características del mismo.
- **Objetivo específicos:**
  - ✓ Conocer las dificultades de la implantación del actual proceso de mejora y el desempeño de quienes lo realizan.
  - ✓ Conocer cómo se realiza la introducción de la mejora en los proyectos.
  - ✓ Conocer la manera en que se está monitoreando y controlando el proceso.
  - ✓ Investigar sobre la necesidad de la automatización del monitoreo y control del proceso de mejora.

- **Desarrollo:**

1. ¿Qué problemas existen actualmente en la implantación del programa de mejora de procesos basado en CMMI?
2. ¿Por qué se escoge este modelo?
3. ¿Se hace uso de IDEAL como método guía para la implantación del modelo CMMI?
4. ¿Se realizan las actividades propuestas en cada una de sus fases?

5. ¿Se hace un chequeo de las mismas?
6. En caso de realizarse , es :

   ------ Bueno

   ------ Regular

   ------ Malo
7. Debido a que es la primera vez que se realiza este programa de proceso en la universidad cree usted que:

a) El personal encargado del mismo presenta:

   --------- Buena experiencia práctica

   -------- Regular experiencia práctica

   -------- Mala experiencia práctica

b) En cuanto al dominio del rol que le corresponde desempeñar, presentan:

   ------- Alto dominio del rol

   ------- Medio dominio del rol

   -------- Bajo dominio del rol

8. ¿Se documenta todo este proceso?
9. ¿Se realiza la mejora de proceso en los proyectos productivos de la UCI?
10. ¿Cómo es la calidad de este proceso?

    ------ Buena

    ------ Mala

    ------ Regular
11. ¿Se monitorea y controla este proceso?
12. En caso de realizarse, ¿Cómo se realiza este monitoreo y control?
13. ¿Crees necesario el uso de una herramienta que permita automatizar el monitoreo y control de la implantación de CMMI?

- **Conclusiones:**
- **Agradecer su cooperación.**
- **Despedida**

Anexo II

Diferencias entre metodologías ágiles y no ágiles.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Anexo III

Casos de pruebas de aceptación de las historias de usuario de la iteración 1

Caso de prueba de aceptación_ 01: Usuario correctamente autenticado

	Caso de Prueba de Aceptación
Código Caso de Prueba: 01	**Número Historia de Usuario:** 1
Nombre del Caso de Prueba: Usuario correctamente autenticado	

Descripción: El usuario desde la página de autenticación intenta acceder al sistema introduciendo usuario y contraseña. El sistema verifica que estos datos sean correctos, en caso afirmativo le da acceso según sus permisos a las funcionalidades correspondientes.
Condiciones de Ejecución: Ninguna
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Aparece un cuadro de texto en el que se solicita el nombre de usuario y la contraseña (password). - El usuario introduce ambos y presiona el botón “entrar” - El sistema verifica ambos campos en la base de datos y comprueba que exista.
Resultado Esperado: El sistema muestra la página principal y mediante el menú accede a las restantes según sus privilegios.
Evaluación de la Prueba: Satisfactoria

Caso de prueba de aceptación_ 02: Usuario incorrectamente autenticado

Caso de Prueba de Aceptación	
Código Caso de Prueba: 02	**Número Historia de Usuario:** 1
Nombre del Caso de Prueba: Usuario incorrectamente autenticado	
Descripción: El usuario desde la página de autenticación intenta acceder al sistema introduciendo su usuario y contraseña. El sistema verifica que estos datos sean correctos, en caso negativo, se muestra un mensaje de error.	
Condiciones de Ejecución: Ninguna	
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Aparece un cuadro de texto en el que se solicita el nombre de usuario y la contraseña (password). - El usuario introduce ambos y presiona el botón “entrar” - El sistema verifica ambos campos en la base de datos y comprueba que no existe tal	

usuario.
Resultado Esperado: El sistema muestra un mensaje de error si los datos de autenticación son inválidos.
Evaluación de la Prueba: Satisfactoria

Caso de prueba de aceptación_ 03: Proyecto de mejora correctamente creado

	Caso de Prueba de Aceptación
Código Caso de Prueba: 03	**Número Historia de Usuario:** 2
Nombre del Caso de Prueba: Proyecto de mejora correctamente creado	
Descripción: La HU inicia cuando el usuario desea acceder a la aplicación, y no se encuentra registrado, se le da acceso a crear un nuevo proyecto de mejora. El sistema verificará que todos los campos obligatorios sean completados y que no exista ya algún proyecto con el nombre dado.	
Condiciones de Ejecución: Ninguna	
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Accede al vínculo de la creación de un nuevo proyecto. - El usuario llena todos los campos requeridos. - El sistema inserta los datos en la base de datos y comprueba que no existe un proyecto con ese nombre.	
Resultado Esperado: ▪ El sistema le muestra automáticamente la página principal con el primer Ciclo del proyecto creado, y sus fases correspondiente al modelo IDEAL, además de las actividades predefinidas por una plantilla.	
Evaluación de la Prueba: Satisfactoria	

Caso de prueba de aceptación_ 04: Proyecto de mejora incorrectamente creado

	Caso de Prueba de Aceptación
Código Caso de Prueba: 04	**Número Historia de Usuario:** 2
Nombre del Caso de Prueba: Proyecto de mejora incorrectamente creado	

Descripción: La HU inicia cuando el usuario desea acceder a la aplicación, y no se encuentra registrado, se le da acceso a crear un nuevo proyecto de mejora. El sistema verificará que todos los campos obligatorios sean completados, en caso contrario mostrará un mensaje.
Condiciones de Ejecución: Ninguna
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Accede al vínculo de la creación de un nuevo proyecto. - El usuario deja de llenar algún campo obligatorio.
Resultado Esperado: ▪ El sistema muestra un mensaje en caso de que falte algún campo que se requiere obligatorio por llenar.
Evaluación de la Prueba: Satisfactoria

Caso de prueba de aceptación_ 05: Gestionar proyecto

	Caso de Prueba de Aceptación
Código Caso de Prueba: 05	**Número Historia de Usuario:** 3
Nombre del Caso de Prueba: Gestionar proyecto	
Descripción: La HU inicia cuando una vez creado el proyecto, el usuario desea modificar algún dato o simplemente eliminarlo.	
Condiciones de Ejecución: Ninguna	
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Accede al menú proyecto. - Se muestra los datos del proyecto. - Selecciona la opción deseada.	
Resultado Esperado: ▪ Según la opción escogida se actualiza la base de datos.	
Evaluación de la Prueba: Satisfactoria	

Caso de prueba de aceptación_ 06: Gestionar artefactos

	Caso de Prueba de Aceptación
Código Caso de Prueba: 06	**Número Historia de Usuario:** 4
Nombre del Caso de Prueba: Gestionar artefacto	
Descripción: La HU inicia cuando una vez creado el proyecto, el usuario desea subir algún artefacto a la aplicación.	
Condiciones de Ejecución: Ninguna	
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Accede al menú proyecto. - Da a examinar para escoger el artefacto - Selecciona el artefacto y lo sube.	
Resultado Esperado: ▪ Se cree una copia del artefacto que estaba en la máquina cliente dentro del directorio uploads de la aplicación web situada en el servidor y que a la vez sea generado el registro correspondiente al artefacto en la base de datos	
Evaluación de la Prueba: Satisfactoria	

Caso de prueba de aceptación_ 07: Cerrar ciclo

	Caso de Prueba de Aceptación
Código Caso de Prueba: 07	**Número Historia de Usuario:** 5
Nombre del Caso de Prueba: Cerrar ciclo	
Descripción: La HU inicia una que terminadas todas las actividades, el usuario selecciona la opción cerrar ciclo en la ventana de gestión de proyectos.	
Condiciones de Ejecución: Ninguna	
Entrada / Pasos de Ejecución: - El estado de todas las actividades es “Terminada”. - Accede al menú proyecto. - Selecciona la opción cerrar ciclo	

Resultado Esperado: ▪ Se cambia el estado del ciclo de “actual” a “cerrado” y se cree automáticamente un nuevo ciclo y sus actividades correspondientes.
Evaluación de la Prueba: Satisfactoria

Anexo IV

Casos de pruebas de aceptación de las historias de usuario de la iteración 2

Caso de prueba de aceptación_ 08: Administrar actividades

	Caso de Prueba de Aceptación
Código Caso de Prueba: 08	**Número Historia de Usuario:** 6
Nombre del Caso de Prueba: Administrar actividades	
Descripción: La HU se inicia cuando el usuario accede a la ventana de actividades por la necesidad de utilizar, crear, editar, o eliminar una actividad. El sistema muestra una ventana con todas las actividades archivadas.	
Condiciones de Ejecución: El usuario se ha autenticado y tiene los permisos para administrar las actividades.	
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Accede a la ventana de las actividades.	
Resultado Esperado: ▪ El sistema muestra una interfaz con todas las actividades registradas con las opciones de eliminar y editar. Además se da la opción de crear alguna nueva.	
Evaluación de la Prueba: Satisfactoria	

Caso de prueba de aceptación_ 09: Crear nueva actividad

	Caso de Prueba de Aceptación
Código Caso de Prueba: 09	**Número Historia de Usuario:** 6
Nombre del Caso de Prueba: Crear nueva actividad	
Descripción: La HU se inicia cuando el usuario accede a la ventana de actividades por la necesidad	

de crear una nueva actividad. El sistema muestra una nueva ventana con todos los campos requeridos para insertar una nueva actividad.
Condiciones de Ejecución: El usuario se ha autenticado y tiene los permisos para administrar las actividades.
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Accede a la ventana de las actividades. - Selecciona la opción para crear una nueva actividad. - Llena todos los campos necesarios para crear la actividad. - El sistema inserta los datos en la base de datos y comprueba que no coincidan con otra actividad.
Resultado Esperado: ▪ El sistema muestra la interfaz de las actividades con los datos de la nueva actividad creada.
Evaluación de la Prueba: Satisfactoria

Caso de prueba de aceptación_ 10: Editar actividad

	Caso de Prueba de Aceptación
Código Caso de Prueba: 10	**Número Historia de Usuario:** 6
Nombre del Caso de Prueba: Editar actividad	
Descripción: La HU se inicia cuando el usuario accede a la ventana de actividades por la necesidad cambiar los datos de alguna actividad. El sistema muestra una ventana con todos los campos requeridos a editar.	
Condiciones de Ejecución: El usuario se ha autenticado y tiene los permisos para administrar las actividades.	
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Accede a la ventana de las actividades. - Selecciona la opción editar. - Modifica los datos que desee en la actividad. - El sistema registra los nuevos datos en la base de datos.	

Resultado Esperado: ▪ El sistema muestra la interfaz de las actividades con los nuevos datos de la actividad.
Evaluación de la Prueba: Satisfactoria

Caso de prueba de aceptación_ 11: Eliminar actividad

	Caso de Prueba de Aceptación
Código Caso de Prueba: 11	**Número Historia de Usuario:** 6
Nombre del Caso de Prueba: Eliminar actividad	
Descripción: La HU se inicia cuando el usuario accede a la ventana de actividades por la necesidad de eliminar alguna actividad. El sistema muestra la ventana con las actividades restantes.	
Condiciones de Ejecución: El usuario se ha autenticado y tiene los permisos para administrar las actividades.	
Entrada / Pasos de Ejecución: - El usuario ejecuta la aplicación. - Accede a la ventana de las actividades. - Elimina la actividad deseada. - El sistema registra la acción en la base de datos.	
Resultado Esperado: ▪ El sistema muestra la interfaz con las actividades restantes.	
Evaluación de la Prueba: Satisfactoria	

Caso de prueba de aceptación_ 12: Activar área de proceso

	Caso de Prueba de Aceptación
Código Caso de Prueba: 12	**Número Historia de Usuario:** 7
Nombre del Caso de Prueba: Activación de las áreas de procesos	
Condiciones de Ejecución: El usuario se ha autenticado y tiene los permisos para navegar por el sistema	
Entrada / Pasos de Ejecución: ▪ El usuario ejecuta la aplicación	

▪ Accede a la página de procesos ▪ Activa área de proceso ▪ Se asigna un responsable
Resultado Esperado: ▪ Mostrar la plantilla básica de actividades para área de proceso.
Evaluación de la Prueba:

Caso de prueba de aceptación_ 13: Gestionar Usuario

	Caso de Prueba de Aceptación
Código Caso de Prueba: 13	**Número Historia de Usuario:** 8
Nombre del Caso de Prueba: Crear usuario	
Condiciones de Ejecución: Solo tiene derecho a esta opción el usuario Jefe de Proyecto	
Entrada / Pasos de Ejecución: ▪ El usuario ejecuta la aplicación ▪ Accede a la página de procesos o de actividades ▪ Activa el link de crear usuario ▪ Se muestra una nueva ventana con el formulario a llenar ▪ Se introducen todos los datos que se exigen correctamente ▪ El sistema registra al nuevo usuario en la base de datos.	
Resultado Esperado: ▪ El usuario podrá acceder al sistema y tendrá navegación en el mismo según sus privilegios.	
Evaluación de la Prueba:	

Caso de prueba de aceptación _ 14: Avance del proyecto

	Caso de Prueba de Aceptación
Código Caso de Prueba: 14	**Número Historia de Usuario:** 10
Nombre del Caso de Prueba: Mostrar avance del proyecto	
Condiciones de Ejecución: Normales	
Entrada / Pasos de Ejecución: ▪ El usuario ejecuta la aplicación	

▪ Accede a la página de avance del proyecto ▪ Se le da la opción de calcular las métricas del estado que desee
Resultado Esperado: ▪ Mostrar el avance del proyecto, mediante los resultados de las métricas calculadas.
Evaluación de la Prueba:

GLOSARIO DE TÉRMINOS

- **Actividades:** Es la suma de tareas, normalmente se agrupan en un procedimiento para facilitar su gestión. La secuencia ordenada de actividades da como resultado un subproceso o un proceso.
- **Bases de datos:** Es un conjunto de ficheros que contienen datos y los programas que gestionan la estructura y la forma en la que éstos se almacenan, así como la forma en la que deben relacionarse entre sí. Algunos ejemplos de sistemas de bases de datos, son: Access, Oracle, SQL, Parados, dBase, etc.
- **Etiqueta:** Una etiqueta será un texto incluido entre los símbolos *menor que* **<** y *mayor que* >.
- **CMMI:** Modelo Integrado de Madurez de la Capacidad
- **COBIT:** Objetivos de Control para la Información y la Tecnología Relacionada.
- **Control:** Se define como las políticas, procedimientos, prácticas y estructuras organizacionales diseñadas para brindar una seguridad razonable que los objetivos de negocio se alcanzarán, y los eventos no deseados serán prevenidos o detectados y corregidos.
- **Fases:** Representa un ciclo de desarrollo en la vida de un producto de software.
- **IDEAL:** Modelo propuesto por CMMI. Guía de usuario para la mejora de proceso de software.
- **IEEE:** Instituto de Ingenieros Eléctricos y Electrónicos.
- **Línea Base:** Conjunto de reglas de negocio que indican cómo y de qué forma se realiza el proceso de desarrollo de software.
- **Metodología:** Es un proceso que define quién debe hacer qué, cuándo y cómo debe hacerlo.
- **Mejora:** Adelantar, acrecentar una cosa, haciéndola pasar de un estado bueno a otro mejo.
- **Métricas:** Son un buen medio para entender, monitorizar, predecir y probar el desarrollo software y los proyectos de mantenimiento.

GLOSARIO DE TÉRMINOS

- **Modelo de Procesos:** Es un conjunto estructurado de elementos que describen características de procesos efectivos y de calidad. Un modelo indica ¨Qué hacer¨, no ¨Cómo hacer¨, ni ¨Quién lo hace¨.

- **Monitoreo:** Proceso de verificación periódica de la situación de un programa, para determinar si las actividades se está n cumpliendo en la forma planeada.

- **Objetivo de Control:** Un objetivo de control de TI es una declaración del resultado o fin que se desea lograr al implantar procedimientos de control en una actividad de TI en particular. Los objetivos de control de COBIT son los requerimientos mínimos para un control efectivo de cadaproceso de TI.

- **Proceso:** Un proceso se define como un conjunto de tareas, actividades o acciones interrelacionadas entre sí que, a partir de una o varias entradas de información, materiales o de salidas de otros procesos, dan lugar a una o varias salidas también de materiales (productos) o información con un valor añadido.

- **Producto:** Es cualquier cosa que puede ser ofrecida al mercado para su compra, para su utilización o para su consideración. Es cualquier bien, servicio o idea capaz de motivar y satisfacer a un comprador.

- **Proyecto:** Elemento organizativo a través del cual se gestiona el desarrollo del software, el resultado de un proyecto es una versión del producto.

- **Página Web:** una de las páginas que componen un sitio de la World Wide Web. Un sitio web agrupa un conjunto de páginas afines. A la página de inicio se la llama "home page".

- **Plugin:** Un plugin es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, generalmente muy específica, como por ejemplo servir como driver (controlador) en una aplicación, para hacer así funcionar un dispositivo en otro programa.

- **Servidor:** Computadora central de un sistema de red que provee servicios y programas a otras computadoras conectadas. Éste término define la relación entre dos programas de computación en

el cual uno, el cliente, solicita un servicio al otro, el servidor, que satisface el pedido. En la web, un servidor web es un ordenador que usa el protocolo http para enviar páginas web al ordenador de un usuario cuando el usuario las solicita.

- **Scripts:** un script es un guión o conjunto de instrucciones.
- **Triggers:** Son objetos relacionados con tablas y almacenados en la base de datos que se ejecutan o se muestran cuando sucede algún evento sobre sus tablas asociadas.
- **Tags:** Los meta tags son unos caracteres pertenecientes al HTML que se deben de escribir dentro del tag general <head> y que lo podemos definir como líneas de código que indican a los buscadores que le indexan por qué términos debe ser encontrada la página.
- **SPI:** Mejora de Procesos de software.
- **SEI:** Instituto de Ingeniería de Software.
- **SPICE:** Mejora de Proceso de software y Determinación de Capacidad.
- **UCI:** Universidad de las Ciencias Informáticas.