Universidad de las Ciencias Informáticas

Facultad 2

Título: Sistema Basado en Casos para contribuir a la disminución de los costos de la calidad de los proyectos del Centro de Telemática.

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

Autora:
Elaine López Rivero.

Tutores:
Ing. Aymara Marín Díaz

Ing. Dasiel Cordero Morales

Ing. Denys Buedo Hidalgo

La Habana, junio de 2013.
"Año 55 de la Revolución"

Declaración de autoría

Declaro ser la autora del presente trabajo de diploma y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de junio del año 2013.

_____ _____

Firma del Autor Firma del Tutor

Elaine López Rivero Aymara Marín Díaz

_____ _____

Firma del Tutor Firma del Tutor

Denys Buedo Hidalgo Dasiel Cordero Morales

Resumen

Los costos de la calidad son considerados una herramienta fundamental dentro del campo de la gestión de la calidad. Constan de dos componentes, lo que se invierte en obtener buena calidad y lo que se paga por no lograrla. Su principal importancia estriba en que indican dónde será más provechosa una acción correctiva para una empresa.

El Centro de Telemática de la Universidad de las Ciencias Informáticas ha definido un procedimiento para el cálculo de los costos de la calidad. Dicho procedimiento no permite identificar en cuáles de los indicadores de costos específicos para cada una de las actividades de calidad se ha incurrido en costos desfavorables, lo cual imposibilita llevar a cabo acciones para disminuirlos.

En el presente trabajo se propone un modelo computacional que empleando el Razonamiento Basado en Casos permite identificar los indicadores de costos de calidad afectados en los procesos de gestión de la calidad de los proyectos del Centro de Telemática. Además, se presentan los artefactos fundamentales generados como parte de la implementación y diseño del sistema. Finalmente, se exponen los resultados de la aplicación de dicho sistema en 3 de los 6 proyectos de desarrollo existentes en el centro.

PALABRAS CLAVE: costos, calidad, disminuir, Sistemas Basados en Casos.

Índice

Índice de Figuras

Índice de Tablas

Introducción

La búsqueda y el afán de la perfección han sido constantes del hombre a través de la historia y la calidad una de sus manifestaciones. [1] La concepción de la misma ha evolucionado con el paso de los años. Hasta la mitad del siglo XX la calidad era vista como un problema que se solucionaba mediante herramientas de inspección. Luego se enfocó en el proceso, a través del Control Estadístico de la Calidad el cual constituía la base de todos los sistemas de calidad y se aplicaba sobre muestras representativas de lotes de productos. En los años sesenta, los departamentos de calidad tenían como función el aseguramiento de la misma, basado en considerar a la calidad como algo de lo que todos eran responsables, teniendo un fuerte desarrollo. Las tendencias actuales consideran a la calidad como un factor estratégico, ‘‘ya no se trata de una actividad inspectora sino preventiva: planificar, diseñar, fijar objetivos, educar e implementar un proceso de mejora continua, la gestión estratégica de la calidad hace de esta una fuente de ventajas competitivas que requiere del esfuerzo colectivo de todas las áreas y miembros de la organización’’. [2] Los procesos de la gestión de la calidad abarcan la planificación, el control, el aseguramiento y el mejoramiento de la misma.

Hoy en día no se puede hablar de gestión de la calidad total sin tener en cuenta los costos de esta. Actualmente, se entienden como costos de calidad los incurridos en el diseño, implementación, operación y mantenimiento de los sistemas de calidad de una organización, los costos de los procesos de mejoramiento continuo de la calidad y los costos de sistemas, productos y servicios que han fracasado al no tener en el mercado el éxito que se esperaba. [3]

Los primeros autores que reconocieron los costos de calidad fueron Miner (1933) y Crockett (1935) en la década de los 30, pero no es hasta finales de los años cincuenta y comienzos de los sesenta cuando diversos autores muestran un creciente interés sobre este tema. Así Juran, en [4] hace referencia al término “costes de calidad”, resaltando la importancia de medir y controlar los costos evitables de la calidad, como oro en la mina que debe de ser extraído. Los trabajos de Masser (1957), Freeman (1960) y de Feingenbaum (1961) establecen las primeras clasificaciones de los costos de calidad. A principios de los 60, la multinacional ITT (International Telephone and Telegraph) es una de las primeras empresas que empiezan a calcular los costos de calidad. [4]

En 1961 la American Society For Quality Control (ASQC) creó el Comité de Costos de Calidad, y en diciembre de 1963 se promulgan por el Ministerio de Defensa de los EEUU las especificaciones militares MIL–Q–9858-A sobre los “requisitos del programa de calidad”. El mismo se constituyó con el objetivo de alertar, a través del seguimiento de los costos de

calidad, sobre la importancia que tiene la calidad para asegurar la supervivencia de las empresas. En 1967 publican su primer documento [5], donde establecían el contenido que debería tener un programa de costos totales de calidad; también se definían los conceptos de los elementos integrantes de los costos de calidad por categorías, siguiendo la clasificación establecida anteriormente por Feingenbaum (1956).

Cuba ante la imperiosa necesidad de insertarse en el mercado mundial trabaja en la búsqueda de nuevas alternativas y vías para un mejor aprovechamiento de sus recursos humanos y materiales. Tiene como pilar una efectiva gestión de los costos de la calidad, herramienta para lograr la eficiencia en los procesos productivos y de prestación de servicios, cuya principal importancia estriba en que indican donde será más provechosa una acción correctiva para una organización.

Las empresas en su desempeño diario generan un gran cúmulo de información, la cual resulta un recurso de vital importancia en disímiles procesos. Uno de ellos es el proceso de toma de decisiones basado en la identificación y selección de la acción adecuada para un problema determinado. Los problemas de toma de decisiones de acuerdo a la información y las características del dominio se clasifican en: estructurados y no estructurados. [6] Los primeros hacen referencia a aquellos cuya solución se sustenta en modelos o métodos cuantitativos. El segundo grupo enmarca los problemas que carecen de esta formulación y que frecuentemente es necesario recalcularlos mediante procesos iterativos y de búsqueda. Es precisamente en esta clase de problemas de toma de decisiones donde las técnicas de Inteligencia Artificial juegan un papel preponderante.

Los Sistemas Basados en el Conocimiento son una técnica de Inteligencia Artificial que ha tenido gran repercusión y desarrollo en esta área de la toma de decisiones. Estos sistemas utilizan conocimiento sobre un dominio específico. La solución que se obtiene es similar a la alcanzada por una persona experimentada en el dominio del problema. Resultaría muy ventajoso utilizar esta potencial técnica en la gestión de los costos de la calidad para el apoyo a la toma decisiones sobre qué medidas o acciones correctivas implementar para disminuir los costos de la calidad.

En esta necesidad por lograr una calidad superior está inmersa la Universidad de las Ciencias Informáticas (UCI). La misma tiene como misión, además de formar profesionales comprometidos con la Revolución y altamente calificados en la informática, la producción de aplicaciones y servicios profesionales. Posee un modelo de formación, que se sustenta en la interrelación de la formación, la investigación y la producción, además de servir de soporte, a la industria cubana de la informática. [7] La actividad de desarrollo–producción de la UCI se

soporta sobre una Red de Centros, tres de los cuales se encuentran certificados con el nivel 2 de CMMI. El Centro de Telemática, forma parte de dicha red y tiene como misión el desarrollo de sistemas y servicios informáticos en las ramas de las Telecomunicaciones y la Seguridad Informática. Dicho centro ha dado pasos en el estratégico campo de la calidad del software al definir un procedimiento [8] que permite calcular los costos de la calidad de los proyectos de investigación y desarrollo, a partir de un conjunto de indicadores que estiman actividades relacionadas con la planificación y control de la calidad. Este procedimiento se realiza de forma manual, lo que conlleva a grandes pérdidas de tiempo al realizar los cálculos y da margen a que se cometan errores en su aplicación. El mismo no permite identificar los indicadores de costos específicos que resultan desfavorables para cada una de las actividades de calidad. Actualmente, no se tiene en cuenta la experiencia acumulada de aplicar dicho procedimiento, aspecto que puede resultar de gran importancia para ejecutar acciones o medidas que permitan disminuir dichos indicadores.

En entrevistas realizadas a líderes y especialistas del centro se evidenció la falta de claridad por parte del equipo de desarrollo para determinar los indicadores de costos de calidad que resultan más perjudiciales, de ahí que no se puedan ejecutar acciones correctivas pertinentes para su mitigación. Esta situación puede ocasionar atrasos en los cronogramas de los proyectos, el incumplimiento de los requerimientos pactados y la entrega de un producto al cliente sin la calidad necesaria. Al mismo tiempo, la experiencia adquirida por los especialistas sobre los costos de la calidad no es documentada, almacenada o estructurada por lo que con la pérdida del personal se produce a su vez la pérdida del conocimiento. Debido a estos elementos, se puede manifestar la disminución de los índices de satisfacción de los clientes lo que atentaría contra la credibilidad de la institución ante la sociedad. Otra consecuencia se podría observar en el factor económico, pues además de disminuir los ingresos a partir de la pérdida de clientes se generarían gastos adicionales por la incorrecta planificación de las actividades que se llevan a cabo en el proceso de desarrollo de software.

A partir de la problemática anterior se plantea el siguiente **problema a resolver:**

¿Cómo identificar los indicadores de costos de calidad afectados y contribuir a la disminución de los mismos durante los procesos de gestión de la calidad en el Centro Telemática haciendo uso del conocimiento y la experiencia que se genera de estos procesos?

Atendiendo al problema de investigación se define el **objeto de estudio**: Los procesos de la gestión de la calidad.

La presente investigación tiene como **objetivo general:** Desarrollar un Sistema Basado en el Conocimiento que permita identificar y contribuir a la disminución de los indicadores de costos

calidad afectados durante los procesos de gestión de la calidad en los proyectos del Centro de Telemática.

De dicho objetivo general se derivan los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación, a partir del análisis de los antecedentes a nivel nacional e internacional de las herramientas existentes para la gestión de los costos de la calidad.
- ✓ Formalizar un modelo computacional que utilizando los Sistemas Basados en el Conocimiento permita identificar los indicadores de costos de calidad afectados en los procesos de gestión de la calidad.
- ✓ Desarrollar el modelo computacional que permite identificar los indicadores de costos de calidad afectados en los procesos de gestión de la calidad.
- ✓ Validar el sistema que permite identificar y contribuir a la disminución de los indicadores de costos de calidad afectados en los procesos de gestión de la calidad.

El **campo de acción** lo constituyen: los sistemas basados en el conocimiento para la gestión de los costos de la calidad.

Para dar cumplimiento a los objetivos específicos se trazan las siguientes **Tareas de Investigación:**

- ✓ Análisis del procedimiento que permite calcular los costos de calidad en los proyectos del Centro de Telemática para sintetizar los elementos de utilidad en la implementación del sistema.
- ✓ Selección de las herramientas, las tecnologías y la metodología de desarrollo de software a utilizar en el desarrollo de la aplicación.
- ✓ Análisis de los diferentes Sistemas Basados en el Conocimiento para seleccionar el óptimo en el desarrollo del sistema.
- ✓ Definición de la estrategia para la realización de pruebas al sistema.

De acuerdo a todo lo antes expuesto se propone la siguiente **idea a defender:**

El desarrollo de un sistema basado en el conocimiento permitirá la identificación de los indicadores de costos de calidad afectados y contribuirá a la disminución de los mismos en los proyectos de desarrollo de software del Centro de Telemática de la Universidad de las Ciencias Informáticas.

Métodos teóricos

Histórico-lógico: En la primera parte de la investigación se desarrollará un estudio del estado del arte de la problemática; así como se analizarán las ventajas y desventajas de cada una de las herramientas utilizadas actualmente para la gestión de los costos de la calidad.

Método analítico–sintético: Se utilizará para el análisis de la información existente sobre: los Sistemas Basados en el Conocimiento, el Procedimiento para el cálculo de los costos de calidad del Centro de Telemática y las herramientas a emplear en el desarrollo del sistema. Luego como parte de la aplicación del método se determinarán los puntos esenciales de cada elemento objeto de análisis y se sintetizarán en aspectos relevantes para la investigación.

Métodos empíricos:

Entrevista: Se utilizará la entrevista como una conversación planificada con especialistas y líderes del Centro de Telemática para obtener información acerca del problema en cuestión. Su uso constituye un medio para el conocimiento cualitativo de las características particulares de un proceso y puede influir en el posterior análisis y diseño del producto de software que contribuirá a la disminución de los costos de la calidad de dicho centro.

El siguiente trabajo de diploma está estructurado en 3 capítulos, a continuación se muestra una breve descripción de cada uno de ellos:

Capítulo 1 denominado "*Fundamentación teórica*", donde se incluyen los resultados del estudio sobre el estado del arte de herramientas existentes en Cuba y en el resto del mundo para la gestión de los costos de la calidad; así como el análisis de la metodología, tecnologías, técnicas y herramientas a utilizar en el desarrollo de la aplicación .Se exponen además los conceptos fundamentales para una mejor comprensión de la investigación.

Capítulo 2 denominado "*Características del Sistema*", donde se expone la propuesta del sistema así como se enumeran los requisitos funcionales y no funcionales del mismo. Se modelan y describen los casos de uso del sistema.

Capítulo 3 denominado "*Diseño, Implementación y Pruebas*", donde se presentan los diagramas de clases del diseño y la realización de los Casos de Uso a través de los diagramas de interacción, específicamente los de secuencia. También se definen los patrones de diseño a emplear. Se muestra la organización de las clases en términos de componentes así como la distribución física del sistema mediante el diagrama de despliegue. Se describe la propuesta del diseño de Casos de Prueba basado en Casos de Uso y el modelo computacional sustentado en técnicas de Inteligencia Artificial para identificar los indicadores de costos de calidad desfavorables en los procesos de gestión de la calidad del Centro Telemática.

Capítulo 1. Fundamentación Teórica

Introducción

En el presente capítulo se exponen los principales conceptos relacionados con los costos de calidad y los Sistemas Basados en el Conocimiento (SBC). Se realiza un análisis de estos sistemas así como de sus antecedentes en la gestión de la calidad. Además se lleva a cabo un estudio sobre las herramientas y tecnologías a emplear en el desarrollo de la aplicación.

1.1 Calidad

Son muchos los autores que han expresado lo difícil de una definición de calidad, el diccionario de la lengua española define este vocablo en los siguientes términos: “Propiedad o conjunto de propiedades inherentes a una cosa, que permite apreciarla como igual, mejor o peor que las restantes de su especie”. El Dr. Ishikawa en [9] define calidad como: “En su interpretación más estrecha, calidad significa calidad del producto, pero en su interpretación más amplia significa calidad del trabajo, calidad del servicio, calidad de la información, calidad del proceso, calidad de la dirección, calidad de la empresa”. Según lo que plantea la norma ISO 9000:2000 [10] la calidad: “Es el grado en el que un conjunto de características (rango diferenciador) inherentes cumple con los requisitos (necesidad o expectativa establecida, generalmente implícita u obligatoria)”. Juran [11] plantea que la calidad de un producto o servicio, es la caracterización del artículo o servicio obtenido en el proceso de producción o servicio que determina el grado de su correspondencia con el conjunto de exigencias establecidas por la documentación técnica y los consumidores. Para Deming [12], la calidad no es más que "Una serie de cuestionamientos hacia una mejora continua”.

Podría concluirse entonces que la calidad constituye un indicador de comparación entre productos o servicios de la misma índole. Para lograr una alta calidad todos los esfuerzos disponibles no deben ser invertidos solamente en el producto o servicio final, sino en cada una de las etapas del proceso de desarrollo de los mismos. La medida en que se distribuyen estos esfuerzos es de gran importancia para mantener la competitividad de las empresas. La mayor cantidad de recursos deberá recaer en la prevención, con el objetivo de que en etapas avanzadas los errores y deficiencias no constituyan un obstáculo imposible de franquear en la carrera por la calidad total.

1.1.1 Calidad del software

Según Pressman la calidad del software es la “Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente”. [13]

La calidad depende de la medida en se hayan satisfecho las solicitudes y requerimientos de los clientes. En la industria de desarrollo del software esta máxima no es la excepción .De una forma u otra en todas las metodologías de desarrollo de software se sigue un proceso de levantamiento de requisitos en el cual se captan las precisiones del cliente sobre el software que se desea obtener. Para dar cumplimiento a estos requisitos con la mayor calidad posible los equipos de desarrollo de software profesionales se basan en estándares mundiales de calidad como el Modelo de Madurez de Capacidad Integrada (del inglés, Capability Maturity Model Integration, CMMI) y las normas ISO. Estos estándares abarcan la experiencia acumulada en el campo de la gestión de la calidad, por lo que contituyen un soporte vital en el proceso de desarrollo de software.

La UCI optó por la selección del estándar CMMI. El mismo constituye un marco de referencia de la capacidad de las organizaciones de desarrollo de software en el desempeño de sus diferentes procesos. Proporciona una base para la evaluación de la madurez de los mismos y es una guía para implementar una estrategia para la mejora continua.

La calidad del software puede medirse después de elaborado el producto, pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.

Las dimensiones clave de la calidad del software son las siguientes [14]:

- **Nivel de satisfacción:**

  Grado en el cual los clientes o usuarios perciben que el producto de software cumple sus necesidades, requerimientos y expectativas.

- **Valor del producto:**

  Grado en el cual un producto de software tiene valor para sus involucrados, en relación al ambiente competitivo.

- **Atributos de calidad:**

  El grado en el cual un producto de software posee una combinación de propiedades deseadas (eficiencia, integridad, portabilidad, entre otras). Existen diversas taxonomías de los atributos de calidad, entre ellas se encuentran las propuestas en el Handbook of

Software Quality Assurance [15] que considera: a) calidad de diseño (correcto, facilidad de mantenimiento, facilidad de verificación), b) calidad de desempeño (eficiencia, integridad, fiabilidad, facilidad de uso, facilidad de prueba), c) calidad de adaptación (facilidad de expansión, flexibilidad, portabilidad, reusabilidad, interoperabilidad).

- **Defectos:**

  Grado en el cual el producto de software presenta fallas al utilizarse, atribuibles a este.

- **Calidad del proceso:**

  En relación al proceso de desarrollo por el cual fue construido el software.

Lograr todas estas dimensiones de la calidad es considerablemente complejo teniendo en cuenta que el software es un producto intangible. Los defectos son una consecuencia indeseable del error humano. En cada etapa de desarrollo se producen errores, es por ello que las pruebas y el aseguramiento de la calidad son actividades necesarias en los proyectos de desarrollo de software.

Al contrastar el costo de encontrar errores en cada fase del desarrollo contra el costo incurrido de descubrirlos en una fase posterior, se demuestra que una inversión temprana en prevención y descubrimiento de errores, mediante aseguramiento de la calidad, estándares, revisiones, auditorías e inspecciones, permiten remover los defectos con un costo significativamente menor [16].

1.2 Gestión de la calidad

Se considera a la gestión de la calidad como, el conjunto de actividades de la función empresarial que determina la política de calidad, los objetivos y las responsabilidades y las implementa mediante los procesos de: planificación de la calidad, control de la calidad, aseguramiento de la calidad y mejoramiento de la calidad, en el marco del sistema de la calidad. [17] Para alcanzar una gestión de la calidad efectiva reviste gran importancia la aplicación del ciclo de Deming. El mismo constituye una de las principales herramientas para lograr la mejora continua en las organizaciones o empresas que desean alcanzar la excelencia en sus sistemas de calidad. Este ciclo, también conocido por sus siglas en español PHVA (Planificar, Hacer, Verificar y Actuar) debe incorporarse al modo habitual de trabajo en la organización y a su cultura, siendo asumida como un valor fundamental. Además debe constituir un estilo de gestión, pensamiento y actuación cuya principal característica es que no finaliza nunca. Emprender acciones puntuales, destinadas a subsanar determinados problemas, por muy efectivas que estas pudieran ser, no es suficiente. [18] En la Figura 1 se muestran cada una de las etapas por la que transita este ciclo.

Figura 1: Ciclo de Deming

Se ha logrado establecer que al utilizar los principios de Deming la calidad aumenta, por lo tanto bajan los costos y los ahorros se le pueden pasar al consumidor. Cuando los clientes obtienen productos de calidad, las compañías logran aumentar sus ingresos y al lograr esto, la economía crece. [17] El mejoramiento continuo permite visualizar un horizonte más amplio, donde se buscará siempre la excelencia y la innovación que llevarán a los empresarios a aumentar su competitividad, disminuir los costos, orientando los esfuerzos a satisfacer las necesidades y expectativas de los clientes. [19]

En la ciencia, para alcanzar nuevos triunfos y mantenerse en la élite se ha de ser inconformes, perseguir nuevos horizontes. El científico que pasa el tiempo ideando, experimentando, es el que alcanza mayor cantidad de resultados. Así también debe ser el desempeño de las empresas en sus procesos productivos o de prestación de servicios. Trabajar incansablemente en la mejora continua es un pilar del que han sido partícipes los ganadores a través de la historia.

El cumplimiento del objetivo que se traza en esta investigación permitirá lograr a su vez una mayor calidad del software que se desarrolla en el Centro deTelemática. Este sistema podrá ser empleado en cada una de las etapas que propone el ciclo de Deming, de manera que permitirá conocer el estado de los costos de la calidad durante todo el proceso de desarrollo de cada uno de los proyectos del centro. Además, identificará las actividades de calidad en las que

se ha incurrido en costos desfavorables y propondrá medidas o acciones como soporte a la toma de decisiones para disminuir estos costos.

1.3 Costos Totales de la Calidad

Actualmente, se entienden como Costos Totales de la Calidad aquellos incurridos en el diseño, implementación, operación y mantenimiento de los sistemas de calidad de una organización, así como los comprometidos en los procesos de mejoramiento continuo de la calidad y los de sistemas, productos y servicios que han fracasado al no tener en el mercado el éxito que se esperaba [3]. El costo de la calidad tiene dos componentes, lo que se invierte en obtener buena calidad y lo que se paga por no lograrla. El primer componente es decidido y controlado por la institución; el segundo se manifiesta en las fallas del producto. Se invierte en tener buena calidad mediante prevención (evitar errores) y evaluación (verificar que no haya errores). Por otro lado, las fallas pueden ser de dos tipos: internas (las que encuentran los desarrolladores) y externas (las que encuentran los clientes). Al costo resultante de sumar todas las categorías se le conoce como Costos Totales de la Calidad [4].

Los costos de calidad se clasifican en:

Costos de prevención

Son todos aquellos costos en que incurre la empresa para prevenir errores y conseguir que los trabajos se planifiquen, se elaboren y se controlen con buena calidad, es decir, son todos aquellos gastos que se realizan con el objetivo de evitar o minimizar los defectos o fallos que se puedan producir posteriormente. [21]

Costos de evaluación

Son los costos relacionados con la medición, evaluación o auditoría de los productos o servicios para asegurar el cumplimiento de las normas de calidad y los requisitos del desempeño. [22]

Costos de fallas

Son los costos resultantes de los productos o servicios que no se ajustan a las especificaciones o a las necesidades de los clientes o usuarios. Se dividen en fallas internas y externas. [21]

- **Fallas internas**

  Son todos aquellos costos en que incurre la empresa para corregir fallos, defectos y errores encontrados en el producto o servicio durante las evaluaciones, antes de que el producto o servicio haya sido entregado al cliente.

- **Costos de fallas externas**

  Son los costos en que se incurre como resultado de defectos encontrados después de la entrega o envío de un producto o servicio determinado al cliente. [22]

El manejo de costos de calidad comienza con la comprensión y el conocimiento de que mejorar el ejercicio de la calidad y el costo de la misma, son sinónimos. El próximo paso es reconocer que el mejoramiento de la calidad cuantificable puede tener un efecto palpable en otras valuaciones del negocio. El costo de calidad debe ser medido y debe reflejar el costo de oportunidades perdidas para la empresa. Existe un grave peligro en responder a un problema de consumidor, solamente añadiendo operaciones internas. Esto puede solucionar el problema del cliente, pero el costo añadido puede destruir el potencial de utilidades. [23]

La medición de costo de calidad proporciona una guía al programa de gestión de la calidad, así como el sistema de costo de cuentas lo da a la administración general. Su función es la de definir y cuantificar los costos que se ven directamente afectados, tanto positiva como negativamente por el programa de administración de calidad, con el fin de hacer un manejo más eficiente de la calidad. [23]

Varios estudios, autores y empresas señalan que los costos totales de calidad representan alrededor del 5 al 25 % sobre las ventas anuales. Alrededor del 95% de los costos de la calidad se desembolsan para cuantificar la calidad así como para estimar el costo de las fallas. [23] Estos gastos se suman al valor de los productos o servicios que paga el consumidor, el cual los percibe en el precio. Al corregir las reelaboraciones, incumplimientos y fallas de manera general los precios disminuyen por lo que la empresa resulta más competitiva.

1.4 Modelo de Madurez de Capacidad Integrada

El CMMI es el marco de referencia que la UCI seleccionó para mejorar sus procesos de desarrollo de productos y servicios. Consiste en las mejores prácticas que tratan las actividades de desarrollo y de mantenimiento que cubren el ciclo de vida del producto, desde la concepción hasta la entrega y soporte. Este modelo de madurez le permite aproximarse a la mejora de procesos y a las evaluaciones usando dos representaciones diferentes: continua y por etapas.

La representación continua ofrece la máxima flexibilidad cuando se utiliza el CMMI para la mejora de procesos. Una organización puede elegir mejorar el rendimiento de un punto problemático relacionado con un solo proceso, o puede trabajar en varios dominios que están fuertemente alineados con sus objetivos estratégicos. La representación continua también permite que una organización mejore diferentes procesos a diferentes niveles. Las

dependencias que existen entre algunas áreas de proceso pueden, sin embargo, limitar un poco las elecciones. [20]

La representación por etapas ofrece una manera sistemática y estructurada de aproximarse a la mejora de procesos basada en el modelo etapa a etapa. El logro de cada etapa asegura que una infraestructura de proceso adecuada se ha establecido como fundamento para la etapa siguiente. [20]

Las áreas de proceso están organizadas por niveles de madurez, los cuales eliminan interpretaciones a la mejora de los procesos. La representación por etapas prescribe un orden para implementar las áreas de procesos según unos niveles de madurez, que determinan el camino seguido por una organización para pasar del nivel inicial al nivel "en optimización". Alcanzar cada nivel de madurez asegura que se ha establecido un fundamento adecuado para el siguiente nivel, lo que permite una mejora incremental y duradera. [19]

Para alcanzar un nivel superior se deben vencer todos los anteriores, por ejemplo: para certificar el nivel tres de CMMI se requiere tener implementado correctamente todos los requerimientos del nivel uno, dos y tres del modelo. En la Figura 2 se representan los 5 niveles de madurez de CMMI.

Figura 2: Niveles de madurez de CMMI

Un proceso de nivel de capacidad 2 se caracteriza como un "proceso gestionado". Un proceso gestionado es un proceso realizado (nivel de capacidad 1) que tiene la infraestructura básica dispuesta para soportar el proceso. Se planifica y ejecuta de acuerdo a políticas; emplea personal con habilidades; tiene los recursos adecuados para producir resultados controlados; involucra a las partes interesadas relevantes; se monitoriza, controla, revisa y evalúa la

adherencia a su descripción de proceso. La disciplina de proceso reflejada por el nivel de capacidad 2 ayuda a asegurar que las prácticas existentes se mantienen durante tiempos de estrés. [20]

La Universidad de las Ciencias Informáticas para la aplicación de este modelo de madurez seleccionó la representación por etapas y actualmente, solo tres de los centros que forman parte de la estructura productiva de la universidad, han alcanzado el segundo nivel de CMMI en sus procesos de desarrollo. El resto de los centros, entre ellos el de Telemática, centran sus esfuerzos en alcanzar esta meta. El presente trabajo pretende contribuir a la mejora de la implementación del área de proceso Aseguramiento de la Calidad de Productos y Procesos (PPQA), en dicho centro. Esta área, tiene como propósito fundamental proporcionar la visión objetiva sobre el aseguramiento de la calidad al personal y a la dirección, en los procesos y los productos de trabajo asociados. Para ello, se propone un sistema que permite conocer el estado de los indicadores de costos de calidad en los procesos de gestión de la calidad para cada proyecto del centro, con el objetivo de ejecutar acciones correctivas en caso de resultar desfavorables.

1.5 Procedimiento para determinar los Costos Totales de la Calidad en los proyectos de software del Centro de Telemática

Con el propósito de realizar la informatización de los cálculos de los costos de calidad y de disminuir los mismos se realizó un estudio de la investigación titulada: “Propuesta de Procedimiento para determinar los Costos Totales de la Calidad en los proyectos de software del Centro de Telemática de la UCI” [8]. El procedimiento que esta investigación propone cuenta con cinco etapas, las cuales se muestran a continuación:

Etapa 1: Diagnóstico de la situación actual del proyecto.

Etapa 2: Identificación y clasificación de los Costos Totales de la Calidad.

Etapa 3: Análisis y cálculo de los Costos Totales de la Calidad.

Etapa 4: Reporte de mejoramiento.

Etapa 5: Presentación de los resultados de los Costos Totales de la Calidad a la dirección del proyecto.

En la **Etapa 3** es donde se procede al cálculo de los Costos Totales de la Calidad por cada una de las categorías de costos antes mencionadas. Con las actividades por cada categoría de costos, se procede a realizar los cálculos de los costos de la calidad y los de la mala calidad. Dentro de los cálculos de la calidad se determinará el Esfuerzo Absoluto de la Calidad y el

Esfuerzo Relativo de la Calidad, para ello se utilizan métricas. Una métrica es una forma de medir, una escala, definida para realizar mediciones de uno o varios atributos.

Los cálculos que incluye esta etapa por cada categoría de costos, más los costos de calidad y mala calidad, son los cálculos a informatizar por el sistema.

En este caso para calcular estos últimos se utilizan las siguientes métricas:

Donde:

P: Prevención**, E:** Evaluación, **Fi:** Fallas internas, **Fe:** Fallas externas, **C:** Creación.

Estos costos se expresan en horas para poderlos comparar fundamentalmente en el tiempo y analizar tendencias.

Esfuerzo de Calidad (EoQ)

$$EoQ = (P+E+Fi+Fe)$$

El Costo Absoluto de la Calidad o Esfuerzo de la Calidad (**EoQ**) es la sumatoria de los costos de las actividades de la calidad.

Costo de Calidad (CoQ)

$$CoQ = (P+E+Fi+Fe) / (P+E+Fi+Fe+C)$$

El costo relativo de la calidad o costo de calidad (CoQ) es el porciento que representa este costo absoluto de la calidad del total de horas dedicadas a la calidad, más las horas dedicadas a la creación. Los costos de creación son aquellas actividades de creación que aportan valor al producto por ejemplo: Levantamiento de requisitos, las actividades de implementación entre otras.

Interpretación del CoQ:

Si el valor de CoQ es ≥ 50%, los costos de la calidad se consideran no adecuados.

Si el valor de 25% ≤ CoQ < 50%, los costos de la calidad se consideran adecuados.

Si CoQ < 25% los costos de la calidad se consideran bastante adecuados.

Esfuerzo de Mala Calidad (EoPQ)

$$EoPQ = Fi+Fe$$

El esfuerzo de la mala calidad (**EoPQ**) es la sumatoria de los costos de las actividades incurridas por costos de fallas internas y externas.

Costo de Mala Calidad (CoPQ)

$$CoPQ = (Fi + Fe) / (P + E + Fi + Fe + C)$$

Si el CoPQ ≤ 5%, los costos de la mala calidad se consideran adecuados.
Si el CoPQ > 5%, los costos de la mala calidad se consideran no adecuados.
El costo de la mala calidad (CoPQ) es el porciento que representa el esfuerzo de la mala calidad del total de horas dedicadas al esfuerzo de la calidad más los costos de creación.
En el desarrollo del sistema se emplearán las fórmulas que propone este procedimiento para el cálculo de los indicadores de costos de calidad. Las mismas se muestran en el **Anexo 1.** Además se utilizarán las métricas expuestas anteriormente para conocer el estado de los costos de buena (prevención y evaluación) y mala (fallas internas y fallas externas) calidad. Este procedimiento no permite identificar los indicadores de costos de calidad que resultaron afectados. Sin embargo, el conocimiento y la experiencia acumulada en los procesos de gestión de la calidad y los casos resultantes de la aplicación del mismo resultan útiles en la construcción de un SBC que permita cubrir dicha limitante.

1.6 Inteligencia Artificial

La Inteligencia Artificial (IA) es una rama de la investigación científica dentro de las ciencias de la computación que se encarga de modelar la inteligencia humana a través de sistemas computacionales. Utiliza técnicas complejas para tratar que un ordenador simule el proceso de razonamiento humano. Pretende también que el ordenador sea capaz de modificar su programación en función de su experiencia y que aprenda. [25]
La IA se divide en dos escuelas de pensamiento, la inteligencia artificial convencional y la inteligencia computacional.
Inteligencia artificial convencional: está relacionada con métodos que actualmente se conocen como máquinas de aprendizaje, se caracteriza por el formalismo y el análisis estadístico. Entre los métodos que incluye esta rama se encuentran los Sistemas Basados en el Conocimiento y la Inteligencia Artificial Basada en Comportamientos.

Inteligencia artificial computacional: consiste en el estudio de mecanismos adaptativos para permitir o facilitar el comportamiento inteligente en sistemas complejos y cambiantes. Se basa en la utilización de heurísticas y computación Los cinco principales paradigmas de la IA computacional son las redes neuronales artificiales, la computación evolutiva, la inteligencia de enjambres (swarm intelligence), los sistemas difusos (fuzzy) y los sistemas inmunitarios artificiales. [26]

1.6.1 Sistemas Basados en el Conocimiento

Los Sistemas Basados en el Conocimiento, son programas para computadoras que utilizan conocimiento sobre un dominio de aplicación determinado para obtener una solución a un problema en este dominio. Dentro de este campo se encuentran los Sistemas Expertos (SE), los cuales tienen los mismos principios de funcionamiento y características que los SBC a excepción de una única especialización que radica en que el conocimiento que manejan los SE proviene, como su nombre lo indica, de un experto. El sistema que se propone en el presente trabajo por la naturaleza del conocimiento con el que se trata constituye un Sistema Experto. Estos sistemas simulan las cadenas de razonamiento que realiza un experto para resolver un problema de su dominio. Para conseguirlo, se dota al sistema de un conjunto de reglas, casos, modelos probabilísticos, entre otros y el sistema infiere nuevas evidencias a partir de la información previamente conocida.

Los Sistemas Basados en el Conocimiento (SBC) son un modelo computacional de más alto nivel que el paradigma de la programación convencional en el cual los sistemas están formados por tres componentes: la base de conocimiento (BC), la máquina de inferencia (MI) y la interfaz.

SBC= BC + MI + Interfaz

En la BC se almacena el conocimiento necesario para resolver los problemas del dominio de aplicación para el cual se desarrolla el SBC. De acuerdo a la forma en que se almacena el conocimiento existen diferentes tipos de SBC, entre ellos se encuentran: los Sistemas Basados en Reglas, los Sistemas Basados en Probabilidades y los Sistemas Basados en Casos.

Una vez que la BC se construye, es necesario la existencia de un programa que acceda a ese conocimiento para la inferencia y toma de decisiones en el proceso de solución del problema. Este programa controla el razonamiento y dirige la búsqueda en la BC y es el que generalmente se conoce con el nombre de máquina de inferencia. [27]

La interacción entre un sistema experto y un usuario se realiza a través de la *interfaz de usuario*. La aplicación de los Sistemas Expertos será propicia donde los expertos dispongan de conocimientos complejos en un área estrecha delimitada, donde no existan algoritmos

elaborados(o donde los existentes no puedan solucionar algún problema) y no existan teorías completas. Otro campo de aplicación es donde hay teorías pero resulta prácticamente imposible analizar todos los casos posibles mediante algoritmos y en un espacio de tiempo razonable. En estas situaciones hace falta el conocimiento que el experto ha adquirido por experiencia, para llegar a una solución en un tiempo moderado.

Ambos tipos de problemas se caracterizan por el hecho de que aunque es posible la existencia de una o más soluciones la vía no está previamente fijada. Sin embargo, el experto encuentra una solución al problema gracias a su experiencia. Mientras esta solución sea susceptible de repetición y el planteamiento del problema sea claro, existe un razonamiento que puede ser reproducido por un Sistema Experto.

Ventajas:

- Mayor disponibilidad: La experiencia está disponible para cualquier hardware de cómputo adecuado.
- Costo reducido.
- Duplicación: Pueden duplicarse ilimitadamente.
- Permanencia: La experiencia es permanente, a diferencia de lo que ocurre con los expertos humanos.
- Experiencia múltiple: El conocimiento de varios especialistas puede estar disponible para trabajar simultánea y continuamente en un problema.
- Rapidez: Algunas situaciones de emergencia pueden exigir respuestas más rápidas que las de un humano.
- Manejo de grandes volúmenes de información: Los SBC proporcionan la capacidad de trabajar con gran cantidad de información. Esta es una de las limitaciones que enfrentan los especialistas humanos que puede afectar negativamente la toma de decisiones. El analista humano puede depurar datos que no considere relevantes, mientras un *SE* debido a su gran velocidad de proceso analiza toda la información incluyendo las no útiles para de esta manera aportar una decisión más sólida.

Desventajas:

- El conocimiento es difícil de extraer de los expertos humanos.
- La representación del conocimiento en términos de una estructura de datos que una máquina pueda procesar.
- La generación de inferencias o cómo hacer uso de esas estructuras abstractas para

generar información útil en el contexto de un caso específico.

1.6.1.1 Sistema basado en reglas

Estos sistemas se usan cuando el dominio del problema es estrecho, es decir, un dominio en el que se comprende bien la teoría. Resultan muy apropiados en situaciones en las que el conocimiento que se desea representar surge de forma natural con estructura de reglas.

En la vida diaria se pudieran citar innumerables ejemplos de situaciones complejas gobernadas por situaciones deterministas: sistemas de control de tráfico, transacciones bancarias, sistemas de seguridad, entre otras. En situaciones deterministas, las relaciones entre un conjunto de objetos pueden ser representadas mediante un conjunto de reglas. El conocimiento se almacena en la base de conocimiento y consiste en un conjunto de objetos y un conjunto de reglas que gobiernan las relaciones entre esos objetos. [28] Los sistemas basados en reglas son una herramienta eficiente para tratar estos problemas.

Una regla es una afirmación lógica que relaciona dos o más objetos e incluye dos partes, la premisa y la conclusión. Cada una de estas partes consiste en una expresión lógica con una o más afirmaciones objeto-valor conectadas mediante los operadores lógicos: y, o, no. [28]

Una regla se escribe normalmente como “Si premisa entonces conclusión”. En general, ambas la premisa y la conclusión de una regla, pueden contener afirmaciones múltiples. La base de conocimiento contiene el conjunto de reglas que definen el problema y el motor de inferencia saca las conclusiones aplicando la lógica clásica de reglas.

1.6.1.2 Sistema basado en probabilidades

Los sistemas basados en probabilidades son un tipo de SBC, útil para tratar situaciones de incertidumbre. En los primeros sistemas que se hicieron eco de este tipo de problemas se eligió la probabilidad para lidiar con la incertidumbre. Dichos sistemas no fueron la solución, debido al uso incorrecto de algunas hipótesis de independencia, utilizadas para reducir la complejidad de los cálculos. Con el paso del tiempo surgieron alternativas a la probabilidad para tratar la incertidumbre. Con el surgimiento de la Red Bayesiana la probabilidad tomó su verdadero valor. Hoy en día la red bayesiana es la más intuitiva y aceptada de las medidas de incertidumbre. Este tipo de sistema resuelve situaciones de incertidumbre.

Las redes bayesianas son una representación gráfica de dependencias para razonamiento probabilístico, en la cual los nodos representan variables aleatorias y los arcos representan relaciones de dependencia directa entre las variables.

La Figura 3 muestra un ejemplo hipotético de una red bayesiana (RB) que representa cierto conocimiento sobre medicina. En este caso, los nodos representan enfermedades, síntomas y factores que causan algunas enfermedades. La variable a la que apunta un arco es dependiente de la que está en el origen de este, por ejemplo fiebre depende de tifoidea y gripe en la red de la Figura 3. [29]

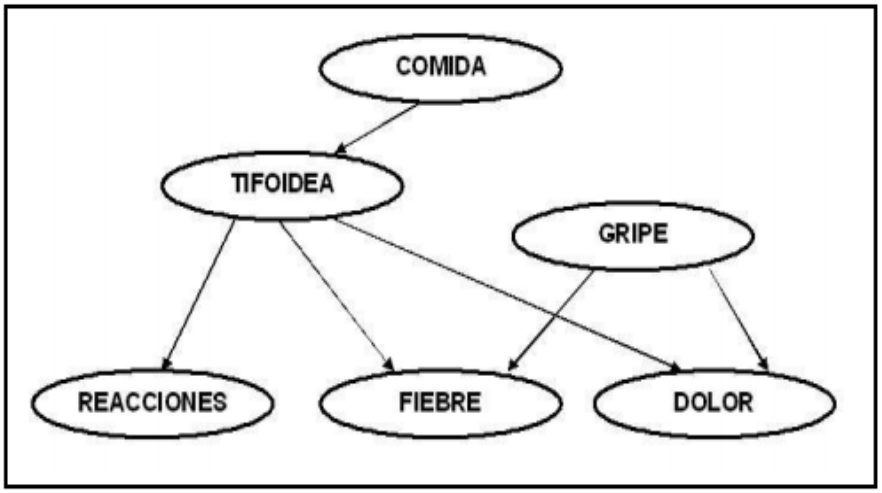

Figura 3: Red Bayesiana

La base de conocimiento de estos sistemas la compone un espacio probabilístico. El motor de inferencia, se basa en la teoría de las probabilidades y específicamente en el Teorema de Bayes .Este método es una herramienta de gran utilidad en la estimación de probabilidades ante nuevas evidencias.

1.6.1.3 Razonamiento Basado en Casos

El Razonamiento Basado en Casos (RBC) representa un nuevo método para resolver problemas no estructurados en el cual el razonamiento se realiza a partir de una memoria asociativa que usa un algoritmo para determinar una medida de semejanza entre dos objetos. En este paradigma la base del comportamiento inteligente de un sistema radica en recordar situaciones similares existentes en el pasado. Debe destacarse que es una técnica en la cual la memoria se sitúa como fundamento de la Inteligencia Artificial y más concretamente de los Sistemas Basados en el Conocimiento. [30]

Razonamiento Basado en Casos significa razonar en base a experiencias o "casos" previos. El RBC es una alternativa entre otras metodologías para construir sistemas basados en el conocimiento. Al razonar basado en casos el solucionador de problemas recuerda situaciones previas similares a la actual y las usa para ayudar a resolver el nuevo problema.

La idea básica del RBC es recuperar, adaptar y validar las soluciones encontradas en experiencia previas en un intento de relacionarlas con un problema actual. Las experiencias previas están representadas como una biblioteca de casos que reside en memoria. Cuando se enfrenta con un nuevo problema, el sistema con RBC recupera un caso similar, y la solución del caso se adapta al nuevo problema en un intento para resolverlo. En la Figura 4 se muestran cada una de las etapas del Razonamiento Basado en Casos.

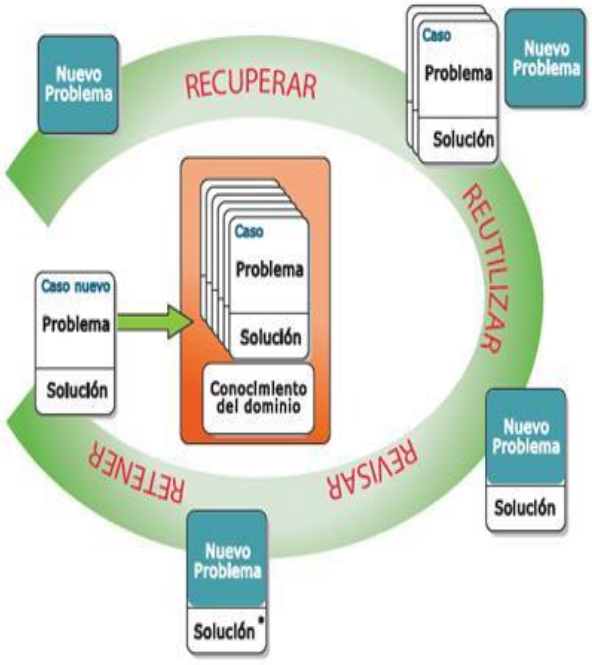

Figura 4: Razonamiento Basado en Casos.

RECUPERAR el caso o casos almacenados más similares.
REUTILIZAR la solución del caso recuperado.
REVISAR la solución propuesta.
RETENER del nuevo caso las experiencias relevantes para el sistema, útiles para la resolución de venideros problemas.

1.7 Selección del SBC

Se desea desarrollar un SBC que permita identificar en cuáles de las actividades de calidad se ha incurrido en costos desfavorables. El objetivo del mismo será apoyar el proceso de toma de decisiones sobre qué medidas o acciones correctivas implementar para disminuir estos costos.

Actualmente se cuenta con las experiencias y casos resultantes de aplicar el procedimiento para el cálculo de los costos de la calidad en el Centro de Telemática. Este conocimiento no es posible representarlo mediante reglas pues el dominio del problema que se presenta no es determinista. Esto quiere decir que un mismo conjunto de indicadores de costos de calidad afectados puede estar asociado a diferentes ejecuciones de los procesos de la gestión de la calidad. Algunos sistemas basados en reglas introducen factores de certeza, de manera que dada una premisa se obtiene una conclusión con un determinado grado de certeza en función de la fuerza de la regla. Estos sistemas no capturan correctamente las dependencias entre variables; cuando se dispara una regla el peso de la conclusión depende únicamente de las premisas, sin tener en cuenta las fuentes de las que provengan. Debido a estas deficiencias los sistemas basados en reglas no son una solución adecuada a problemas no deterministas.

A diferencia de los sistemas basados en reglas, las redes bayesianas sí resultan de gran utilidad en el tratamiento de este tipo de problemas. Sin embargo, por las particularidades y naturaleza del dominio en cuestión dificultarían la implementación del sistema debido a la gran cantidad de combinaciones en las relaciones entre los indicadores de costos de calidad. Ante una situación de incertidumbre la respuesta de una red bayesiana será aquella cuya probabilidad de ocurrencia sea mayor. En cambio, el razonamiento basado en casos recurre a la experiencia acumulada recordando situaciones similares existentes en el pasado, dando como respuesta la solución del caso más semejante. Es precisamente con este tipo de información, específicamente sobre el proceso de gestión de los costos de la calidad, con el que cuenta el Centro de TLM. Por lo anteriormente expuesto, de los tres SBC el RBC es el que mejor permite modelar el conocimiento, experiencia e información con que se cuenta en dicho centro. El mismo permitirá dar solución a los nuevos casos recuperando de la base de conocimiento los más semejantes y reutilizando la solución de los mismos. El sistema podrá aprender del nuevo caso después de haber realizado una evaluación y adaptación del mismo así como un análisis de la factibilidad de almacenarlo en la base de conocimiento.

1.8 Análisis de soluciones existentes

En este siglo XXI que se mueve vertiginosamente, la competitividad es la piedra angular de todos los procesos empresariales que se gestan a diario. Muchas son las empresas, compañías y entidades que prestan similares servicios o venden productos similares, la clave para mantenerse en la preferencia de los clientes radica en una buena gestión de la calidad que a criterio de Bernillion y Cerrutti [31], no es más que un sistema que permita librar los productos conforme a las especificaciones mejorando los costos inútiles de no calidad.

Existen en el mundo potentes herramientas para la gestión de proyectos que abarcan el campo de la gestión de costos. La gestión de proyectos se puede describir como un proceso de planeamiento, ejecución y control de un proyecto, desde su comienzo hasta su conclusión, con el propósito de alcanzar un objetivo final en un plazo de tiempo, costo y nivel de calidad determinado, a través de la movilización de recursos técnicos, financieros y humanos. Incorporando varias áreas del conocimiento, su objetivo final es el de obtener el mejor resultado posible del trinomio coste-plazo-calidad. [32] A continuación se exponen y analizan algunas de estas herramientas:

1.8.1 Microsoft Project

Microsoft Project es un programa o software para la gestión de proyectos. Esta aplicación permite organizar la información acerca de la asignación de tiempos a las tareas, los costos asociados y los recursos, tanto de trabajo como materiales del proyecto para que se puedan respetar los plazos sin exceder el presupuesto y conseguir así los objetivos planteados. [33]
Esta herramienta permite la realización del cálculo de costos de los recursos y la mano de obra una vez que los recursos son asignados a cada tarea. Permite llevar a cabo una planificación estructurada y la realización de cronogramas de desarrollo. Estos elementos inciden en la calidad del proceso de desarrollo de software y por ende en el producto final. Este software posee una licencia privativa y solo puede ser ejecutado sobre Windows; elementos que no acompañan las políticas de la Universidad de las Ciencias Informáticas de migrar a software libre.

1.8.2 OPX2-NPD

Se trata de una herramienta multiplataforma diseñada por la compañía Planisware como un sistema integrado de gestión de proyectos, que cubre, entre otros aspectos: control de costos, gestión de portafolios, gestión de recursos, planificación de tiempos y trabajo colaborativo. En ese sentido, conceptualmente es diferente al Microsoft Project, siendo mucho más poderosa esta aplicación. La misms debe entenderse como una plataforma de trabajo para toda la empresa, más que como un software de usuario. [34]
Esta herramienta hace uso de las buenas prácticas de innovación tecnológica requeridas por el CMMI y la metodología del PMI (Project Managment Institute), la organización más grande asociada a profesionales relacionados con la Gestión de Proyectos en el mundo.El software ofrece la posibilidad de generar gráficos de burbujas, matrices cruzadas, tablas de indicadores, entre otros, como elemento clave para la supervisión del portafolio. En una gráfica de burbujas,

por ejemplo, se resume gran cantidad de información para la toma de decisiones, permitiendo realizar simulaciones para predecir cambios de comportamiento.

1.8.3 Project Objects

Se trata de una plataforma desarrollada por la compañía Project ObjectsLtd con arquitectura modular, basada en web. Cuenta con un módulo que incluye un conjunto de funcionalidades para apoyar el proceso de gestión de costos desde el momento mismo de la financiación hasta los resultados y seguimiento de los mismos.
Utiliza los estándares más reconocidos para control de costos, pero se adapta a cualquier estructura de costos que lleve la empresa, de tal manera que es posible cruzar datos entre proyectos y entre estos y las demás acciones de la empresa. Además facilita la actualización permanente de los datos del proyecto. [34]
Cuenta además con otro módulo llamado ProjectFolio que incluye funciones diseñadas para optimizar el soporte de gestión y del negocio en general. Facilita la toma oportuna de decisiones.

1.8.4 Sistema inteligente para la gestión de la calidad en la producción de habanos. Calidad+IA.

En Cuba se ha desarrollado un sistema computacional llamado Calidad+IA para la fábrica de tabaco torcido de Camajuaní, Unidad Empresarial de Base (UEB) de la Empresa Provincial Tabaco Torcido de Santa Clara dedicada a la fabricación del famoso puro cubano. Este sistema se llevó a cabo con el objetivo de que a partir de la cantidad de los tipos de defectos encontrados en la producción se diagnostique la causa que los provocó lo más rápido y exacto posible, pudiendo reproducir el comportamiento de un experto de la materia. Esta herramienta emplea la técnica de la Inteligencia Artificial: Razonamiento Basado en Casos y fue desarrollada en el lenguaje de programación Java. [35]

Calidad+IA emplea ConFuCiuS [36], un modelo conexionista borroso para desarrollar Sistemas Basados en Casos. El mismo se define como una instancia del clasificador basado en los vecinos más cercanos, pero usando una función de distancia que utiliza los pesos del modelo FSIAC [37]. Este sistema cuenta con una interfaz para mostrar a través de Gráficos de Pareto las cantidades de defectos de los tabacos y las causas de los mismos por categorías. Brinda la posibilidad de generar Gráficos de Control que permiten visualizar el comportamiento del índice

de tabacos defectuosos para una selección determinada. Posibilita además la definición de los valores de los rasgos manualmente y diagnosticar según el modelo escogido.

1.8.5 Herramienta de gestión de costos de calidad para la Empresa Empleadora de Níquel en Moa.

La Empresa Empleadora de Níquel de Moa (EMPLENI) cuenta con un sistema informático para la gestión de los costos de la calidad haciendo uso del enfoque de procesos como medida de desempeño del Sistema de Gestión de la Calidad y de la metodología Hernández Concepción y Moreno Pino.[38]

Este sistema calcula los costos por cada una de las categorías de costos de calidad .En el mismo están definidos los indicadores de costos de calidad para esta empresa.

1.8.6 Calidad de los proyectos de software: Revisiones utilizando Razonamiento Basado en Casos

En el Centro de Estudios de Ingeniería de Sistemas (CEIS) se ha desarrollado un sistema que permite hacer un seguimiento de las inspecciones a los proyectos de la empresa, desde el momento en que se define el plan de aseguramiento de la calidad. El mismo incluye las inspecciones, pasando por la definición de los miembros del equipo de inspectores, las posibles listas de chequeo y terminando con la documentación final de la inspección. Esta documentación es almacenada con el objetivo de que pueda ser consultada y de esta manera comparar los resultados que produce la revisión en la calidad final del producto de software. Además, permitirá llevar métricas asociadas a los defectos, lo que proporciona a la empresa una base de datos con elementos a tener en cuenta en el momento de analizar los procesos involucrados en la producción del software. La aplicación utiliza el RBC a partir del tipo de proyecto para proponer el plan de revisiones, con la lista de chequeo asociada a cada inspección de acuerdo con la experiencia acumulada por la empresa en inspecciones realizadas a proyectos con características similares.

1.8.7 Conclusiones del análisis

El Project Objects es una aplicación web que brinda numerosas facilidades para la gestión de costos, entre las que destacan: la construcción de gráficos para una mejor visualización del comportamiento de los costos y la simulación de procesos como apoyo a la toma de decisiones. Esta herramienta al igual que OPX2-NPD se basa en las mejores prácticas y

estándares existentes para la gestión de proyectos a nivel mundial lo cual permite cruzar información y datos entre proyectos.

La herramienta Microsoft Project realiza el cálculo de los costos una vez que los recursos son asignados a cada tarea. Esta idea pudiera ser seguida en la implementación del sistema al asignar los recursos a cada actividad de calidad, determinando de esta manera los costos de calidad.

Todas las herramientas de gestión de proyectos presentadas fueron concebidas de manera genérica, pudiendo adaptarse para ser empleadas en la gestión de los costos de la calidad del Centro Telemática. Sin embargo, el desarrollo de una herramienta que atienda específicamente a los indicadores de costos de calidad del centro, resultaría una mejor solución pues permitiría el cálculo de los indicadores de acuerdo a lo establecido en el procedimiento con el que cuenta el centro para este fin. Por otra parte, la gestión que llevan a cabo estas herramientas no abarca aspectos como la identificación de costos desfavorables ni la proposición de acciones correctivas para erradicarlos o disminuirlos. En aditamento a lo señalado todas estas herramientas son propietarias, la Universidad de las Ciencias Informáticas tendría que invertir una considerable suma de dinero para adquirir cada una de las licencias que exigen las empresas comercializadoras de estos productos de software.

Calidad+IA y la herramienta de gestión de costos de calidad para la Empresa Empleadora de Níquel de Moa fueron desarrolladas con el objetivo de satisfacer las necesidades y requerimientos de los procesos productivos de la UEB de tabaco torcido de Camajuaní y de la Empresa Empleadora de Níquel de Moa respectivamente. Es por esto que no pueden ser empleadas de manera íntegra para la disminución de los costos de la calidad del centro de TLM pero algunos de sus componentes resultan relevantes en esta investigación. En la implementación del sistema que se propone en este trabajo se empleará en la etapa de recuperación de los casos de la base de conocimiento como medida de similitud, la distancia Euclídea ponderada a semejanza del método que emplea la herramienta para la fábrica de tabaco torcido de Camajuaní. Dicho sistema permitirá al igual que Calidad+IA la asignación manual del peso a cada rasgo. Además posibilitará el cálculo para cada uno de los indicadores de costos de calidad del Centro de TLM al igual que el sistema desarrollado para EMPLENI lo hace para los indicadores de costos de calidad definidos en esta entidad.

La herramienta que propone las revisiones a los proyectos de software empleando el Razonamiento Basado en Casos permite controlar y dar seguimiento al plan de revisiones de cada proyecto, así como hacer un análisis de la calidad a partir de las métricas empleadas.

Todo esto se lleva a cabo a partir de la experiencia acumulada por la empresa en inspecciones realizadas a proyectos con características similares. Este sistema tiene en cuenta la visión estratégica de la calidad recurriendo más que a una inspección final, a los procesos de gestión de la calidad, pudiendo planificar una inspección en cualquier etapa de desarrollo del proyecto. De manera general este sistema se enfoca en la proposición de listas de chequeo para el control de la calidad de los proyectos de software y no en la factibilidad de los costos de este tipo de actividades.

1.9 Herramientas de desarrollo

1.9.1 Lenguaje de programación. Groovy.

Es un lenguaje orientado a objetos, nacido con la misión de acoplarse de forma efectiva con Java, trabajar en conjunto con la Máquina Virtual de Java JVM y soportar los tipos de datos estándar, pero añadiendo características dinámicas y sintácticas presentes en otros lenguajes como Python, Smalltalk o Ruby.[39]

Desde Groovy se puede acceder directamente a todas las API existentes en Java. Lo que Groovy aporta es una sintaxis que aumenta la productividad y un entorno de ejecución que permite manejar los objetos de formas que en Java serían extremadamente complicados. Todo lo anterior unido a que la mayor parte de código escrito en Java es totalmente válido en Groovy hace que este lenguaje sea de muy fácil adopción para programadores de Java.

1.9.2 Marcos de trabajo

1.9.2.1 Grails

Grails es un marco de trabajo libre, del inglés framework, para aplicaciones web desarrollado sobre el lenguaje de programación Groovy (el cual a su vez se basa en la plataforma Java). Grails pretende ser un marco de trabajo altamente productivo siguiendo paradigmas tales como Convención sobre configuración (en inglés Convention over Configuration) y No te repitas (en inglés Don’t repeat yourself o DRY) , proporcionando un entorno de desarrollo estandarizado y ocultando gran parte de los detalles de configuración al programador.

Grails se ha desarrollado con los siguientes objetivos:

- ✓ Ofrecer un framework web de alta productividad para la plataforma Java.
- ✓ Reutilizar tecnologías Java ya probadas como Hibernate y Spring bajo un interfaz simple y consistente.

- ✓ Ofrecer un framework que reduzca la confusión y que sea fácil de aprender.

Grails tiene tres características que intentan incrementar su productividad comparándolo con los framework Java tradicionales:

- ✓ **Inexistencia de configuración XML:** Grails elimina la necesidad de configurar ficheros XML. En su lugar el framework utiliza una serie de reglas de convención mientras examina el código de las aplicaciones basadas en Grails. Por ejemplo, una clase que termina en Controller es considerada un controlador web.
- ✓ **Entorno de desarrollo preparado para funcionar desde el primer momento:** Mientras que para el uso de herramientas Java tradicionales, es tarea del desarrollador ensamblar los componentes, Grails tiene un servidor web integrado preparado para desplegar la aplicación desde el primer momento. Todas las librerías requeridas son parte de la distribución de Grails y están preparadas para ser desplegadas automáticamente.
- ✓ **Funcionalidad disponible mediante métodos dinámicos:** Grails proporciona métodos dinámicos en varias de sus clases. Un método dinámico se añade a la clase en tiempo de ejecución, como si su funcionalidad hubiera sido compilada. Estos métodos dinámicos permiten a los desarrolladores realizar operaciones sin tener que implementar interfaces o heredar clases base. Grails proporciona métodos dinámicos basándose en el tipo de clase. Por ejemplo, las clases de dominio tienen métodos para automatizar operaciones de persistencia, como save para salvar, delete para borrar y find para buscar.

Grails al igual que la mayoría de los frameworks web se basa en la arquitectura MVC (Modelo-Vista-Controlador). El hecho más significativo dentro de la arquitectura de Grails es la integración trasparente de los frameworks Java de código abierto más utilizados, como lo son: Spring, Hibernate, SiteMesh y JUnit. Grails se ha diseñado para el desarrollo ágil de aplicaciones web ofreciendo un balance adecuado entre consistencia y funcionalidades potentes.

1.9.2.2 Bootstrap

Bootstrap es un potente framework que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. Ha sido desarrollado por Twitter. La mayor ventaja es que facilita la creación de interfaces que se adaptan a los distintos navegadores. Se integra

perfectamente con los principales frameworks Javascript como JQuery. Ofrece un diseño sólido usando estándares como CSS3/HTML5. [40]

1.9.3 Entorno Integrado de Desarrollo. IntelliJ IDEA.

IntelliJ IDEA es un entorno de desarrollo Java creado por JetBrains del que existen dos distribuciones: Comunity Edition (código abierto) y Ultimate (comercial).
En este trabajo se empleará la distribución Comunity Edition porque está exenta de pago. La misma provee soporte para el lenguaje de programación seleccionado, Groovy. Consta de un editor de código inteligente que realiza refactorizaciones, inspecciones de código, identifica acciones intencionales y brinda una fácil navegación. Integra los frameworks de prueba JUnit y TestNG además de una interfaz de usuario para ejecutar pruebas. [41]

1.9.4 Lenguaje de Modelado. UML.

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad puesto que se utiliza para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

Otra característica de este modelado visual es que es independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (lenguajes orientados a objetos).

1.9.5 Herramienta CASE. Visual Paradigm.

La herramienta CASE seleccionada para el desarrollo del sistema es Visual Paradigm for UML 8.0. La misma soporta la metodología de desarrollo de software seleccionada, pudiéndose por consiguiente, generar todos los diagramas y esquemas que define la misma en notación UML. Es multiplataforma y cubre todas las fases del ciclo de vida de desarrollo del software. Es una herramienta CASE de diseño orientada a facilitar el desarrollo de software. Ofrece una colección de herramientas de desarrollo de software para la captura de requisitos, modelamiento de clases, modelado de datos, entre otras funcionalidades.

Posee características gráficas amigables que facilitan la realización de diagramas de modelado basado en el estándar UML tales como los diagramas de clases, casos de uso, comunicación, secuencia, estado, actividades, componentes, entre otros.

1.9.6 Sistema Gestor de Bases de Datos. PostgreSQL.

PostgreSQL es un Sistema Gestor de Base de Datos objeto-relacional de código abierto. Posee excelente ajuste al estándar SQL. Es distribuido bajo la Licencia PostgreSQL, que permite a los usuarios disponer del código (puede ser usado, modificado y distribuido por cualquiera libremente), incluso revender el producto compilado sin el código fuente. La única restricción es la imposibilidad de responsabilizar legalmente a los desarrolladores por problemas con el software. PostgreSQL funciona en cualquier plataforma compatible con UNIX además de ejecutarse en los sistemas operativos Windows.

El proyecto de PostgreSQL vuelca sus esfuerzos fundamentalmente en la robustez y la calidad así como en el alto rendimiento, compatibilidad y seguridad. PostgreSQL destaca por el amplio espectro de características avanzadas que posee.

Es necesario, en el desarrollo del sistema que se propone el almacenamiento de un gran volumen de casos, por lo que se requiere de un SGBD robusto, de alto rendimiento, flexibilidad y seguridad. PostgreSQL satisface todos estos requerimientos.

1.10 Metodología de desarrollo. RUP.

RUP es un proceso de desarrollo de software y como tal define un conjunto de actividades necesarias para transformar los requisitos del usuario en un software. Sin embargo, no solo contempla los aspectos relacionados con el proceso de desarrollo sino que constituye un marco de trabajo genérico que puede especializarse para ser adaptado a sistemas de software, diferentes áreas de aplicación, diferentes tipos de organizaciones y diferentes tamaños de proyecto. Es un proceso que define “quién” está haciendo “qué”, “cuándo” y “cómo” para alcanzar un determinado objetivo. [42] En la Figura 5 se muestran las 4 fases de desarrollo y los 9 flujos de trabajo que conforman esta metodología de desarrollo de software.

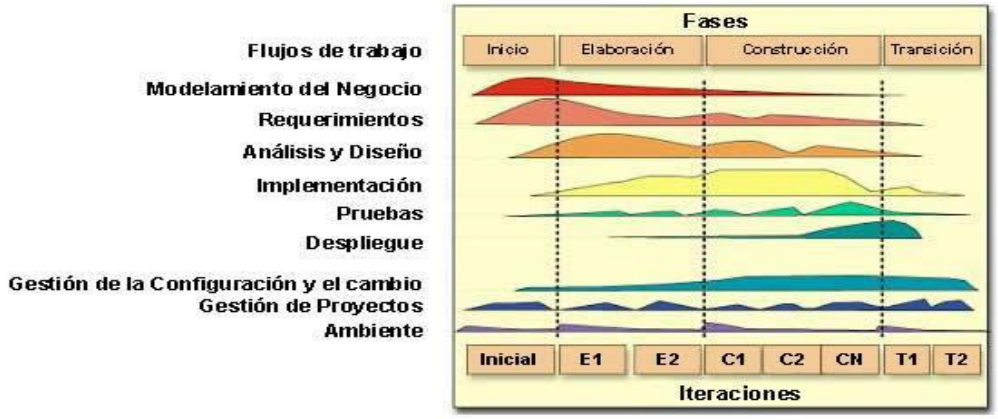

Figura 5: Fases y flujos de trabajo de RUP.

RUP posee tres características fundamentales:

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

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

- ✓ Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Por ejemplo, una iteración de elaboración centra su atención en el análisis y diseño, aunque refina los requerimientos y obtiene un producto con un determinado nivel, pero que irá creciendo incrementalmente en cada iteración. Es práctico dividir el trabajo en partes más pequeñas. Cada parte es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

Se decidió usar el Proceso Unificado de Desarrollo por ser una metodología basada en las mejores prácticas que se han intentado y probado en el campo de la Ingeniería de Software. Además, su empleo genera una gran cantidad de documentación la cual resultará de gran utilidad para el mantenimiento del sistema y el desarrollo de próximas versiones del mismo. En aditamento a lo planteado, es en esta metodología en la que el desarrollador del sistema cuenta con más experiencia.

Conclusiones del capítulo

En el presente capítulo se analizaron las herramientas con fines similares al de la investigación, concluyendo que para satisfacer las necesidades del Centro de TLM en el campo de la gestión de los costos de la calidad se requiere el desarrollo de una nueva herramienta inteligente que hago uso del conocimiento de expertos en el área. Se expusieron los conceptos fundamentales sobre los procesos de gestión de la calidad así como un análisis de los diferentes Sistemas Basados en el Conocimiento, seleccionando para la implementación de la aplicación el Sistema Basado en Casos. Se determinó usar RUP como metodología para guiar el proceso de desarrollo de software. El lenguaje de programación para la implementación del lado del servidor será Groovy 1.8.6 y los marcos de trabajo que se emplearán son: Grails en su versión 2.0.1 basado en Groovy y Bootstrap en su versión 2.3. Se usará la herramienta CASE Visual Paradigm para UML versión 8.0 en su Edición Empresarial y como lenguaje de modelado UML. El Entorno Integrado de desarrollo que se empleará es IntelliJ IDEA y como Sistema Gestor de Bases de Datos PostgreSQL en su versión 9.2.

Capítulo 2. Características del sistema.

Introducción

En el presente capítulo se detalla la propuesta del sistema. Se presenta el diagrama de casos de uso del sistema así como la descripción de los mismos. Además se enumeran los requisitos funcionales y no funcionales del sistema.

2.1 Objeto de estudio

2.1.1 Problema y situación problemática

El Centro de Telemática de la Universidad de las Ciencias Informáticas es una entidad destinada al desarrollo de servicios y aplicaciones en el campo de las Telecomunicaciones y la Seguridad Informática. Dicho centro cuenta con un procedimiento para el cálculo de los costos de la calidad que permite tener una visión general sobre el desempeño de los proyectos en cuanto a la Gestión de la Calidad. El mismo se realiza de forma manual lo que conlleva a grandes pérdidas de tiempo al realizar los cálculos y da margen a que se cometan errores en su aplicación. Este carece de una vía para identificar en cuáles de las actividades de calidad se ha incurrido en costos desfavorables a lo largo del ciclo de vida del software, lo cual imposibilita llevar a cabo acciones correctivas para disminuirlos o evitarlos.

En entrevistas realizadas a líderes y especialistas del centro se evidenció la falta de claridad por parte del equipo de desarrollo para determinar los indicadores de costos de calidad que resultan más perjudiciales, de ahí que no se puedan ejecutar acciones correctivas pertinentes para su mitigación. Esta situación puede ocasionar atrasos en los cronogramas de los proyectos, el incumplimiento de los requerimientos pactados y la entrega de un producto al cliente sin la calidad necesaria. Al mismo tiempo, la experiencia adquirida por los especialistas sobre los costos de la calidad no es documentada, almacenada o estructurada por lo que con la pérdida del personal se produce a su vez la pérdida del conocimiento. Debido a estos elementos, se puede manifestar la disminución de los índices de satisfacción de los clientes lo que atentaría contra la credibilidad de la institución ante la sociedad. Otra consecuencia se podría observar en el factor económico, pues además de disminuir los ingresos a partir de la pérdida de clientes se generarían gastos adicionales por la incorrecta planificación de las actividades que se llevan a cabo en el proceso de desarrollo de software.

2.1.2 Objeto de automatización

Actualmente el Centro de Telemática no cuenta con una herramienta informática que permita, ayude o agilice el proceso de gestión de costos de calidad en el ciclo de vida de los proyectos de desarrollo de software. Este trabajo tiene como objetivo proveer a los proyectos del centro de una herramienta inteligente que contribuya a la disminución de los costos de calidad.

El cálculo de los costos de la calidad en el Centro de Telemática se realiza de forma manual mediante el procedimiento con que cuenta para este fin. Este proceso es realizado por los especialistas de calidad de cada uno de los proyectos.

El Sistema Basado en Casos que se propone en el presente trabajo, permitirá identificar los indicadores de costos de calidad afectados en el proceso de gestión de la calidad del Centro Telemática. Se dispone además, de posibles acciones correctivas a ejecutar para evitarlos o disminuirlos.

2.1.3 Información que se maneja

La información que se maneja está relacionada con los costos de la calidad de los proyectos del Centro de Telemática así como las medidas y acciones correctivas para evitar o disminuir los mismos.

2.2 Propuesta de sistema

El Sistema Basado en Casos que se propone en este trabajo se basa en la información que introduce el usuario sobre los costos de la calidad de un proyecto determinado del Centro de Telemática. El sistema calcula los costos de calidad por cada una de las categorías de costos de la calidad: prevención, evaluación, fallas internas y fallas externas determinando si los costos de calidad y los de mala calidad son adecuados o no. En caso de resultar no adecuados el sistema recupera de la base de conocimiento los casos más semejantes a partir de un umbral. Luego se reutilizan estos casos con el objetivo de darle solución al nuevo problema. A partir de la identificación de las actividades de calidad con costos desfavorables se proponen medidas o acciones correctivas para disminuir o evitar estos costos. Estas medidas fueron tomadas de la investigación titulada “Propuesta de acciones para disminuir los costos de calidad en los proyectos del Centro de Telemática y diseño de una aplicación que informatice el cálculo de los costos de calidad”. [43] Las mismas se encuentran en el **Anexo 2**. Finalmente se evalúa el almacenamiento del caso o no en la base de conocimiento. Si es posible obtener

información relevante sobre este nuevo caso por sus características, el sistema aprende del mismo sino es desechado.

2.3 Modelo de dominio

El objetivo del modelo de dominio es contribuir a la comprensión del contexto del sistema, y por lo tanto favorecer al entendimiento de los requisitos del sistema que se desprenden de este entorno. [44] En la Figura 6 se muestra el modelo de dominio confeccionado.

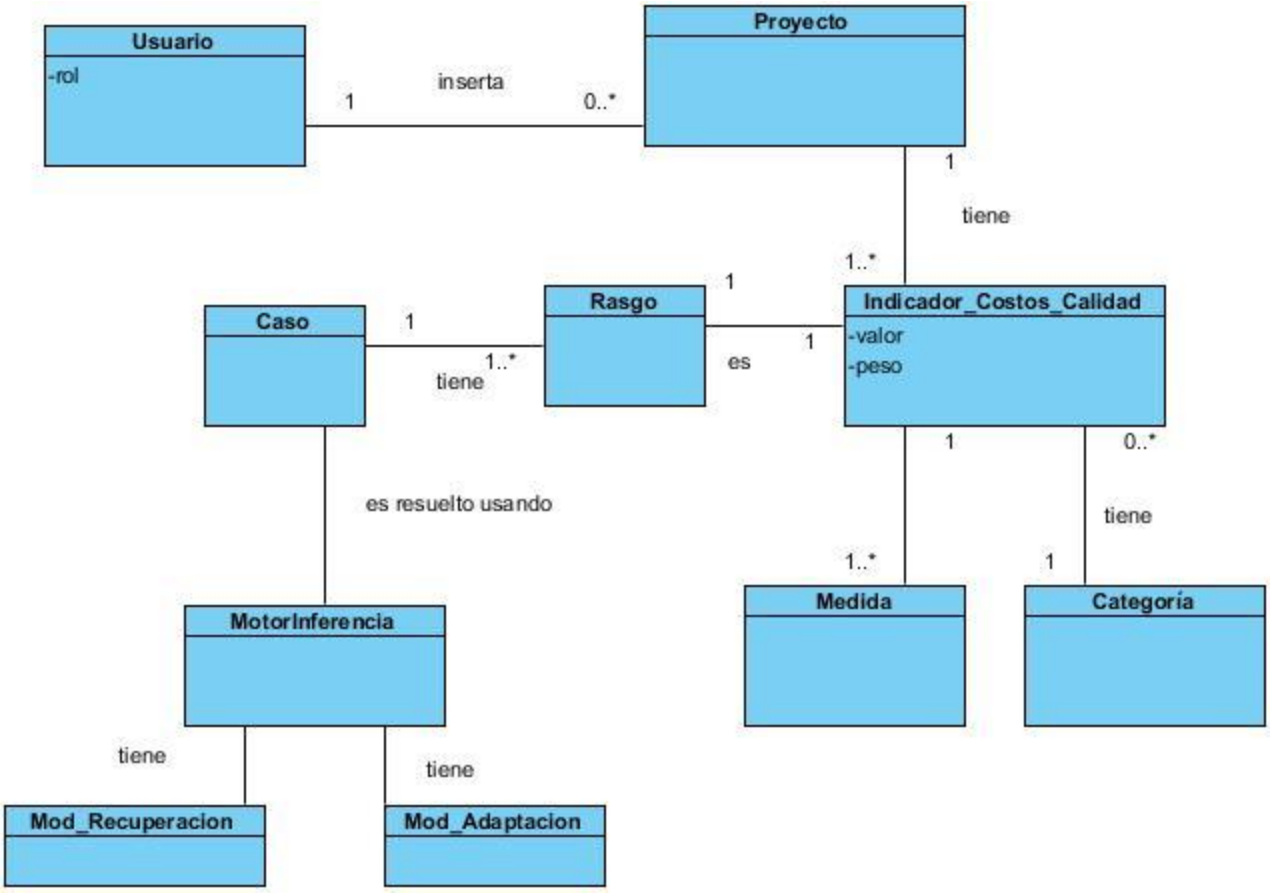

Figura 6: Modelo de dominio.

Usuario: define a los actores del sistema: administrador de la calidad general y administrador de calidad del proyecto.

Proyecto: define los proyectos del Centro Telemática.

Caso: contiene un problema y su solución. El problema está compuesto por un conjunto de valores de indicadores de costos de calidad y la solución, por el subconjunto de indicadores afectados.

Rasgo: es un indicador de costos de la calidad.

Categoría: agrupa a los indicadores de costos de calidad. Las posibles categorías son: Prevención, Evaluación, Fallas Internas, Fallas Externas.

Medida: acción correctiva para disminuir un indicador de costo de calidad en caso que resulte desfavorable.

Indicador_Costos_Calidad: define a cada uno de los indicadores de costos de calidad del Centro de TLM.

MotorInferencia: programa que controla el razonamiento y dirige la búsqueda en la base de casos.

Mod_Recuperación: módulo del sistema para la recuperación de los casos de la base de casos.

Mod_Adaptación: módulo del sistema para la adaptación de los casos.

2.4 Relación de los Requerimientos

2.4.1 Requerimientos funcionales.

RF1. Autenticar usuario.

- Nombre de usuario.
- Contraseña.

RF2.Insertar Usuario.

- Nombre.
- Nombre de usuario
- Rol.
- Nombre proyecto.
- Contraseña.

RF3. Modificar usuario.

Datos a modificar (Remitirse a **RF2**).

RF4. Eliminar usuario.

Datos (Remitirse a **RF2**)

RF5. Listar usuarios.

Datos a mostrar (Remitirse a **RF2**)

RF6. Insertar Proyecto.

- Nombre

RF7. Modificar Proyecto.

- Modificar cualquiera de los datos insertados (Remitirse a **RF6**).

RF8. Listar Proyectos.

Datos a mostrar (Remitirse a **RF6**)

RF9.Eliminar Proyecto

Datos (Remitirse a **RF6**)

RF 10. Calcular el tiempo total por revisiones y documentación.

- Tiempo en revisiones y documentación por persona.

RF 11. Calcular el tiempo total por planeación de la calidad.

- Horas dedicadas por persona para planificar la calidad.

RF12. Calcular el tiempo total por mejoramiento de la calidad.

- Tiempo dedicado por persona al mejoramiento de la calidad.

RF 13. Calcular el tiempo de adiestramiento y capacitación del personal del proyecto.

- Personal matriculado en curso.
- Cantidad de horas del curso.
- Tiempo en elaboración de un plan de proyecto.

RF 14. Calcular el tiempo total por planificación de medidas y programas de seguridad.

- Tiempo en planificar las medidas por persona.

RF 15. Calcular el tiempo total por mantenimiento preventivo.

- Tiempo en reparaciones por equipo.

RF 16. Calcular el tiempo total por gestión de riesgos.

- Tiempo invertido en actividades para la gestión de riesgos.

RF 17. Calcular el tiempo total de auditorías de calidad.

- Tiempo por cada auditoría de calidad.

RF 18. Calcular el tiempo total en revisiones de documentación.

- Tiempo por cada revisión de la documentación.

RF 19. Calcular el tiempo total de pruebas al producto.

- Tiempo en planificación de prueba.
- Tiempo en aplicación de prueba.
- Tiempo en verificar resultado de la prueba.

RF 20. Calcular el tiempo total en verificar el nivel de cada empleado.

- Tiempo en realizar pruebas a cada empleado.

RF 21. Calcular el tiempo total de inspección del producto.

- Tiempo en realizar cada inspección general.

RF 22. Calcular el tiempo total de refinamientos.

- Tiempo por cada refinamiento.

RF 23. Calcular el tiempo total en reparaciones de máquinas.

- Tiempo en reparaciones de maquinarias.

RF 24. Calcular el tiempo total por insuficiente capacitación.

- Tiempo de duración de cada curso de capacitación.
- Matrícula de cada curso.

RF 25. Calcular el tiempo total por descuido de seguridad.

- Tiempo en garantizar la seguridad.

RF 26. Calcular el tiempo total por reproceso.

- Tiempo en eliminar cada reproceso.

RF 27. Calcular el tiempo total en análisis de ausencias.

- Tiempo en corregir cada ausencia.

RF 28. Calcular el tiempo total invertido en corregir errores de diseño y procesamiento de datos.

- Tiempo en corregir errores de diseño.
- Tiempo en corregir errores de impresión.

RF 29. Calcular el tiempo total por cambio de ingeniería.

- Tiempo por cada cambio de ingeniería.

RF 30. Calcular el tiempo total por reclamaciones del cliente.

- Tiempo en atender cada reclamación del cliente.

RF 31. Calcular el tiempo total en materiales devueltos.

- Tiempo en corregir cada material devuelto.

RF 32. Calcular el tiempo total por descuentos.

- Tiempo en realizar cada descuento.

RF 33. Calcular el tiempo total de demandas por incumplimientos.

- Tiempo en atender demandas por incumplimiento.

RF 34. Calcular el tiempo total en garantías.

- Tiempo por cada garantía.

RF 35. Calcular el tiempo total por entrevistas.

- Tiempo por cada entrevista.

RF 36. Calcular el tiempo total por levantamientos de requisitos.

- Tiempo de levantamiento de requisitos.

RF 37. Calcular el tiempo total por implementación.

- Tiempo por cada actividad de implementación.

RF 38. Insertar indicador

- Nombre indicador
- Categoría a la que pertenece
- Peso

RF 39. Modificar indicador

Modificar cualquiera de los datos insertados (Remitirse a RF38)

RF 40. Eliminar indicador

- Nombre indicador
- Categoría a la que pertenece
- Peso

RF 41. Mostrar indicador

Datos a mostrar (Remitirse a **RF38**)

RF 42. Insertar medida

- Nombre indicador
- Medida

RF 43. Eliminar medida

- Nombre indicador
- Medida

RF 44. Modificar medida

- Modificar cualquiera de los datos (**RF42**).

RF 45. Listar medidas

Datos a mostrar (Remitirse a **RF42**).

RF 46. Identificar indicadores de calidad afectados.

Si el estado de los costos fue no adecuado determinar cuáles resultaron afectados en los procesos de gestión de la calidad.

- Nombre parámetro
- Valor parámetro
- Descripción indicador

- Nombre indicador

RF 47. Mostrar resultados

- Nombre de los indicadores afectados
- Medidas para disminuir cada indicador afectado

RF 48. Aprender caso.

- Indicadores.
- Indicadores afectados

RF 49. Determinar estado de los indicadores de costos de calidad.

Insertar los indicadores de costos de calidad y determinar si son adecuados o no.

RF 50. Adicionar parámetro

- Nombre
- Descripción
- Nombre Indicador

RF 51. Eliminar parámetro

Datos (Remitirse a **RF 49**)

RF 52. Modificar parámetro

Datos a modificar (Remitirse a **RF 49**).

RF 53. Mostrar parámetros

Datos a mostrar (Remitirse a **RF 49**).

RF 54. Adicionar categoría

- Nombre

RF 55. Eliminar categoría

Datos (Remitirse a **RF 53**)

RF 56. Modificar categoría

Datos a modificar (Remitirse a **RF 53**).

RF 57. Mostrar categoría

Datos a mostrar (Remitirse a **RF 53**).

2.4.2 Requerimientos no funcionales

Apariencia o interfaz externa.

- La interfaz debe ser intuitiva y agradable para el usuario. Contará con un menú lateral izquierdo donde se mostrarán las funcionalidades del sistema a las que tiene acceso el usuario autenticado. Todas las letras de la aplicación serán de color negro excepto las

del menú lateral que serán azules. Se empleará la fuente de letra Helvetica Neue con tamaño 18 px para los títulos de las páginas y 14 px para el resto. El color de fondo que se utilizará será el gris.

- Los mensajes de error se mostrarán en color rojo e indicarán al usuario las posibles causas de estos. Los mismos aparecerán en la parte superior del sistema excepto en la página de la autenticación que aparecerán en el esquina inferior derecha.
- Los mensajes de información sobre las transacciones en el sistema se mostrarán en color azul.
- Los botones cuya funcionalidad es la eliminación de un elemento determinado aparecerán en color rojo.

Usabilidad.

- El sistema podrá ser usado por cualquier persona que posea conocimientos en el área de la calidad.

Seguridad

- El sistema debe tener la capacidad de verificar la identidad de los usuarios. El proceso de autenticación se llevará a cabo mediante el servidor LDAP de la UCI.
- El sistema establecerá roles para asegurar el acceso por cada usuario a la información autorizada.
- La comunicación entre el cliente y el servidor web se realizará utilizando el protocolo HTTPS.
- El sistema debe estar disponible al menos durante la jornada laboral, de lunes a sábado de 8:00 am a 5:00 pm.

Legales

- La Universidad de las Ciencias Informáticas poseerá la propiedad intelectual sobre el producto final.

Ayuda y documentación en línea.

- El sistema deberá contar con un manual en línea que apoye al usuario durante su trabajo en el mismo.

Software

- La PC donde se ejecute la aplicación debe tener instalado Firefox 3.X o superior, Chrome 5.X o superior o Safari, debido a que los estándares de CSS3 no son reconocidos por muchas de las versiones de Internet Explorer (6.X, 7.X). Debe tener alojado el SGBD PostgreSQL 9.2 o contar con dicho servidor externo.

- En caso de ser externo el servidor de base de datos, debe tener instalado PostgreSQL en su versión 9.2.
- El servidor de aplicaciones debe tener instalado el Apache Tomcat en la versión 6.0 o superior y la Máquina Virtual de Java.

Hardware

- Se requiere una computadora con un microprocesador cuya velocidad sea superior a los 1.6 GHz, y con una memoria RAM superior a los 256 MB, si en ella no se encuentra el servidor de Base de Datos.
- Si el servidor de base de datos es externo, debe estar alojado en una computadora con al menos 1 GB de RAM y con una capacidad para el almacenamiento de información 160 GB.

Restricciones del diseño

- El sistema que se desarrollará será una aplicación web.
- El sistema se implementará usando la plataforma Java.
- El patrón arquitectónico que empleará el sistema será MVC.

2.5 Modelo de casos de uso del sistema

2.5.1 Definición de los actores del sistema a automatizar

Actores	Justificación
Administrador de calidad	Es el encargado de introducir los parámetros que requiere el sistema para la identificación de los indicadores de costos de calidad afectados para el proyecto al que pertenece y de gestionar las medidas para disminuir los mismos. Realiza la autenticación en el sistema. Además es el responsable de la gestión de categorías, indicadores y parámetros de los mismos.

Administrador de calidad general	Es un actor del sistema especializado de Administrador de calidad. Tiene acceso a las funcionalidades del administrador de calidad del proyecto con la especificidad de que el proceso de identificación de los indicadores de costos de calidad afectados puede llevarlo a cabo para todos los proyectos del centro. Además es responsable por la gestión de los proyectos y los usuarios del sistema.

Tabla 1: Descripción de los actores del sistema.

2.5.2 Diagrama de casos de uso

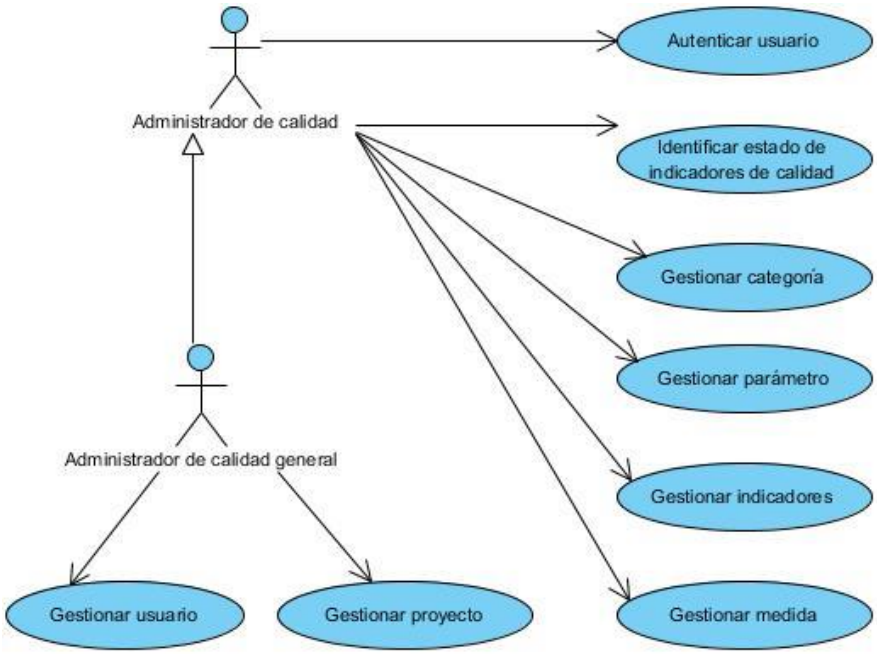

Figura 7: Diagrama de casos de uso del sistema

Conclusiones del capítulo

En el presente capítulo se realizó el modelamiento del dominio para una mejor comprensión del sistema a desarrollar. Además se expuso la situación problemática, el objeto de automatización y la información que se maneja en este proceso. Se definieron los requisitos funcionales y no funcionales del sistema. Para darle cumplimiento a los mismos se llevó a cabo el modelamiento y descripción de los casos de uso.

Capítulo 3. Diseño, Implementación y Pruebas.

Introducción

En el presente capítulo se describen los patrones empleados en la implementación del sistema. Se llevan a cabo los diagramas del diseño web y los de interacción, específicamente los de secuencia para cada uno de los casos de uso del sistema. También es presentado el modelo lógico de datos así como una descripción de las clases fundamentales del sistema. Además se exponen el diagrama de componentes del sistema y el diagrama de despliegue, artefactos generados durante el flujo de trabajo de Implementación de la metodología de desarrollo de software RUP. Por otro lado se presentan las técnicas empleadas como parte de la implementación del RBC. Finalmente se lleva a cabo la realización de pruebas al sistema.

3.1 Patrones de diseño

Según el arquitecto Alexander Cristopher un patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma.[45]

Un patrón de diseño nomina, abstrae e identifica los aspectos claves de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se centra en un problema concreto, describiendo cuándo aplicarlo y si tiene sentido hacerlo teniendo otras restricciones de diseño, así como las consecuencias y las ventajas e inconvenientes de su uso. [46]

El empleo de patrones de diseño en el desarrollo de software ofrece numerosas ventajas, entre ellas:

- Reducción de tiempos.
- Disminución del esfuerzo de mantenimiento.
- Aseguramiento de la consistencia.
- Aumento de la fiabilidad.
- Reutilización de código.

3.1.1 Inversión del control

La Inversión del Control es un patrón de diseño pensado para permitir un menor acoplamiento entre los componentes de una aplicación y fomentar de esta manera la reutilización. Según este patrón las dependencias de un componente no deben gestionarse desde él mismo para que solo contenga la lógica necesaria para realizar su trabajo. Este patrón es empleado por Grails, framework seleccionado para el desarrollo del sistema. Dicho marco de trabajo es el encargado de configurar Spring para que gestione el ciclo de vida de los componentes que se creen en la aplicación. Cuando en un controlador se requiere emplear las funcionalidades que brinda un servicio determinado solo es necesario definir una variable con el nombre del mismo. El controlador no es responsable por la instanciación o configuración del servicio, en su lugar Grails utiliza el contenedor de Spring para estas tareas. [47] Con el empleo de este patrón los componentes del sistema basado en casos que se propone en este trabajo serán más sencillos. De esta manera la aplicación será más fácil de entender y mantener.

3.1.2 Patrones GOF

Singleton:

Este patrón garantiza que una clase tenga una única instancia, proporcionando un punto de acceso global a la misma. Grails controla el ciclo de vida de los servicios, decidiendo cuándo se crean instancias de los mismos. Por defecto, todos los servicios son singleton: solo existe una instancia de la clase que se inyecta en todos los artefactos que declaren la variable correspondiente. [48]

Decorator:

Tiene como propósito añadir responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. Grails integra el framework SiteMesh para la composición de las vistas. El mismo se basa en el patrón Decorator, partiendo de un layout inicial de la página e incorporando modificaciones a partir de cada vista particular. Cuando se realiza una petición HTTP, el controlador procesa la lógica de negocio necesaria y decide la vista o página GSP (tecnología empleada por Grails para el trabajo con las vistas) correspondiente a mostrar al usuario. Esta tecnología es la responsable por el procesamiento de la página y la generación del documento HTML, el cual es utilizado por el framework SiteMesh para decorar o especializar la vista en cuestión. En la Figura 8 se muestra el flujo de ejecución del sistema al procesar una petición HTTP anteriormente descrito.

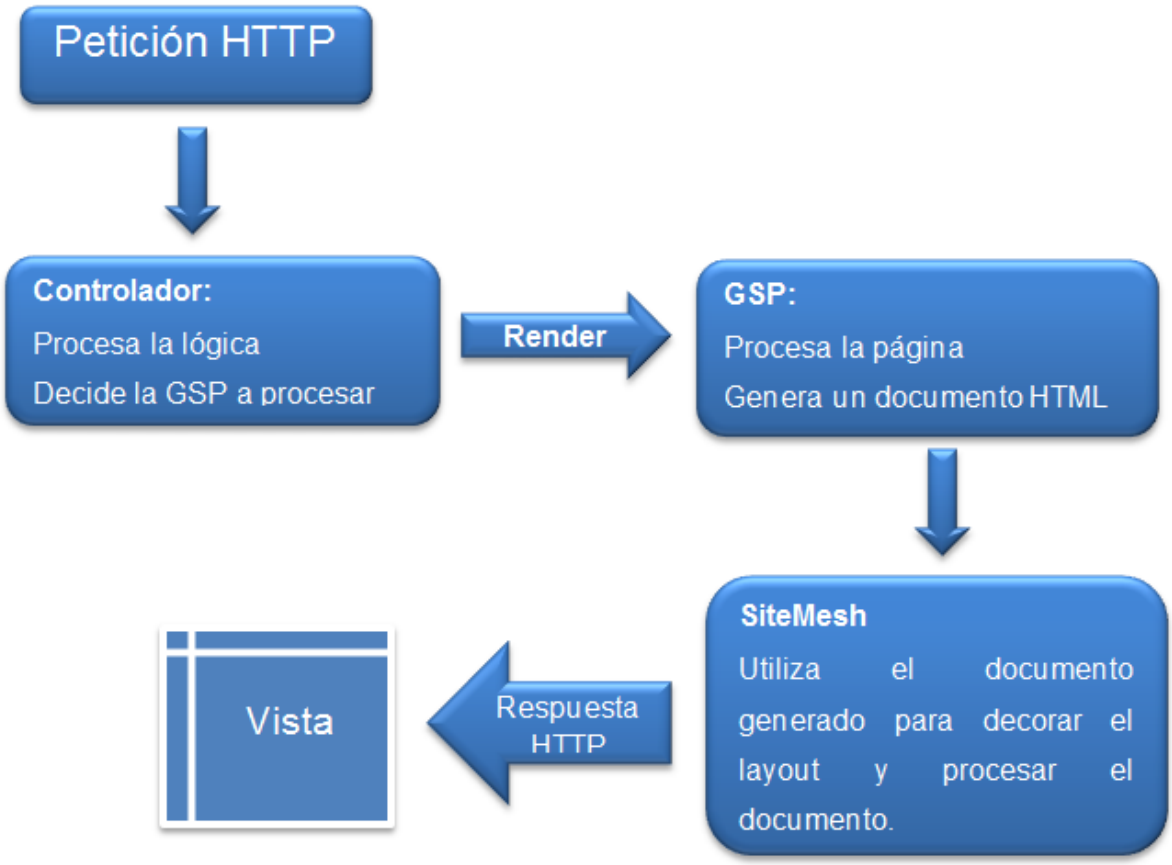

Figura 8: Flujo de ejecución del sistema al procesar una petición HTTP.

3.2 Patrón arquitectónico. MVC.

Este patrón arquitectónico permite la separación del software en capas, facilitando así la programación de manera paralela e independiente. MVC propone tres capas:

Modelo: Es la representación de la información que maneja la aplicación.

Vista: Es la representación del modelo en forma gráfica disponible para el usuario.

Controlador: Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información y modificando el modelo y/o las vistas en caso de ser necesario.

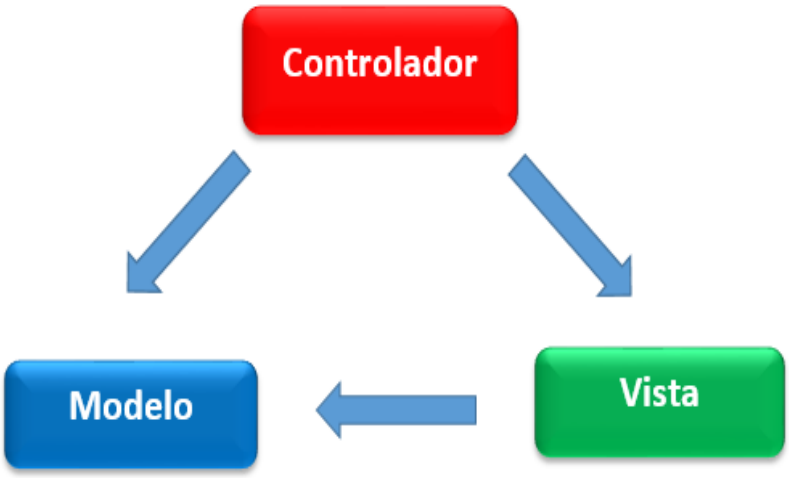

Figura 9: Modelo Vista Controlador.

Grails se basa como la mayoría de los frameworks web en este patrón arquitectónico Modelo-Vista- Controlador. Dicho framework lo implementa siguiendo el paradigma Convention over Configuration. Grails define cuatro paquetes para este fin: Domain donde se posiciona el modelo de datos de la aplicación, Views destinado a las vistas, Controllers para el almacenamiento de las clases controladoras y por último el paquete Services donde se implementa la lógica de negocio, separándola de los controladores. De manera que estos últimos, solo son garantes del correcto flujo del sistema. El contenido de las carpetas Domain responde a la capa Modelo, Views a la capa Vista mientras que Controllers lo hace a la capa Controlador. Los nombres de las clases en cada una de estos paquetes con excepción de la carpeta Domain tienen un formato predeterminado debido a la aplicación del paradigma Convention over Configuration que se utiliza como soporte a la ejecución del patrón MVC.

El empleo del patrón MVC facilita el mantenimiento del sistema en caso de errores y crea independencia en el funcionamiento del mismo. Además provee una aplicación más escalable y la posibilidad de separar los datos de su representación visual.

3.3 Subsistema de diseño

Los subsistemas de diseño son una forma de organizar los artefactos del modelo de diseño en piezas más manejables. Un subsistema puede constar de clases del diseño, realizaciones de casos de uso, interfaces y otros subsistemas (recursivamente). [44] En la Figura 9 se muestra el artefacto subsistema de diseño del sistema, también conocido como diagrama de paquetes.

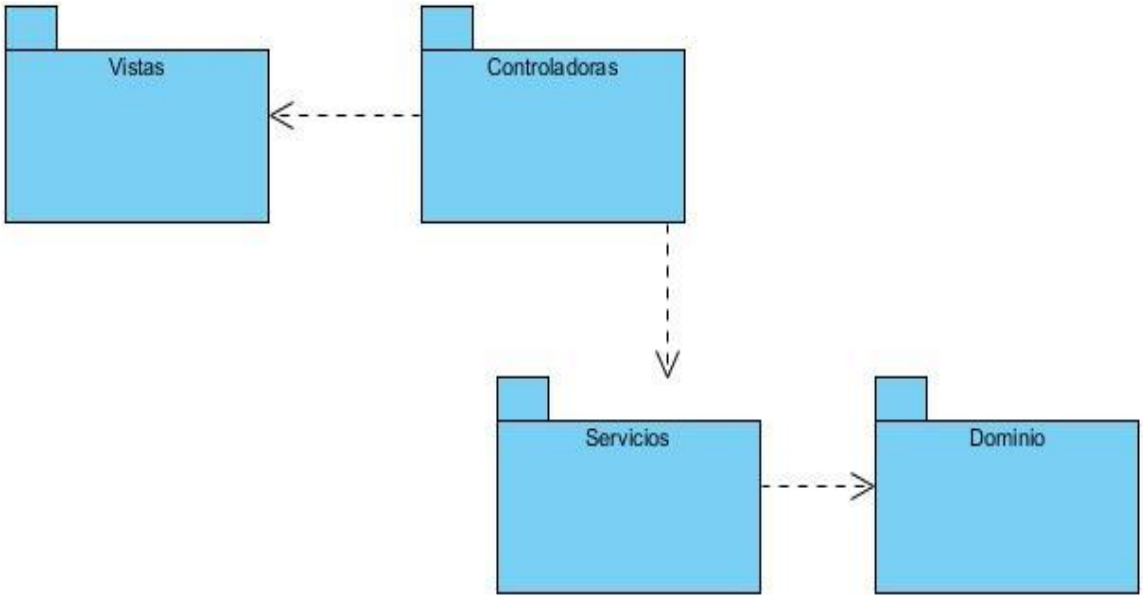

Figura 10: Subsistema de diseño

El paquete Vistas incluye como su nombre lo indica todas las vistas o páginas GSP (del inglés Groovy Server Pages). El paquete Controladoras contiene todos los controladores groovy responsables de la recepción y envío de mensajes desde y hacia las vistas respectivamente. Los controladores además hacen uso de las funcionalidades concernientes a la lógica de negocio implementadas en los servicios. Estos a su vez acceden a las clases de dominio para cumplimentar su objetivo. A partir de las clases de dominio se genera el modelo físico de datos.

3.4 Diagramas de clases del diseño

Un diagrama de clases es una presentación gráfica de la vista estática, que muestra una colección de elementos declarativos (estáticos) del modelo, como clases, tipos, y sus contenidos y relaciones. [50] En la Figura 10 se muestra el diagrama de clases del diseño web del caso de uso Identificar estado de los costos de calidad. En este caso de uso intervienen las vistas correspondientes a cada una de las categorías de los costos de calidad: prevención, evaluación, fallas internas y fallas externas. Además está presente la vista encargada de recoger los datos relativos a los costos incurridos en materia de creación en cada uno de los proyectos así como también la página GSP donde se mostrarán los indicadores que resultaron desfavorables. En el diagrama se aprecia igualmente la clase CalcularController, responsable de la recepción y respuesta de las solicitudes del usuario desde y hacia las mencionadas vistas. Esta clase controladora hace uso del servicio CalcularService que implementa toda la

lógica de negocio referente al modelo computacional sustentado en el Razonamiento Basado en Casos que permite identificar los indicadores de costos de calidad desfavorables. El mismo requiere de la mayor parte de las clases de dominio del sistema para llevar a cabo todas las funcionalidades que le conciernen, entre ellas: la determinación del estado de los indicadores de costos de calidad de un proyecto determinado.

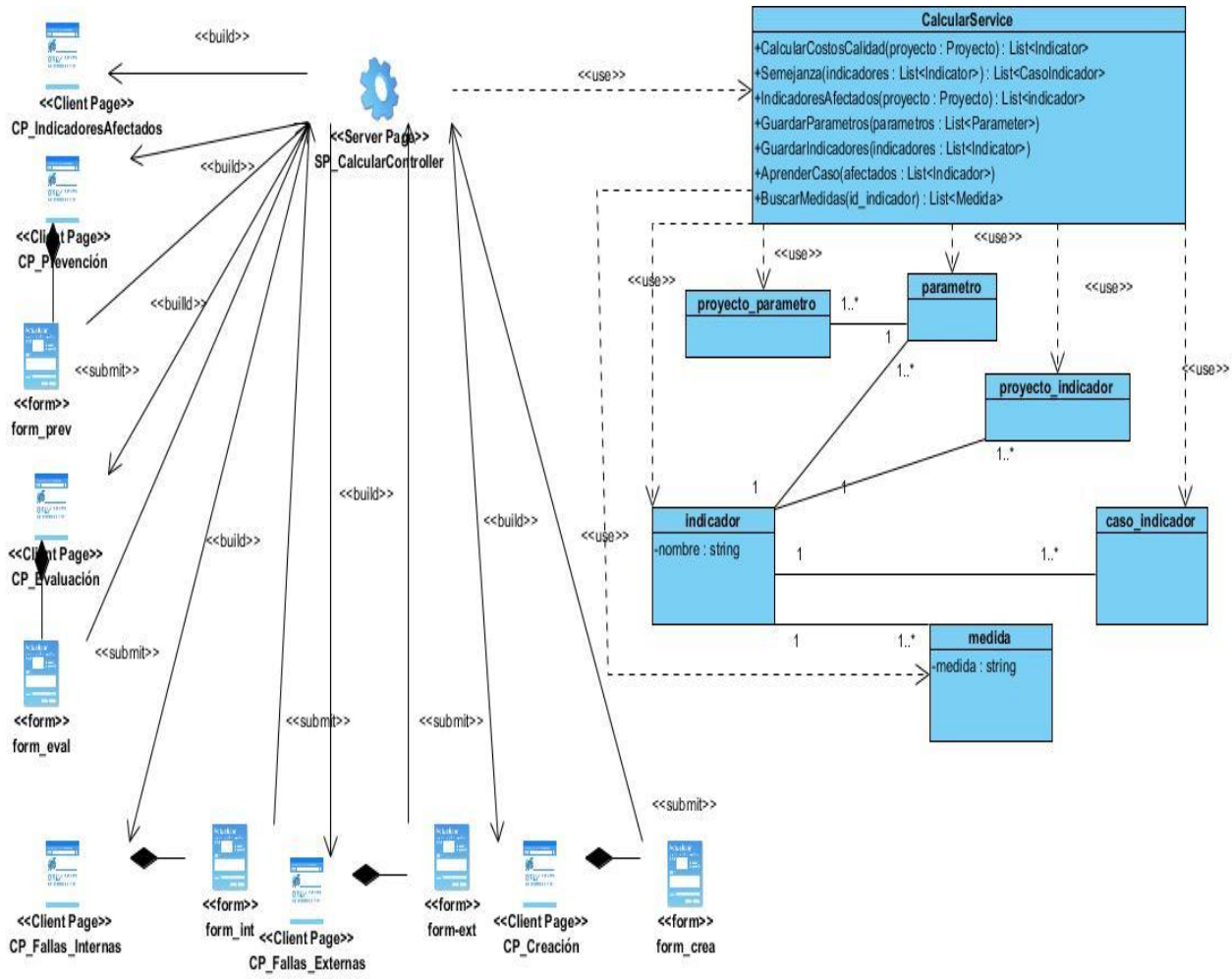

Figura 11: Diagrama de Clases del Diseño del CU Identificar estado de los indicadores de costos de calidad.

En el **Anexo 4** se expone el resto de los diagramas de clases de diseño.

3.5 Servicios principales

Los servicios en Grails son componentes destinados a la implementación de la lógica del negocio. De esta manera se separa el acceso a los datos de las clases controladoras. Siendo estas responsables únicamente del correcto flujo de información. Al llevar a cabo la

estratificación de responsabilidades los componentes del sistema serán más pequeños y fáciles de mantener. Además, el grado de reutilización de los mismos aumentará en gran medida.

3.5.1 CalcularService

Este servicio constituye el principal de todo el sistema pues es el que implementa íntegramente todo el RBC. Permite recuperar de la base de conocimiento el caso más semejante al actual, pudiendo dar respuesta al mismo reutilizando la solución almacenada. Incluye la lógica de negocio asociada al cálculo de los indicadores de costos de calidad así como al almacenamiento de los mismos, entre otros. En la Figura 12 se muestran las principales funcionalidades de dicho servicio.

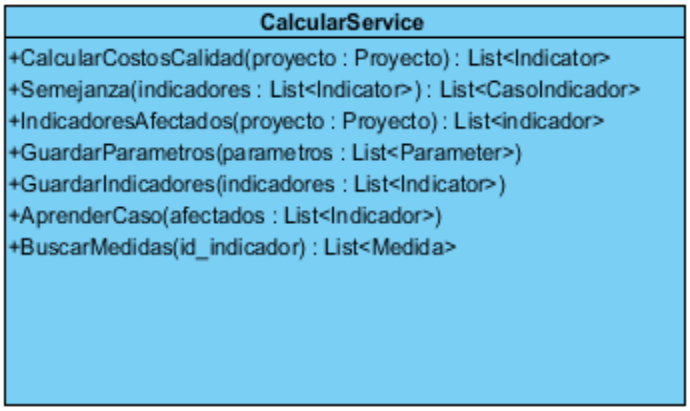

Figura 12: CalcularService

3.5.2 IndicadorService

Este servicio tiene como objetivo la gestión de los indicadores de costos de calidad. La misma abarca los procesos de inserción, actualización, eliminación y recuperación de los indicadores. En la figura 13 se muestran las principales funcionalidades de este servicio.

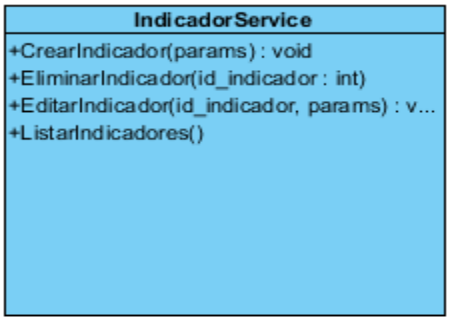

Figura 13: IndicadorService

3.6 Modelo físico de datos

El modelo físico de datos cuenta con un total de 10 tablas con el objetivo de almacenar todos los datos relevantes para el correcto funcionamiento del sistema. Estas tablas tienen su origen en las clases de dominio y relaciones definidas entre ellas. La tabla usuario tiene como función principal la persistencia de los datos de los usuarios del sistema. Un usuario pertenece a un proyecto y este a su vez puede tener asociados uno o muchos usuarios. Entre la tabla proyecto e indicador se establece una relación de muchos a muchos, generándose una tabla intermedia proyecto_indicador donde se guardará el valor del indicador para ese proyecto y si resultó afectado o no en el proceso de gestión de la calidad. Así mismo, entre las tablas proyecto y parámetro también existe una relación de muchos a muchos y por consiguiente una tabla intermedia proyecto_parametro donde se almacenará el valor de cada parámetro para un proyecto determinado. Un indicador pertenece a una categoría, puede tener uno o muchos parámetros y una o muchas medidas correctivas para su disminución. En la tabla caso_indicador se almacenan los casos resultantes de aplicar el procedimiento para el cálculo de los costos de calidad en el Centro de Telemática y del propio aprendizaje del sistema. En la Figura 13 se muestra el modelo físico de datos que recoge de manera más detallada todas las entidades de la BD del sistema y las relaciones entre ellas.

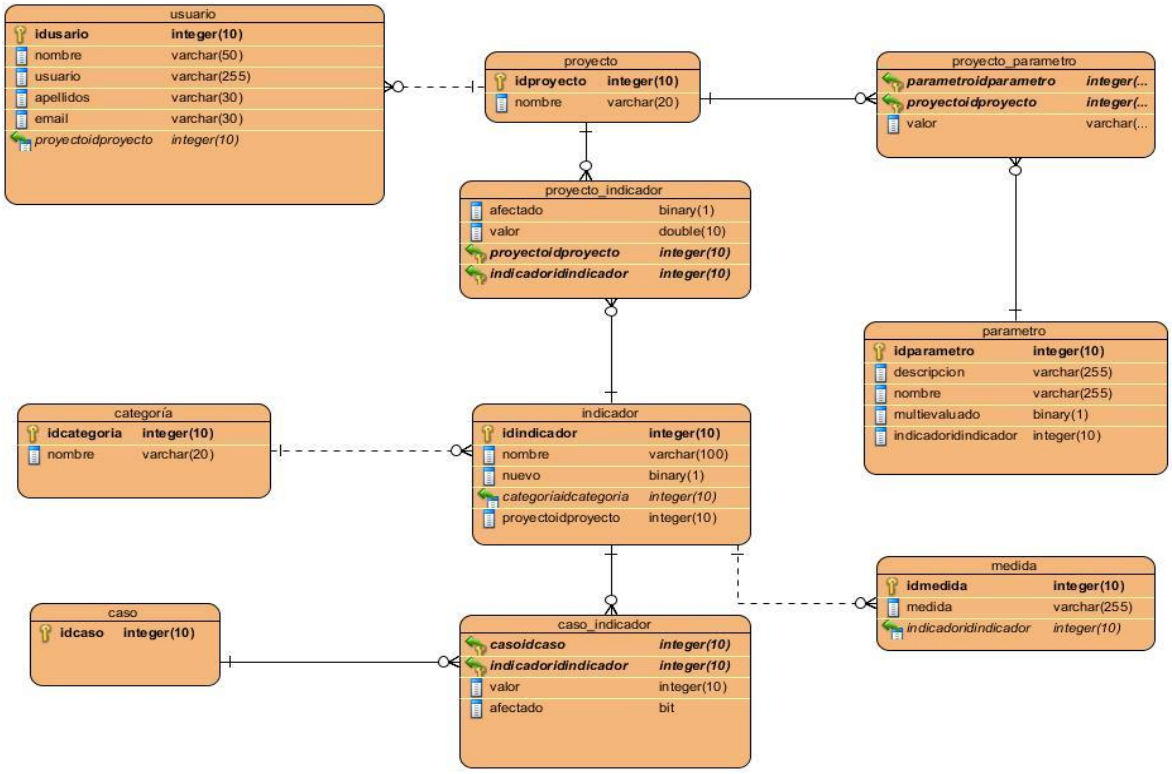

Figura 14: Modelo físico de datos.

3.6.1 Descripción de las tablas de la base de datos

3.6.1.2 Tabla usuario

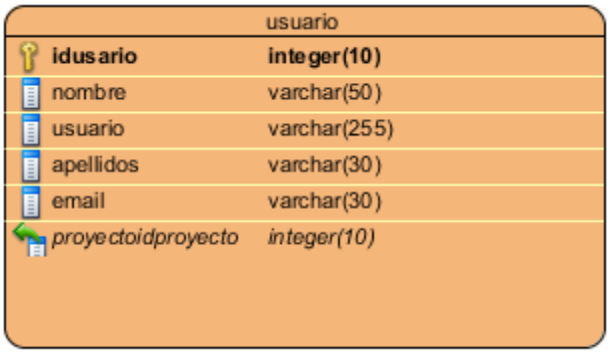

Figura 15: Tabla usuario.

Propósito: Almacenar los datos de los usuarios del sistema.

3.6.1.3 Tabla proyecto

Figura 16: Tabla proyecto.

Propósito: Almacenar los proyectos del Centro de TLM pudiendo establecer la pertenencia de los usuarios a estos.

3.6.1.4 Tabla proyecto_indicador

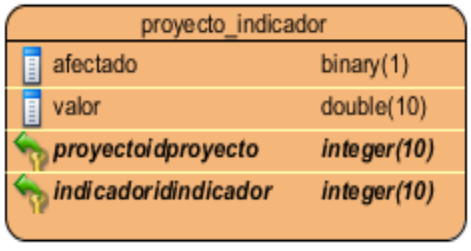

Figura 17: Tabla proyecto_indicador.

Propósito: Almacenar el valor de cada uno de los indicadores para cada uno de los proyectos del centro. Además guarda si un determinado indicador después de realizado el cálculo de los costos de la calidad de un proyecto resultó desfavorable o no.

3.6.1.5 Tabla categoría

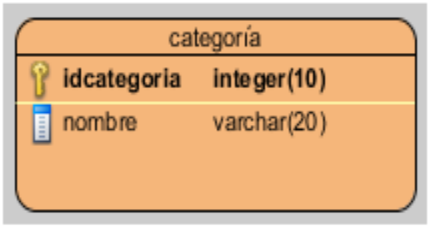

Figura 18: Tabla categoría.

Propósito: Almacenar las categorías a las que pertenecen los indicadores de costos de calidad.

3.6.1.6 Tabla medida

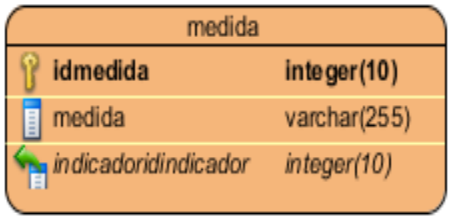

Figura 19: Tabla medida.

Propósito: Almacenar las medidas para disminuir cada uno de los indicadores de costos de la calidad.

3.6.1.7 Tabla indicador

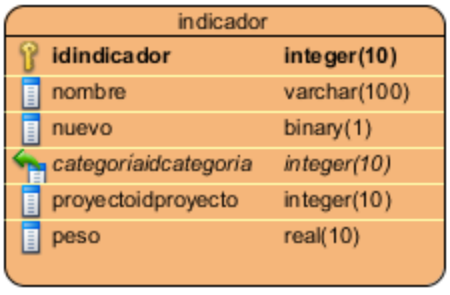

Figura 20: tabla indicador.

Propósito: Almacenar los indicadores de costos de calidad y el peso de cada uno de estos.

3.6.1.8 Tabla parámetro

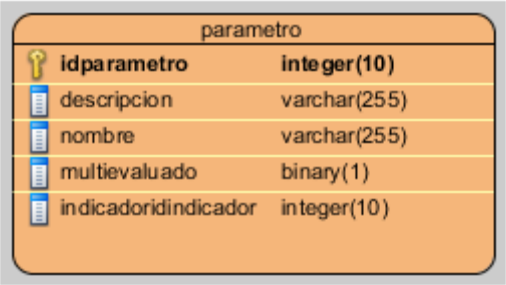

Figura 21: Tabla parámetro.

Propósito: Almacenar los parámetros para cada uno de los indicadores de costos de calidad.

3.6.1.9 Tabla proyecto_parametro

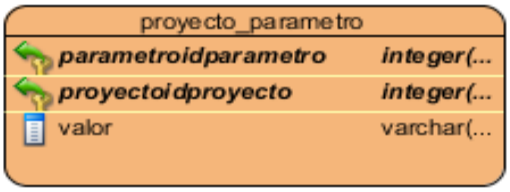

Figura 22: Tabla proyecto_parametro

Propósito: Permite almacenar el valor de un parámetro para un proyecto determinado.

3.6.1.10 Tabla caso_indicador

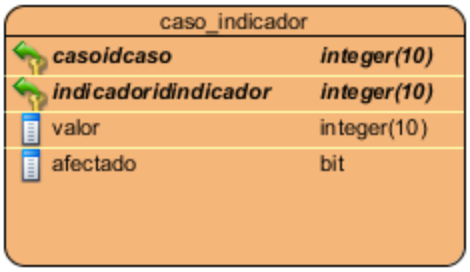

Figura 23: Tabla caso_indicador.

Propósito: Almacenar los casos resultantes del aprendizaje del sistema.

3.7 Diagramas de interacción

Un diagrama de interacción muestra gráficamente cómo los objetos interactúan a través de mensajes para realizar las tareas. [49] Los diagramas de interacción existentes son los de colaboración y secuencia. Los diagramas de colaboración destacan la organización estructural de los objetos que envían y reciben mensajes mientras que los de secuencia hacen mayor énfasis en el orden temporal de los mensajes. Con el uso de la herramienta Visual Paradigm es posible generar uno a partir del otro pues son semánticamente equivalentes. En el presente trabajo se confeccionaron como parte del modelo de diseño, los diagramas de secuencia. Los mismos se muestran en el **Anexo 5.**

3.8 Modelo computacional para la identificación de los indicadores de costos de calidad afectados.

La manera de representar y almacenar los indicadores de costos de calidad para identificar los que resultan desfavorables en los procesos de gestión de la calidad de los proyectos del Centro Telemática se realiza mediante casos. Los mismos están conformados por los rasgos predictores y los rasgos objetivos. Los rasgos predictores son los indicadores de costos de calidad definidos en el procedimiento (**Anexo 1**) y los objetivos, son los que resultan desfavorables. Cada rasgo tiene un valor numérico y un peso asociado. El sistema que se propone permitirá a los administradores de calidad del sistema definir este nivel de importancia para cada uno de los atributos.

La recuperación de casos es el proceso de búsqueda en la base de conocimiento de los casos más similares al actual. Para llevar a cabo este proceso es necesario un criterio para determinar la semejanza entre los casos, un algoritmo para buscar en la base de casos y un umbral.

3.8.1 Recuperación

La distancia Euclídea es una de las técnicas más empleadas en la etapa de recuperación en los sistemas de razonamiento basado en casos como medida de similitud. La misma se define entre dos casos ***x*** y ***y*** para todos sus atributos de la siguiente forma:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (1)$$

donde ***i*** constituye el atributo sobre el cual se está determinado la semejanza entre los dos casos y ***n*** representa la cantidad total de atributos.

Este método es aplicable en dominios donde todas las características tengan el mismo nivel de importancia, no siendo así para el dominio en el que se enmarca este trabajo. Todos los indicadores de costos de calidad no tienen la misma influencia en los procesos de gestión de la calidad. Para determinar la semejanza entre dos casos en el sistema que se propone, se emplea la distancia Euclídea [51] con una modificación: la ponderación de los atributos, representada en la ecuación como w. A menor distancia existirá mayor semejanza entre los casos ***x*** y ***y***.

$$D(x,y) = \sqrt{\sum_{i=1}^{n} w_i^2 (x_{i-} y_i)^2} \qquad (2)$$

La ponderación introducida proporciona protección contra características irrelevantes y potencia los atributos que mayor importancia tienen. El algoritmo de búsqueda que se empleó en la Recuperación utiliza la distancia Euclídea ponderada (2) para determinar la semejanza entre el caso actual y los almacenados en la base de conocimiento.

Los casos más semejantes al actual se recuperan a partir de un valor umbral. El mismo, para este modelo que emplea la distancia Euclídea ponderada, constituye una cota superior, ya que se trata de distancias. La selección de un umbral muy bajo implica la recuperación de los casos con un alto nivel de semejanza, en cambio presenta la desventaja de que no sea recuperado ninguno. En el presente trabajo se realiza el cálculo del valor umbral a partir de la semejanza entre los propios casos de la base de conocimiento. Para ello se construye una matriz de semejanza como se muestra en la Figura 33 donde las filas y las columnas representan los casos de la BC y la intersección de estas, la semejanza entre los casos en cuestión.

	Caso 1	**Caso 2**	**Caso 3**	**Caso n**
Caso 1	$D(C_1, C_1)$	$D(C_1, C_2)$	$D(C_1, C_3)$	$D(C_1, C_n)$
Caso 2	$D(C_2, C_1)$	$D(C_2, C_2)$	$D(C_2, C_3)$	$D(C_2, C_n)$
Caso 3	$D(C_3, C_1)$	$D(C_3, C_2)$	$D(C_3, C_3)$	$D(C_3, C_n)$
Caso n	$D(C_n, C_1)$	$D(C_n, C_2)$	$D(C_n, C_3)$	$D(C_n, C_n)$

Tabla 2: Matriz de semejanza.

Luego de construida la matriz, se procede a calcular el valor umbral mediante la fórmula extraída de la investigación [52]:

$$\mu = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} D(O_o, O_t) \qquad \textbf{(3)}$$

Donde:

μ : valor umbral.

m: cantidad de casos de la BC.

i: valor que recorrerá las filas.

j: valor que recorrerá las columnas.

$D(O_o, O_t)$: semejanza (calculada mediante la distancia Euclídea ponderada) entre los casos o y t.

3.8.2 Adaptación

Una vez que se han recuperado los casos más similares de la base de conocimiento se lleva a cabo generalmente la adaptación puesto que rara vez los casos almacenados coinciden con los nuevos. Esta etapa consiste en el proceso de modificar y combinar las soluciones de los casos recuperados para dar respuesta al nuevo problema.

Janet Kolodner en [53], Mitra en [54] y Bonzano en [55] hacen alusión a la siguiente clasificación de los métodos de adaptación:

Sustitución: Partes seleccionadas del caso almacenado son sustituidas con la ocurrencia de nuevos casos. Para determinar una sustitución apropiada, es requerido el conocimiento sobre el dominio del problema. [54]

Transformación: Se basa en el cambio estructural de la solución para lo cual es requerido al igual que en el método de sustitución, el conocimiento sobre el dominio del problema. [54]

Adaptación generativa: es la más compleja de las adaptaciones y requiere de una nueva realización del proceso de razonamiento en el contexto del nuevo problema. [55]

Como parte de los métodos de sustitución según Kolodner [53], se encuentran:

Reinstanciación: En este tipo de técnica se utiliza el marco o contexto de situaciones anteriores, pero con nuevos argumentos.

Ajuste de parámetros: Se ajustan los parámetros de la solución de casos anteriores de acuerdo con las diferencias entre las descripciones de los casos en cuestión.

Búsqueda local: Se realizan búsquedas dentro de jerarquías semánticas y se soluciona el problema por analogía.

En el presente trabajo, en la etapa de Reutilización, se hace uso del método de sustitución Ajuste de Parámetros.

3.9 Diagrama de componentes

Un diagrama de componentes muestra la organización y dependencias entre los diferentes componentes de un software. Son usados para mostrar la implementación estática de un sistema. Incluye el modelado de elementos concretos tales como ejecutables, bibliotecas de clases, tablas, archivos y documentos. En esencia, son diagramas de clases enfocados en los componentes de un sistema. Son importantes para la visualización, especificación y documentación de sistemas basados en componentes y para implementación de sistemas ejecutables. [50] En la figura 24 se muestra el diagrama de componentes del sistema.

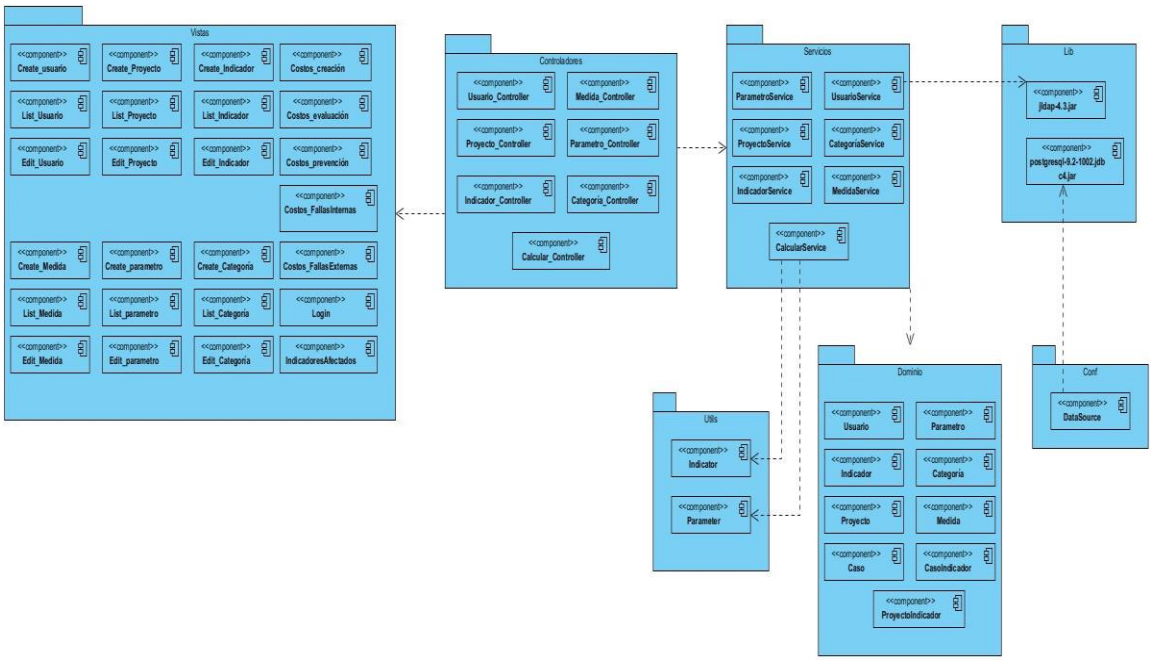

Figura 24: Diagrama de componentes

3.10 Descripción de los componentes

3.10.1 Paquete Vistas

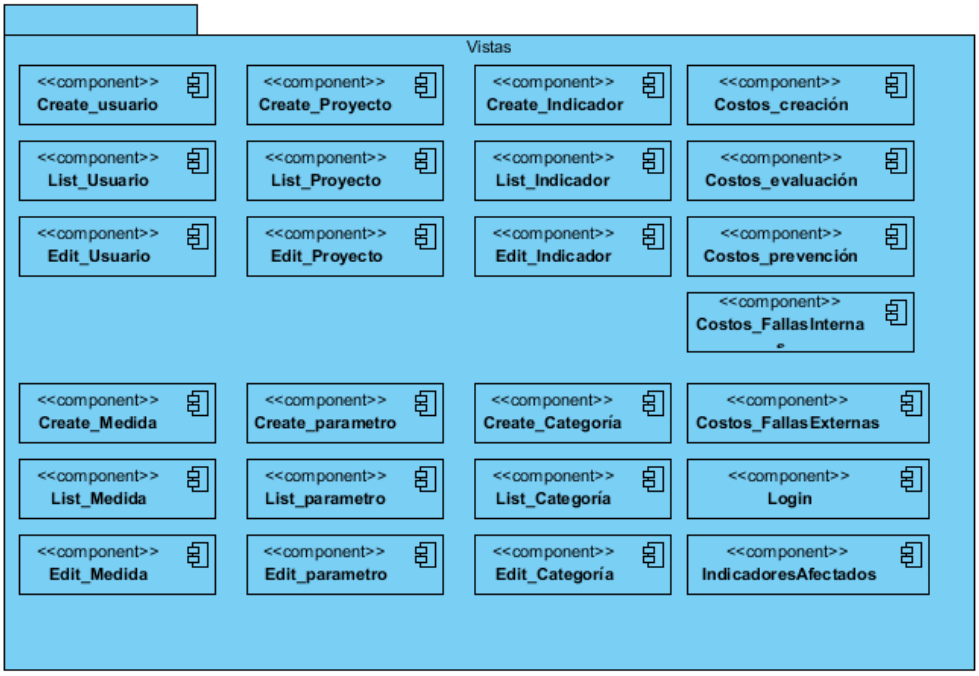

Figura 25: Paquete Vistas

El paquete de componentes Vistas agrupa todas las páginas GSP o vistas. Las mismas permiten la interacción del usuario con el sistema.

3.10.2 Paquete Controladores

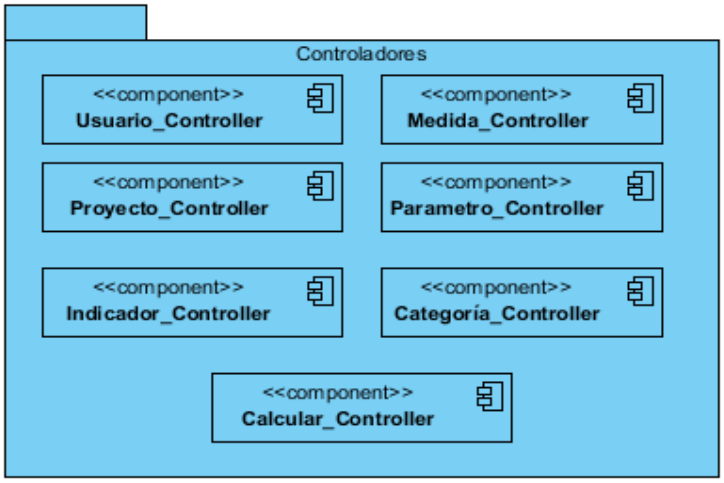

Figura 26:Paquete de componentes Controladores

El paquete de componentes Controladores agrupa todas las clases controladoras del sistema. Estas son las encargadas de manejar y responder las solicitudes del usuario, procesando la información y modificando el modelo y/o las vistas en caso de ser necesario.

3.10.3 Paquete Servicios

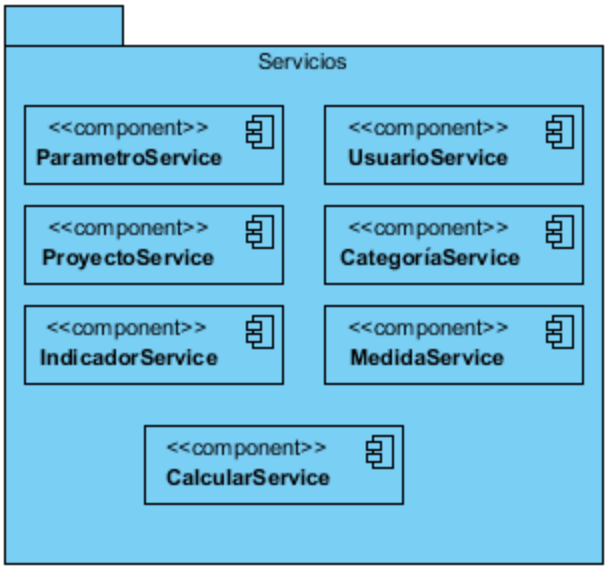

Figura 27: Paquete de componentes Servicios

Este paquete agrupa todas las clases de servicios del sistema. En estas se implementa la lógica de negocio.

3.10.4 Paquete Dominio

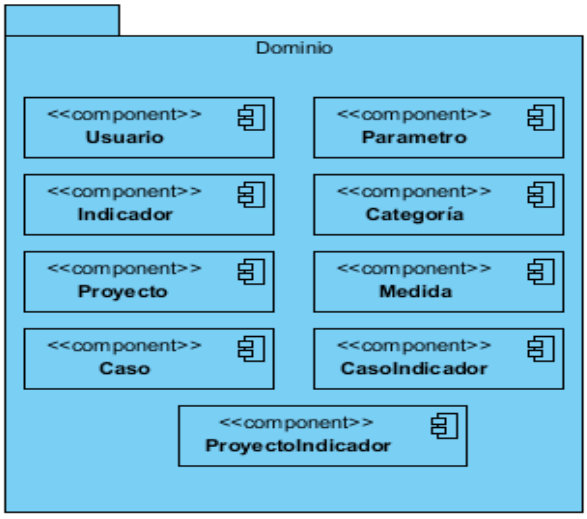

Figura 28:Paquete de componentes Dominio.

Las clases de dominio son los componentes que serán mapeados como tablas de la BD. En ellas se incluyen las restricciones a las que están sujetos cada uno de los atributos.

3.10.5 Paquete Lib

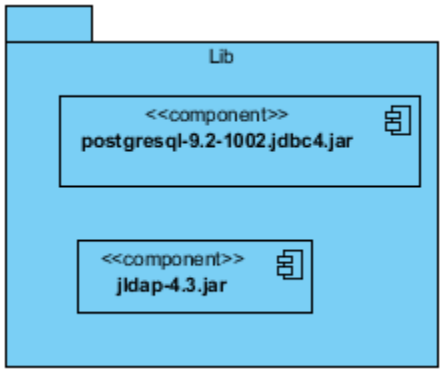

Figura 29:Paquete de componentes Lib.

Este paquete recoge las librerías que son empleadas en la implementación del sistema. Ellas son: el driver correspondiente al gestor de BD empleado y la librería que permite la conexión con el servidor LDAP.

3.10.6 Paquete Conf

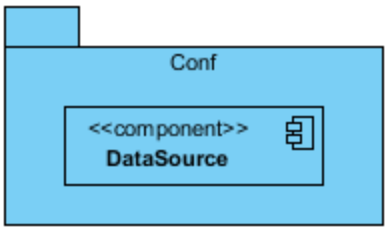

Figura 30: Paquete de componentes Conf.

Este paquete es utilizado para la configuración y definición del acceso local o remoto a los datos. En él se ubican ficheros groovy de configuración.

3.10.7 Paquete Utils

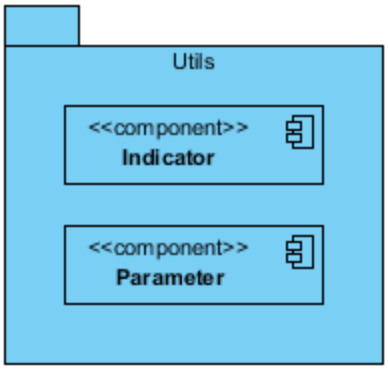

Figura 31: Paquete de componentes Utils.

Este paquete contiene dos clases auxiliares que se emplean en el cálculo de los costos de calidad. La primera de ellas Indicator se utiliza para guardar el identificador de un indicador y su correspondiente valor y la segunda Parameter tiene la misma función pero en este caso la información que se almacena es relativa a los parámetros de los indicadores.

3.11 Diagrama de despliegue

Un diagrama de despliegue es utilizado para representar la distribución física de un sistema. El mismo muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, las instancias de los componentes y objetos que residen en ellos. [50] En la figura 22 se muestra el diagrama de despliegue del sistema.

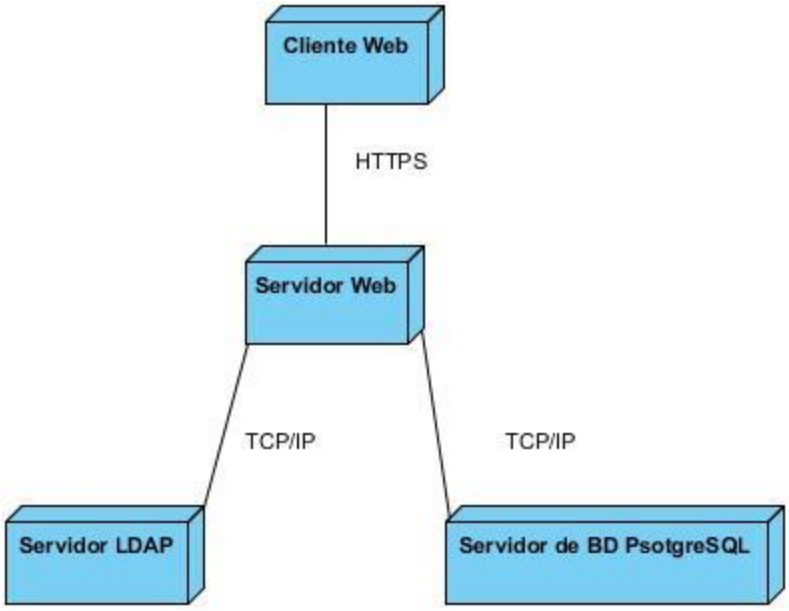

Figura 32: Diagrama de despliegue.

3.11.1 Descripción de los nodos del diagrama de despliegue

Nodo	Descripción
Cliente Web	Computadora que se conecta con el servidor web donde se encuentra la aplicación. Esta conexión se realiza a través del protocolo HTTPS.
Servidor Web	Computadora que tiene instalado un Servidor web. Se utiliza el Servidor Web Apache Tomcat en la versión 6.0 o superior y la Máquina Virtual de Java.
Servidor LDAP	Posibilita la autenticación de los usuarios con el sistema.
Servidor de BD PostgreSQL	Computadora que contiene la BD donde se encuentran almacenados los datos del sistema.

Tabla 3: Descripción de los nodos del diagrama de despliegue del sistema.

3.12 Pruebas

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. [56] Constituye un proceso de ejecución de un programa con la intención de descubrir un error. La misma no puede asegurar la ausencia de errores, solo puede demostrar que existen defectos en el software.

3.12.1 Métodos de pruebas

Pruebas de Caja Negra

Se enfocan en los requisitos funcionales del sistema y se aplican sobre la interfaz de usuario. No consideran la codificación dentro de sus parámetros a evaluar, es decir, no están basadas en el conocimiento interno del programa.

Pruebas de Caja Blanca

A diferencia de las Pruebas de Caja Negra, se basan en la lógica interna del código del sistema. En este método se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos que surgen debido a estructuras condicionales y/o bucles.

3.12.2 Niveles de pruebas

Los niveles de prueba existentes son los siguientes:

- Prueba unitaria
- Prueba de integración
- Prueba de funcionalidad
- Prueba de sistema
- Prueba de aceptación
- Prueba de instalación

Prueba unitaria: Es realizada con el objetivo de verificar la buena estructura y el correcto funcionamiento de una parte determinada del código. Está enfocada a los elementos más pequeños del software que se pueden probar. Está orientada al método de prueba Caja Blanca.

Prueba de sistema: Las pruebas de sistema engloban las pruebas funcionales y no funcionales. Las primeras tienen comprobar objetivo probar que los sistemas desarrollados cumplen con las funciones específicas para los que han sido creados. Confirman que los requerimientos previstos han sido satisfechos tal y como se especifica en los documentos de diseño funcional. Las no funcionales hacen referencia a la seguridad, rendimiento, usabilidad, entre otros parámetros. [57]

Prueba de aceptación: Las pruebas de aceptación comparan el comportamiento del sistema con los requisitos del cliente, sea cual sea la forma en que estos se hayan expresado. El cliente realiza, o especifica, tareas típicas para comprobar que se satisfacen sus requisitos o que la organización los ha identificado para el mercado al que se destina el

software. Esta actividad de pruebas puede incluir o no a los desarrolladores del sistema. [58] Existen dos tipos de prueba en este nivel: las pruebas alfa y las pruebas beta. Las pruebas alfa consisten en hacer pruebas funcionales como clientes en un entorno de desarrollo. Se trabaja en un entorno controlado y siempre se cuenta con la asistencia de un experto para ayudar al cliente a usar el sistema y para analizar los resultados. Las pruebas beta se realizan después de las pruebas alfa, y se desarrollan en el entorno de producción; un entorno que está fuera del control de los desarrolladores. Aquí el cliente trabaja de forma independiente con el producto y trata de encontrar fallos (reales o imaginarios) de los que informa al desarrollador. [59]

3.12.3 Estrategia de prueba seguida

La estrategia de prueba seguida incluye tres niveles: prueba unitaria, prueba de sistema y prueba de aceptación. Las pruebas del primer nivel se llevarán a cabo con el empleo del framework JUnit. En el segundo nivel de prueba se diseñarán los casos de prueba de Caja Negra con el objetivo de probar las funcionalidades del sistema. Por último, se realizarán las pruebas de aceptación alfa como parte de las pruebas de aceptación.

Con el comando **grails test-app**, JUnit genera todas las pruebas unitarias al sistema, brindando un reporte en formato html sobre los resultados de las mismas. En el **Anexo 6** se muestra el resumen de estos resultados. En el caso de las pruebas unitarias a los servicios es el desarrollador el encargado de implementarlas en las clases respectivas creadas por el framework JUnit. Esto se debe a que las pruebas unitarias tienen como objetivo la verificación del correcto funcionamiento de los métodos sin tener en cuenta su entorno. De ahí que Grails no inyectará ninguno de los métodos con los que cuenta el sistema durante su ejecución.

Las pruebas de sistema que componen la estrategia de prueba seguida están dirigidas a las pruebas de funcionalidad. Este tipo de pruebas se enfoca en los requisitos funcionales del software y está orientado a pruebas de Caja Negra. En el **Anexo 7** se expone el diseño de casos de prueba para el sistema.

En la realización de la prueba de aceptación alfa se aplicó el sistema en 3 de los 6 proyectos de desarrollo de software existentes en el centro pudiendo identificar los indicadores de costos de calidad afectados en los procesos de gestión de la calidad de cada uno de ellos. Este proceso fue validado por los líderes de estos proyectos, el jefe y la especialista de calidad del centro. Las cartas de aceptación se presentan en el **Anexo 8**.

Conclusiones del capítulo

En el presente capítulo siguiendo la metodología RUP se llevó a cabo el diseño del sistema a través del modelo de datos, los diagramas de clases del diseño y los diagramas de interacción. De esta manera quedó confeccionado el modelo de diseño, entrada fundamental para proceder a efectuar la implementación del sistema. Se definieron como patrones de diseño a emplear: Singleton, Decorator e Inversión del Control y como patrón arquitectónico MVC. Se definió el modelo computacional que permite identificar los indicadores de costos de calidad afectados en los procesos de gestión de la calidad de los proyectos del Centro TLM. El mismo emplea la distancia Euclídea ponderada como medida de semejanza entre los casos, en la Recuperación de estos de la base de conocimiento así como la técnica Ajuste de parámetros en el proceso de Reutilización. A través del modelo de implementación se evidenció cómo los elementos del modelo de diseño se implementan en términos de componentes, representando y describiendo la distribución física de estos últimos. Además se aplicó el sistema en 3 de los 6 proyectos de desarrollo existentes en el centro, siendo validado por los líderes de estos proyectos, el jefe y la especialista de calidad del centro.

Conclusiones

En el presente trabajo se describe la solución desarrollada con el objetivo de identificar los indicadores de costos de calidad afectados en los procesos de gestión de calidad del Centro de TLM. Una vez finalizada la investigación se concluye que:

- Se realizó un estudio de las herramientas existentes en el mundo para la gestión de los costos de la calidad concluyendo que ninguna de ellas permite identificar los indicadores de costos de calidad afectados en los procesos de gestión de calidad del Centro de Telemática.
- Se formalizó un modelo que utiliza la técnica de IA llamada Razonamiento Basado en Casos, la distancia Euclídea ponderada como medida de semejanza en la Recuperación de los casos de la base de conocimiento y el método de sustitución Ajuste de parámetros en la Reutilización para identificar los indicadores de costos de calidad afectados en los procesos de gestión de la calidad del Centro de Telemática.
- Se llevó a cabo el desarrollo del sistema que emplea el modelo computacional para identificar los indicadores de costos de calidad afectados en los procesos de gestión de la calidad generándose los artefactos de los flujos de trabajo Diseño e Implementación, propuestos por la metodología de desarrollo seleccionada, RUP. En la implementación del mismo se utilizaron: el framework del lado del servidor Grails en la versión 2.0.1 basado en el lenguaje Groovy, Bootstrap como framework del lado del cliente, IntelliJ IDEA como Entorno Integrado de Desarrollo y PostgreSQL como SGBD.
- Se aplicó el sistema en 3 de los 6 proyectos de desarrollo de software existentes en el centro pudiendo identificar los indicadores de costos de calidad desfavorables en los procesos de gestión de la calidad de cada uno de ellos, siendo este proceso validado por los líderes de estos proyectos, el jefe y la especialista de calidad del centro.

Recomendaciones

- Realizar los procesos de recuperación y adaptación considerando la incertidumbre involucrada en la información.
- Investigar sobre las ventajas de la vinculación entre el sistema basado en casos que se propone en este trabajo y otras técnicas de IA como las redes neuronales y los algoritmos genéticos en el aprendizaje del sistema que se propone.
- Implementar el proceso de revisión del Razonamiento Basado en Casos del sistema que se propone.

Bibliografía

Bello, Rafael. *Solución de problemas bajo incertidumbre.* 1998.

Bonzano, A. *ISAC: a Case-Based Reasoning System for Aircraft Conflict Resolution*. A thesis submitted to the University of Dublin, Trinity College, for the degree of Doctor in Philosophy, 1998.

Castillo E., Gutiérrez J.M. y Hadi A.S. *Sistemas Expertos y Modelos de Redes Probabilísticas*. Academia de Ingeniería, Madrid, 1997.

Chrissis, Mary Beth, Konrad, Mike y Shrum, Sandy. *Guía para la integración de procesos.Segunda edición. Madrid : Endesa, 2009.*

Cordero, Dasiel y Torres, Yoanny. *Sistema Inteligente de Mitigación de Riesgos para el Centro ISEC,* Tesis de pregrado, La Habana, 2011.

García, M. R. B. *El empleo del razonamiento basado en casos en el desarrollo de Sistemas basados en el conocimiento para el diagnóstico*. UCLV, 1997, Cuba.

Gutiérrez, Iliana, Bello, Rafael E. y Tellería, Andrés. *Un Sistema Basado en Casos para la toma de decisiones en condiciones de incertidumbre*, Santa Clara, Cuba: s.n., 2002, Vol. 23.

Juran, J. M. y Gryna, F. M. *Análisis y planeación de la calidad*. Tercera Edición. México: Mc Graw Hill, 1995

Juran, Joseph. *Quality Control Handbook.* s.l. : Mcgraw-Hill, 1999. ISBN-13: 978-0070331761.

Kolodner, J*: Case-Based Reasoning*, Ed Morgan Kaufmann, San Mateo CA, 1993, 199.

Kolodner, J. *Maintaining organization in a dynamic long-term memory*. Cognitive Science, 7, 1983.

Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. 2da. México: Pearson Education, 2003. pág. 148. 8420534382.

Leake, D. *Case-Based Reasoning: experiences, lessons, and future directions*. AAAI Press / The MIT Press, Menlo Park, CA, 1996.

López de Mántaras, R. Retrieval, reuse, revision, and retention in case based reasoning. The Knowledge Engineering Review, Vol. 00:0, 1–2. 2005, Cambridge University Press DOI: 10.1017/S000000000000000 Printed in the United Kingdom

Martínez, Natalia, García, María M y Zenaida, Zoila. *El Paradigma del Razonamiento Basado en Casos en el Ámbito de los Sistemas de Enseñanza/Aprendizaje Inteligentes.* Santa Clara, Cuba: s.n., 2009. No. 30.

Mitra, R. y Basak, J. *Methods of Case Adaptation*. A Survey. International Journal of Intelligent Systems, vol. 20, 627-645, 2005.

Peña Ayala, Alejandro. 2006. *Sistemas Basados en Conocimiento.* México D.F : Dirección de Publicaciones , 2006. 970-94797-4-1.

Pressman, R. S. *Ingeniería del Software. Un enfoque Práctico.* México, McGraw-Hill: Version 1.1. CMMI. C. M. S. E. Institute. Pittsburgh, Carnegie Mellon Software Engineering Institute. 2002.

Rivera Rodríguez, Sergio Michel. *Modelo de un Sistema de Razonamiento Basado en Casos para el Análisis en la Gestión de Riesgos*, Tesis de Maestría, Unidad de Compatibilización, Integración y Desarrollo de Productos Informáticos para La Defensa, La Habana,2010

Referencias bibliográficas

1. **Caballano Álcantara, José Luis.** www.caballano.com. [En línea] *www.caballano.com.* [Citado el: 27 de Noviembre de 2012.] http://www.caballano.com/ca.htm.
2. **Romero López, Javier Álvaro.** *Principios de Contabilidad.* México : Mc Craw-Hill, 2003.
3. **Straub, P.** *El costo de la calidad y el costo de la mala calidad,* 2006.
4. **Juran, Joseph.** *Quality Control Handbook.* s.l. : Mcgraw-Hill, 1999. ISBN-13: 978-0070331761.
5. **ASQC (American Society for Quality Control)** *Quality Costs What and How.* Committee for Quality cost, Milwaukee, 1974.
6. **Gutiérrez Martínez, I., Bello Pérez, R., y Tellería Rodríguez, A.** Vol 2,No.2. Un *Sistema Basado en Casos para la toma de decisiones en condiciones de incertidumbre.* Revista Investigación Operacional, 2002, 103-121.
7. **Sitio web de la UCI**. www.uci.com [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://www.uci.com
8. **Pérez, Yeisis y Jiménez, Jorge Félix**. *Propuesta de procedimiento para determinar los Costos Totales de la Calidad en los proyectos de software del Centro de Telemática de la UCI*. Tesis de pregrado, UCI, Ciudad de la Habana, 2010.
9. **Ishikawa, Kaoru.** *¿Qué es el control total de calidad? La modalidad japonesa.* Décima reimpresión. Colombia: Norma, 1998
10. **Norma Internacional ISO 9000:2000** *Sistemas de Gestión de la Calidad-Fundamentos y vocabulario*. © ISO 2000. Ginebra. Zuiza. 2000. www.iso.ch
11. **Juran, J. M. y Gryna, F. M.** *Análisis y planeación de la calidad*. Tercera Edición. México: Mc Graw Hill, 1995
12. **Deming.** *Calidad, Productividad y Competitividad.* La salida de la crisis. Díaz de Santos. Madrid, 1993
13. **Pressman, Rogers.** Ingeniería del Software. *Un enfoque práctico*. Quinta Edición
14. **Krasner H**.: *Using the Cost of Quality Approach for Software*. CROSSTALK The Journal of Defense Software Engineering, 1998
15. **Ashrafi N.:** *The Impact of Software Process Improvement on Quality: in Theory and Practice. Information & Management* No. 40, 2003
16. **Rubey R., Browning L., Roberts A.:** *Cost Effectiveness of Software Quality Assurance.* IEEE, 1989

17. **Universidad Veracruzana.** [En línea] 2013. [Citado el: 25 de Enero de 2013.] www.uv.mx/gestion/proyectos/documents/AurelianoAguilarBonilla.pdf.
18. **Talavera Pleguezuelos, Clemente**. Aiteco Consultores. [En línea] [Citado el: 8 de Junio de 2013.] http://www.aiteco.com/ciclo-pdca-de-mejora-continua/.
19. **Morera Cruz, José Orlando**. [En línea] *Mejoramiento Continuo*. 2002. Disponible en: http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/meconti.htm
20. **Chrissis, Mary Beth, Konrad, Mike y Shrum, Sandy.** *Guía para la integración de procesos.Segunda edición. Madrid : Endesa, 2009.*
21. **Tejada, D. R.** *Sistema de costos de calidad en el proceso de Mantenimiento e instalación de aire acondicionado*.Obtenido de Biblioteca USAC Universidad de San Carlos de Guatemala: biblioteca.usac.edu.gt/tesis/08/08_2157_IN.pdf, 2010
22. **Asesoría en sistemas de calidad y mejora continua.** Obtenido de www.mpasesorias.cl/Files/CostosCalidad2.pdf,2013.
23. **Curso Costos de calidad**. Obtenido de Servicios de Consultoría Dr. Primitivo Reyes: www.icicm.com/files/CursoCostosCalidad.doc
24. **PLUNKETT, B. G.** *Quality* Costing, 1992.
25. **Inteligencia Artificial** [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://www.inteligenciaartificial.cl/ciencia/software/ia/inteligencia_artificial.htm
26. **Jornada sobre Inteligencia Computacional Aplicada al Negocio, JICAN** http://jican.iti.upv.es/index.php/es/que, 2009.
27. **Arcia Montes de Oca J.** *Modelo de un Sistema Basado en Casos para el diagnóstico del Síndrome de Maloclusión.* [Citado el: 12 de Noviembre de 2012]. Disponible en: http://www.uvfajardo.sld.cu/Members/joel_arcia/plonearticlemultipage.2007-01-04.6812930506/3-2-modelo-de-la-maquina-de-inferencia.]
28. **Castillo E., Gutiérrez J.M. y Hadi A.S.** *Sistemas Expertos y Modelos de Redes Probabilísticas*. Academia de Ingeniería, Madrid, 1997.
29. **Sucar, L. E.**. Ciencias Computacionales del Insituto Nacional de Astrofísica,Óptica y Electrónica. Obtenido de http://ccc.inaoep.mx/~esucar/Clases-mgp/caprb.pdf, 2013
30. **López de Mántaras, R.** Retrieval, reuse, revision, and retention in case based reasoning. The Knowledge Engineering Review, Vol. 00:0, 1–2. 2005, Cambridge University Press DOI: 10.1017/S000000000000000 Printed in the United Kingdom
31. **Bernillion y Cerrutti O**. *Implantar la calidad total ,* Ed Gestión 2000, 1993.
32. **Knoow**. [En línea] [Citado el: 8 de Junio de 2013.] http://www.knoow.net/es/cieeconcom/gestion/gestiondeproyectos.htm.

33. **Geofisica**. [En línea] [Citado el: 8 de Noviembre de 20 (Sucar, 2013)12.] http://www.geofisica.cl/macam/project2.htm.
34. **Centro de Gestión de Estudios Estratégicos**. www.cgee.org.br. [En línea] [Citado el: 8 de Noviembre de 2012.] www.cgee.org.br/atividades/redirKori/4756.
35. **Rodríguez, Alonso y Rodríguez, Sarabia**: *"Calidad+IA, software basado en inteligencia artificial para la gestión de la calidad en la producción de habanos."* en Contribuciones a la Economía, mayo 2009 en http://www.eumed.net/ce/2009a/
36. **Rodríguez, Y.** *Generalización de la métrica basada en la diferencia de valores (VDM) para variables lingüísticas y su aplicación en sistemas basados en el conocimiento.* s.l. : UCLV, 2007.
37. **Rodríguez, Y., y otros.** *Connectionist Fuzzy Case-Based Reasoning Model,* 2006
38. **Sánchez, D. C.** *Diseño del Sistema de Costos de Calidad bajo el enfoque de procesos en la empresa Empleadora del Níquel en Moa.* Ciencia & Futuro, 2012, 45-53.
39. **Dierk Konig, Andrew Glover**. *Groovy in Action.* s.l. : Manning Publications Co, 2007.
40. **GenbetaDev**. *Sitio web GenbetaDev.* [Online] [Cited: Enero 17, 2013.] http://www.genbetadev.com/frameworks/bootstrap.
41. **Sitio oficial JetBrains.** Obtenido de Sitio oficial JetBrains: http://www.jetbrains.com/idea/features/index.html, 2013
42. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software*. Madrid: Addison Wesley, 2000. 84-7829-036-2.]
43. **Machado López, Dayris.** *Propuesta de acciones para disminuir los costos de calidad en los proyectos del Centro de Telemática y diseño de una aplicación que informatice el cálculo de los costos de calidad,* 2011.
44. **Jacobson, I., Booch, G., y Rumbaugh, J.** *El proceso unificado de desarrollo de software.* Madrid: Pearson Educación, 1999
45. **Cristopher, A.** *A pattern language,* 1997
46. **Gamma, E. Helm, R. Jonson, R. y Vlissides, J.** *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
47. **Smith, G., y Ledbrook, P.** *Grails in action.* Greenwich: Manning Publications Co., 2009
48. **Brito Calahorro, N.** *Manual de desarrollo web con Grails*, 2009
49. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. 2da. México: Pearson Education, 2003. pág. 148. 8420534382.

50. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar.** *The Unified Modeling Language User Guide.* Massachusetts: Addison-Wesley Longman Inc., 1998. 0-201-57168-4.
51. **García Herrero, Jesús.** *Otras técnicas de análisis y minería de datos*. Universidad Carlos III de Madrid, 2006.
52. **Martínez Sánchez, Dra. Natalia, García Lorenzo, Dra. Maria Matilde y García Valdivia, Dra. Zenaida.** *Modelo para diseñar sistemas de enseñanza-aprendizaje inteligentes utilizando un razonamiento basado en casos,* 2009.
53. **Kolodner, J***: Case-Based Reasoning*, Ed Morgan Kaufmann, San Mateo CA, 1993, 199.
54. **Mitra, R. y Basak, J**. *Methods of Case Adaptation*. A Survey. International Journal of Intelligent Systems, vol. 20, 627-645, 2005.
55. **Bonzano, A**. *ISAC: a Case-Based Reasoning System for Aircraft Conflict Resolution*. A thesis submitted to the University of Dublin, Trinity College, for the degree of Doctor in Philosophy, 1998.
56. **Software, C. d. a. D. d. l. y. G. d.** "*Fase de Elaboración. FT Prueba (procedimientos genéricos y aplicación de algunos tipos de pruebas simples).",* 2005 Métrica, Metodología.
57. **INSA**. *Ingeniería de Software Avanzada S.A.* [En línea] 2013. [Citado el: 8 de Junio de 2013.] http://www.insags.com/servicios/calidad_pruebas_software.html.
58. **Ovalle, C.** Obtenido de Creatividad y Tecnología: http://creatividadytecnologia.com/web/wp-content/uploads/2012/09/Capitulo-5-Pruebas-del-Software.pdf, 2013
59. **TestHouse Consultores.** [En línea] 2013. [Citado el: 8 de Junio de 2013.] http://www.es.testhouse.net/pruebas-de-aceptacion.