Sistema para auditar el cumplimiento de CMMI-SW nivel 2.

César Gabriel Vargas[1] **Germán Biagioli**[2]

Trabajo final para obtener el grado de

Licenciado en Informática / Licenciatura en Sistemas

De la
Facultad de Informática,
Universidad Nacional de La Plata,
Argentina

Director: Dr. Gustavo Rossi
Co-Director: Dr. Alejandro Fernández

La Plata, Junio de 2009

[1] Nro. Alumno: 3965/4 – varces@gmail.com
[2] Nro. Alumno: 2612/5 – germanbiagioli@gmail.com

UNIVERSIDAD NACIONAL DE LA PLATA

RESUMEN

SISTEMA PARA AUDITAR EL CUMPLIMIENTO DE CMMI-SW NIVEL 2.

El objetivo de este trabajo es construir una herramienta para certificar la madurez de procesos que requiere CMMI nivel 2, ayudando en la tarea diaria a la persona que realiza aseguramiento de la calidad (QA). El foco de la misma estará en la generación de evidencia de la aplicación de control de procesos a través del seguimiento de las auditorías y del registro y tratamiento de las no conformidades.

El producto propuesto en esta tesis, es una herramienta que podrá utilizarse en el área de QA de una fábrica de sistemas.

Índice General

Agradecimientos ..... - 3 -
Índice General ..... - 4 -
Capítulo 1 ..... - 6 -
1.1 Introducción ..... - 6 -
1.2 Objetivo ..... - 7 -
1.3 Motivación ..... - 8 -
1.4 Organización del documento ..... - 8 -
Capítulo 2 – Ideas generales de CMMI-SW ..... - 10 -
2.1 Introducción ..... - 10 -
2.2 Procesos ..... - 10 -
2.3 Modelo de procesos ..... - 10 -
2.4 CMM ..... - 11 -
2.5 Madurez vs. Inmadurez ..... - 12 -
2.6 CMMI ..... - 13 -
2.7 Evolución de madurez de los procesos CMMI ..... - 13 -
2.9 Descripción del modelo ..... - 16 -
2.10 Representaciones ..... - 18 -
2.11 Por qué se aplica CMMI ..... - 20 -
Capítulo 3 - Aseguramiento de la calidad del software ..... - 22 -
3.1 Introducción ..... - 22 -
3.2 Calidad del software ..... - 22 -
3.3 Aseguramiento de la calidad del software ..... - 26 -
3.3.1 Surgimiento de SQA (Software Quality Assurance) ..... - 26 -
3.3.2 Definición de SQA (Software Quality Assurance) ..... - 28 -
3.3.3 SQA no es lo mismo que SQC (Software Quality Control) ..... - 29 -
3.3.4 Funciones generales del SQA ..... - 30 -
3.3.5 Consideraciones ..... - 32 -
3.5 Aseguramiento de la calidad de procesos y productos en CMMI (PPQA). - 33 -
3.5.1 Introducción ..... - 33 -
3.5.2 Metas y prácticas. ..... - 33 -
Capítulo 4 – Certificación CMMI-SW ..... - 36 -
4.1 Introducción ..... - 36 -
4.2 Requerimientos para la certificación de CMMI ..... - 37 -
4.2.1 Introducción ..... - 37 -
*4.2.2 Principios y conceptos fundamentales de ARC (*Appraisal Requirements for CMMI*).* ..... - 37 -
4.2.3 Tipos de indicadores de implementación de prácticas ..... - 41 -
*4.2.4 Estructura de los diferentes métodos de ARC (*Appraisal Requirements for CMMI*).* ..... - 42 -
4.3 SCAMPI - Método Estándar de Evaluación de CMMI para Mejora de Procesos ..... - 44 -
4.3.1 Introducción ..... - 44 -
4.3.2 Objetivos ..... - 45 -
4.3.3 Clases de Métodos ..... - 46 -
4.4 SCAMPI Leader Appraiser ..... - 47 -
Capítulo 5 - Presentación del sistema ..... - 48 -

5.1 Introducción ..... - 48 -
5.2 Objetivo del sistema ..... - 49 -
5.3 Funcionalidad ..... - 49 -
5.3.1 Casos de uso ..... - 50 -
5.4 Arquitectura y diseño de alto nivel ..... - 68 -
5.4.1 Arquitectura. ..... - 68 -
5.4.2 Diseño de alto nivel ..... - 77 -
5.5 Detalles de implementación ..... - 79 -
5.6 Herramientas de desarrollo ..... - 84 -
Capítulo 6 - Caso de estudio ..... - 87 -
6.1 Definición del dominio y el alcance ..... - 87 -
6.2 Aplicación de la herramienta al caso de estudio ..... - 90 -
6.2.1 Estados ..... - 90 -
6.2.2 Tipos de Proyectos ..... - 91 -
6.2.3 Tipos de Auditorías y sus tipos de tareas ..... - 91 -
6.2.4 Roles y Niveles de escalamiento de no conformidad ..... - 94 -
Capítulo 7 – Conclusiones. ..... - 95 -
7.1 Sobre el trabajo ..... - 95 -
7.2 Generales ..... - 96 -
7.3 Personales ..... - 97 -
Capítulo 8 - Trabajos futuros ..... - 99 -
Capítulo 9 - Apéndice ..... - 100 -
9.1 Definición de Casos de uso ..... - 100 -
9.1.1 *Qué es un caso de uso* ..... - 100 -
9.1.2 Relaciones entre casos de uso ..... - 102 -
9.1.3 Como vamos a documentar nuestros casos de usos ..... - 104 -
9.2 Guía de uso – Manual del usuario ..... - 106 -
9.2.1 Guía de uso del administrador ..... - 107 -
9.2.2 Guía de uso del usuario ..... - 115 -

Capítulo 1

En este primer capítulo se plantean los objetivos y motivación de esta tesis, para terminar con una explicación general de la distribución de los temas estudiados en cada capítulo.

1.1 Introducción

La calidad de un sistema está altamente influenciada por la calidad de los procesos usados, para adquirirla, desarrollarla y mantenerla. Esta premisa indica que hay que poner el foco en los procesos tanto como en lo productos. Para certificar madurez de procesos, hay que evidenciar que uno aplica un cierto proceso y para esto se deben registrar las distintas actividades de tal proceso de desarrollo.

En particular, CMMI (Capability Maturity Model Integration)[28] define en sus distintos niveles, áreas, y una de ellas es el aseguramiento de la calidad de procesos y productos (PPQA del inglés Process and Product Quality Assurance), cuyo propósito es garantizar que los procesos definidos están siendo seguidos en la organización, así como poder detectar incumplimientos en la forma de trabajar definida.

Certificar CMMI nivel 2 de acuerdo al modelo CMMI-SW (Capability Maturity Model Integration for Software)[31] consiste, entre otras cosas, para el área PPQA, en generar evidencia del control realizado sobre la forma de trabajar de una organización. Esto es, generar evidencia que muestre que el área de aseguramiento de calidad controla que la organización realmente aplica el proceso que dice aplicar y que cuando no lo hace, registra tales fallas como no conformidades y pone en funcionamiento los mecanismos que sean necesarios para corregirlas.
Que una organización haya alcanzado nivel 2 de CMMI implica, en síntesis, que aplica procesos básicos de gestión de los proyectos (costo, calendario, funcionalidad). Estos procesos hacen que se puedan repetir éxitos en proyectos de similares características.

Como veremos con mayor detalle en el siguiente capítulo, el nivel 2 requiere que se consideren los siguientes aspectos:

- Gestión de requisitos
- Plan de Proyecto
- Monitoreo y control del proyecto
- Gestión de acuerdos con proveedores
- Medida y análisis
- Medidas de calidad en el proceso y producto
- Gestión de la configuración

Una organización que se encuentra en este nivel planea sus procesos, los ejecuta, los mide y los controla.

En particular el área de PPQA(Process and Product Quality Assurance) requiere:

- Evaluar objetivamente los procesos, productos y servicios realizados contra los procedimientos, estándares y descripción de procesos aplicables.
- Identificar y registrar las no conformidades.
- Proveer información de las actividades de QA
- Asegurar que las no conformidades son manejadas.

1.2 Objetivo

El objetivo de este trabajo es construir una herramienta para ayudar a certificar la madurez de procesos que requiere CMMI nivel 2, ayudando en la tarea diaria a la persona que realiza QA. El foco de la misma estará en la generación de evidencia de la aplicación de control de procesos a través del seguimiento de las auditorías y del registro y tratamiento de las no conformidades. Requerimientos esenciales para el área de PPQA.

El producto propuesto en esta tesis, es una herramienta que podrá utilizarse en el área de aseguramiento de la calidad (QA) de una fábrica de sistemas.

1.3 Motivación

Una de las premisas para certificar CMMI durante un SCAMPI (Standard CMMI Appraisal Method for Process Improvement) es la disponibilidad para el equipo de certificadores de evidencia objetiva de la implementación de las práctica requeridas. En aquellas organizaciones donde el equipo de QA no trabaja de forma ordenada y no sigue un proceso bien definido, la evidencia puede quedar distribuida por toda la empresa con diferentes formatos y en distintos soportes dificultando su recolección.
Para mitigar este problema se puede utilizar un sistema como el propuesto, para dejar evidencia del proceso de desarrollo en un lugar centralizado.

Como parte de su trabajo el certificador (Leader Appraiser) recoge evidencia y evalúa la coherencia entre la metodología escrita, la evidencia y la información que consiguió en las diferentes entrevistas que le hace a los empleados. El software propuesto unificará criterios y facilitará que un proyecto sea auditable por profesionales externos en cualquier momento, proveyendo interfaces amigables para el certificador, haciendo más fácil su tarea.

1.4 Organización del documento

La tesis se dividirá en 9 capítulos. El primer capítulo, éste que estamos transitando, será de introducción.
En el capítulo 2 veremos las ideas y conceptos principales detrás de CMMI-SW (Capability Maturity Model Integration for Software)[31].

El capítulo 3 hablará de la calidad del software, el aseguramiento de la calidad y del rol de ésta dentro del proceso de desarrollo de software, para finalizar con una breve descripción del área de proceso PPQA(Process and Product Quality Assurance) dentro de la categoría de soporte de CMMI, área que define los lineamientos de SQA dentro de este modelo. Este será la principal fuente de requerimientos de nuestro sistema.

En el capítulo 4, veremos las características generales de los métodos de certificación para CMMI. SCAMPI es la implementación de estos métodos realizada por el SEI para terminar el capítulo hablando de la persona que guía la evaluación, denominada Leader Appraiser.

Usando estos primeros capítulos de introducción al tema, en el capítulo 5 presentaremos el sistema objeto de ésta tesis, comenzado por describir el ambiente de trabajo en el que se usará el software, los usuarios y sus interacciones con el mismo. Luego describiremos el objetivo y la funcionalidad del sistema, hablando de los casos de uso que implementamos y de la manera que utilizamos para documentarlos.
Finalizando éste capítulo con la descripción de la arquitectura, las herramientas tecnológicas que usamos en el desarrollo de la herramienta y una breve descripción del diseño y de los detalles de implementación mas importantes.

En el capítulo 6 , mostraremos como caso de estudio, la aplicación de ésta herramienta en un escenario real.
En el capítulo 7, expondremos las conclusiones generales.
En el capítulo 8 veremos los trabajos futuros y aquellas mejoras que se pueden aplicar al sistema, que por ser limitado el plazo de construcción no alcanzamos a incorporar.

En el capítulo 9, como apéndice de esta tesis, incluimos también una guía de uso para el usuario de la herramienta.

Capítulo 2 – Ideas generales de CMMI-SW

2.1 Introducción

El Capability Maturity Model Integration (en adelante CMMI), se compone de un conjunto de modelos, métodos de evaluación y cursos de formación para diseñar procesos efectivos (tiempo y costo), en distintos dominios (desarrollo de productos y servicios, adquisiciones y mantenimiento), dentro del ámbito de una organización.

Este modelo fue desarrollado por el Instituto de Ingeniería del Software de la Universidad Carnegie Mellon [29] (Software Engineering Institute, en adelante SEI), y publicado en su primera versión en enero de 2002.

2.2 Procesos

Un proceso es un conjunto de prácticas que se ejecutan con un propósito determinado, las cuales transforman entradas en salidas que son de valor para el cliente. El proceso puede incluir herramientas, métodos, materiales y/o personas [30].
Proceso es uno de los 3 puntos de apalancamiento de la mejora del desempeño de una organización. Para mejorar el desempeño, se pueden cambiar los procesos, las personas, la tecnología o una combinación de ellos.

2.3 Modelo de procesos

Un modelo de procesos es un conjunto estructurado de elementos que describen características de procesos efectivos y de calidad. Un modelo indica "Qué hacer", no "Cómo hacer", ni "Quién lo hace". Un modelo proporciona:

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

Los procesos incluidos en el "modelo de procesos" son aquellos que la experiencia ha demostrado que son efectivos [30].

2.4 CMM

CMM (Capability Maturity Model)[31] es un modelo desarrollado por el Instituto de Ingeniería de Software – SEI de la Universidad Carnegie Mellon, basado en las mejores prácticas actuales de desarrollo de software. CMM es un marco para la mejora de procesos de software y para la evaluación de la madurez de las capacidades.

El modelo CMM aplica conceptos de gerencia de procesos, mejora de calidad al desarrollo y mantenimiento de software, y describe los estados a través de los cuales las organizaciones de software evolucionan a medida que definen, implementan, miden, controlan y mejoran sus procesos de software.

El modelo CMM puede ser abordado a través de dos representaciones(2.10), la escalonada y la continua, de las cuales hablaremos más adelante. Dentro de la representación por estados o escalonada, el modelo CMM aplica a la Organización niveles de madurez mediante los cuales se mide el avance de los procesos de la misma. El modelo define 5 niveles de madurez de una organización, como podemos ver en la siguiente tabla:

Nivel de madurez	Estados
1	Inicial
2	Administrado
3	Definido
4	Administrado cuantitativamente
5	Optimizado

Nivel de madurez según la representación por estados

Tabla 1

El nivel 1 (inicial) le corresponde a aquellas organizaciones donde los procesos de desarrollo son informales y ad hoc.

El siguiente nivel, el 2 (repetible o administrado), se aplica a aquellas empresas donde las prácticas de gestión de proyectos están institucionalizadas. Donde se pueden repetir éxitos de proyectos pasados.

En el nivel 3 (definido), una organización tiene las prácticas técnicas integradas con el nivel de gestión y están institucionalizadas.

En el nivel 4 (controlado o administrado cuantitativamente) los productos y procesos de la organización están controlados cuantitativamente.

Finalmente en el nivel 5 (optimizado) las mejoras de los procesos están institucionalizadas en la organización.

2.5 Madurez vs. Inmadurez

En esta sección a través de la siguiente tabla mostraremos la comparación entre una organización madura y una inmadura:

Organizaciones con procesos inmaduros	**Organizaciones con procesos maduros**
Procesos improvisados por los gerentes y desarrolladores	Procesos documentados
Cada uno posee sus propios procesos	Procesos seguidos consistentemente
Procesos comprometidos en orden a cumplir los costos y las fechas acordadas	El rendimiento de los procesos es medido, seguido y entendido
Calidad difícil de predecir	La calidad es predecible porque los procesos están bajo control
Los procesos "viven" mientras viven los desarrolladores	Los procesos "viven" por sí solos y son mejorados continuamente
Las nuevas tecnologías corren riesgos de caer en desuso.	Las nuevas tecnologías son incorporadas de una manera disciplinada.

Comparación entre Madurez e Inmadurez
Tabla 2

2.6 CMMI

CMMI(Capability Maturity Model Integration) es un conjunto de herramientas que ayudan a una organización a mejorar sus procesos de desarrollo de productos y servicios, adquisiciones y mantenimiento. El modelo CMMI ha evolucionado el concepto del Modelo CMM establecido para la maduración de la capacidad de Software, a un nivel que permite el crecimiento continuo y la expansión del concepto CMM para múltiples disciplinas, tales como SW-CMM[32], EIA/IS 731 [33], IPDCMM [34] y otros modelos de mejora de procesos.

La madurez de procesos es un indicador del potencial de crecimiento en capacidad de los procesos. La capacidad es el rango esperado a ser alcanzado por los procesos, y el rendimiento es el resultado real alcanzado por el proceso. Entonces podemos decir que el rendimiento de procesos pasados es un indicador de la capacidad de los procesos y que la capacidad de procesos es una predicción del rendimiento futuro de los procesos.

De la misma manera que el modelo CMM, el modelo CMMI tiene una representación(2.10) por estados de la cual hablaremos más adelante, que enfoca o mide la mejora de los procesos usando niveles de madurez.

2.7 Evolución de madurez de los procesos CMMI

Las Organizaciones en el camino hacia lograr la madurez de los procesos pasan por 5 niveles de maduración:

- **Nivel 1: Inicial** Procesos impredecibles, pobremente controlados y reactivos. El proceso de software se caracteriza como ad hoc y ocasionalmente caótico. Pocas actividades están definidas y el éxito de los proyectos depende del esfuerzo individual. Hay carencia de procedimientos formales, estimaciones de costos, planes del proyecto, mecanismos de administración para asegurar que los procedimientos se siguen.

- **Nivel 2: Administrado**. Procesos caracterizados en proyectos y acciones reactivas con frecuencia. Son establecidas las actividades básicas para la administración de proyectos de software para el seguimiento de costos, programación y funcionalidad. El éxito está en repetir prácticas que hicieron posible el éxito de proyectos anteriores, por lo tanto hay fortalezas cuando se

desarrollan procesos similares, y gran riesgo cuando se enfrentan nuevos desafíos.

Las áreas de proceso de nivel 2 son:

- Administración de Requerimientos (RM)
- Planeamiento de Proyectos (PP)
- Seguimiento y Control de Proyectos (PMC)
- Administración de acuerdos con Proveedores (SAM)
- Aseguramiento de la Calidad de Productos y Procesos (PPQA)
- Administración de la Configuración (CM)
- Mediciones y Análisis (MA)

Nivel 3: Definido. Procesos caracterizados en la organización, y con acciones proactivas.

Las actividades del proceso de desarrollo de software para la administración e ingeniería están documentadas, estandarizadas e integradas en un proceso de software estándar para la organización.

Las áreas de proceso de nivel 3 son:

- Foco en los Procesos Organizacionales (OPF)
- Definición de Procesos Organizacionales (OPD)
- Entrenamiento Organizacional (OT)
- Desarrollo de Requerimientos (RD)
- Solución Técnica (TS)
- Integración de Productos (PI)
- Verificación (VER)
- Validación (VAL)
- Administración del Proyecto Integrado (IPM)
- Administración de Riesgos (RiskM)
- Equipo Integrado (IT)
- Ambiente Organizacional (OE)
- Análisis de Decisión y Resolución (DAR)

Nivel 4: Administrado cuantitativamente. Los procesos son medidos y controlados. Se registran medidas detalladas de las actividades del Proceso y calidad del Producto. El

proceso de desarrollo de software y el producto son entendidos cuantitativamente y controlados.

Las áreas de proceso de nivel 4 son:

- Administración Cuantitativa de Proyectos (QPM)
- Performance de los Procesos Organizacionales (OPP)

Nivel 5: Optimizado. Enfoque continuo en la mejora de procesos.

Existe una mejora continua de las actividades, las que se logran a través de un feedback con estas áreas de procesos y también a partir de ideas innovadoras y tecnología. La recolección de datos es automatizada y usada para identificar elementos más débiles del proceso. Se hace un análisis riguroso de causas y prevención de defectos.

Las áreas de proceso de nivel 5 son:

- Análisis de Causas y Resolución (CAR)
- Innovación Organizacional (OI)

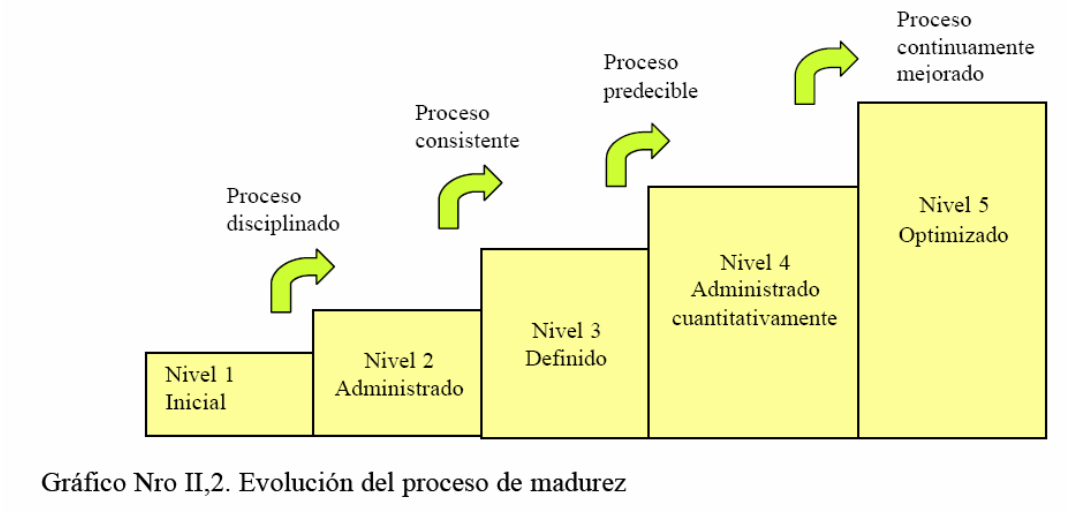


Gráfico Nro II,2. Evolución del proceso de madurez

Figura 1

2.8 Antecedentes

El modelo CMMI tiene los siguientes modelos como antecedentes de referencia:

- CMM-SW [32]: CMM para software
- P-CMM [35]: People CMM
- SA-CMM [36]: Software Acquisition CMM.
- SSE-CMM [37]: Security Systems Engineering CMM.
- SE-CMM [38]: Systems Engineering CMM.

- IPD-CMM [34]: Integrated Product Development CMM.

CMMI se desarrolló para facilitar y simplificar la adopción de varios modelos de forma simultánea, y su contenido integra y da relevo a la evolución de sus predecesores, como se muestra en la siguiente figura:

- CMM-SW (CMM for Software)
- SE-CMM (Systems Engineering Capability Maturity Model)
- IPD-CMM (Integrated Product Development)

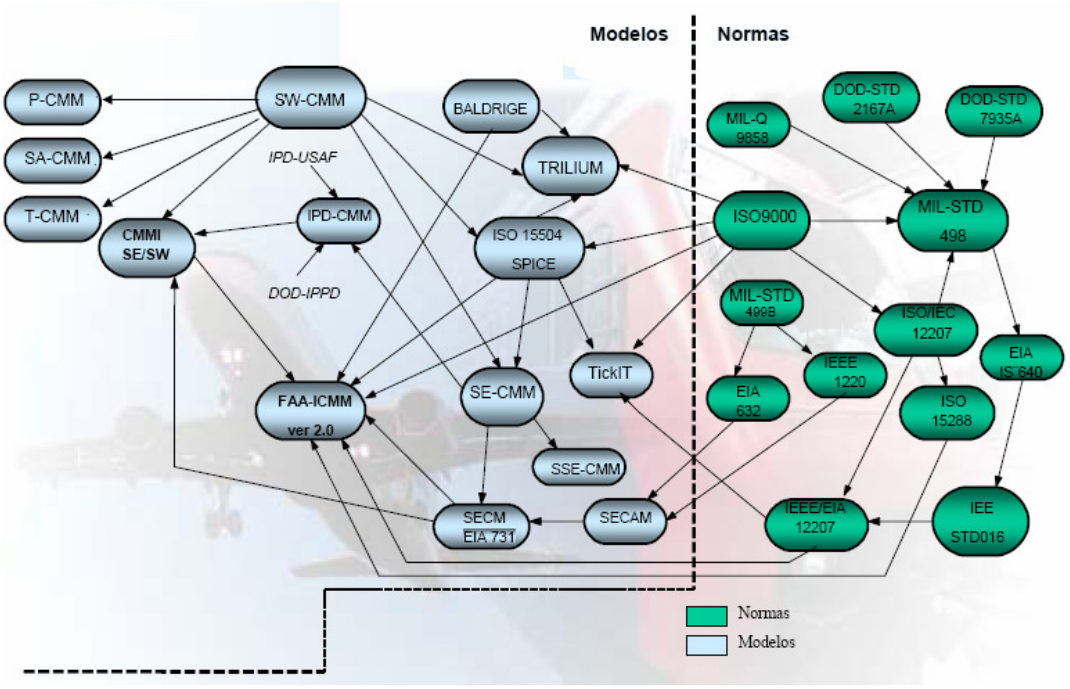


Figura 2: Modelos y normas relacionados con CMMI

2.9 Descripción del modelo

CMMI se compone de un conjunto de modelos, métodos de evaluación y cursos de formación para diseñar procesos efectivos (tiempo y costo), en distintos dominios (desarrollo de productos y servicios, adquisiciones y mantenimiento), dentro del ámbito de una organización.

El modelo CMMI declara el conjunto de prácticas que se deben llevar a cabo sin detallar la forma de implantarlas en cada empresa, y se estructura en un conjunto de

áreas de proceso, entendidas como el conjunto de prácticas relacionadas que son ejecutadas de forma conjunta para conseguir un conjunto de objetivos. Cada área de proceso se descompone en objetivos:

- Objetivos genéricos: asociados a un nivel de capacidad establecen lo que una organización debe alcanzar en ese nivel. El logro de cada uno de esos objetivos en un área de proceso significa mejorar el control en la ejecución del área de proceso
- Objetivos específicos: se aplican a una única área de proceso y localizan las particularidades que describen que se debe implementar para satisfacer el propósito del área de proceso.

Cada uno de estos objetivos se descomponen a su vez en prácticas:

- Prácticas genéricas: se aplican a cualquier área de proceso porque puede mejorar el funcionamiento y el control de cualquier proceso.
- Prácticas específicas: actividades que se consideran importantes en la realización del objetivo específico al cual está asociado. Las prácticas específicas describen las actividades esperadas para lograr la meta específica de un área de proceso

A partir de la versión 1.2 del modelo, se han creado las constelaciones de CMMI, como un conjunto de productos de trabajo que son relevantes para mejorar en un área de interés particular. En la actualidad existen tres constelaciones:

- Desarrollo (CMMI for Development, CMMI-DEV [39]): de aplicación para la creación de sistemas intensivos en software.
- Adquisición (CMMI for Acquisition, CMMI-ACQ [40]): para la creación de productos y servicios bajo la modalidad de subcontratación contractual.
- Servicios (CMMI for Services, CMMI-SVC [41]): para productos intangibles y que no se pueden almacenar (formación, mantenimiento, consultoría, soporte, etc.)

Las áreas de proceso que componen el modelo CMMI en su versión 1.2 son las siguientes:

- **CAR**: Causal Analysis and Resolution
- **CM**: Configuration Management
- **DAR**: Decision Analysis and Resolution

- **IPM**: Integrated Project Management +IPPD
- **MA**: Measurement and Analysis
- **OID**: Organizational Innovation and Deployment
- **OPD**: Organizational Process Definition +IPPD
- **OPF**: Organizacional Process Focus
- **OPP**: Organizational Process Performance
- **OT**: Organizational Training
- **PI**: Product Integration
- **PMC**: Project Monitoring and Control
- **PP**: Project Planning
- **PPQA**: Process and Product Quality Assurance
- **QPM**: Quantitative Project Management
- **RD**: Requirements Development
- **REQM**: Requirements Management
- **RSKM**: Risk Management
- **SAM**: Supplier Agreement Management
- **TS**: Technical Solution
- **VAL**: Validation
- **VER**: Verification

2.10 Representaciones

El modelo CMMI puede ser abordado en sus dos representaciones, la escalonada y la continua. La representación escalonada (staged) hace especial énfasis en el grado de madurez de los procesos, de forma que cada área de proceso se asocia a uno de los 5 niveles de madurez, que sirven como punto de referencia para conocer el grado de madurez total que posee una organización.

Estos cinco niveles reflejan el hecho de que CMMI es un modelo para la mejora de la capacidad de las organizaciones de software. Las prioridades en el modelo no están dirigidas hacia proyectos individuales sino a procesos que aporten valor a la organización en su conjunto.

Una organización alcanza un nivel de madurez determinado cuando ha puesto en práctica todas y cada una de las áreas de proceso aplicables a ese nivel y a los niveles

inferiores. Los diversos niveles de madurez de la representación escalonada quedan definidos en la siguiente tabla:

Nivel	Descripción
1 Inicial	El proceso de desarrollo de software está caracterizado como 'ad hoc', y en ocasiones puede ser incomprensible. Algunos procesos están definidos y el éxito depende de los esfuerzos a nivel de individuo.
2 Gestionado	Los procesos de gestión de proyectos están definidos de una manera básica para realizar el seguimiento de los costos, fechas y funcionalidad. El rigor en la definición de los procesos es el justo para poder repetir éxitos previos en proyectos de similares características.
3 Definido	El proceso de desarrollo de software para las actividades de gestión e ingeniería está documentado, estandarizado e integrado en el proceso estándar dentro de la organización. Todos los proyectos utilizan una versión estándar del proceso de desarrollo de software aprobado por la organización y adaptado a las necesidades del proyecto para desarrollo y mantenimiento del software.
4 Cuantitativamente Gestionado	Se recogen de forma detallada medidas de los procesos software y la calidad de los productos. Los procesos y productos software son entendidos cuantitativamente y controlados.
5 Optimizado	La mejora continua de procesos se basa en los resultados cuantitativos de la aplicación de innovaciones y tecnologías en los procesos ya establecidos.

Tabla 3: Niveles de la representación escalonada

La representación continua (continuous) hace hincapié en la capacidad de ciertas áreas para realizar adecuadamente sus actividades. En la representación continua, los niveles de madurez no existen como tales y, en cambio, los niveles de capacidad se designan para cada área de proceso, proporcionando un orden recomendado para acercarse a la mejora dentro de cada área de proceso.

Una representación continua favorece la flexibilidad en el orden hacia el cual se dirigen las mejoras, por lo que suele ser una forma frecuente de abordar un programa de mejora de procesos en pequeñas organizaciones.

En la representación continua las áreas de proceso se pueden agrupar en cuatro categorías generales: Gestión de Proyectos, Gestión de Procesos, Ingeniería y Soporte.

Resumiendo, en la representación escalonada se define, para cada nivel de madurez un conjunto de áreas de proceso. En la representación continua para cada una de las categorías de áreas de proceso se define el área a mejorar para alcanzar cierto nivel.

Nivel	Ingeniería	Gestión de Proyectos	Gestión de Procesos	Soporte
5			Innovación y despliegue organizativo(OID).	Innovación y despliegue organizativo(OID). Análisis causal(CAR).
4		Gestión de proyectos cuantitativa(QPM).	Rendimiento de procesos organizativos(OPP).	
3	Desarrollo de requerimientos(RD). Solución técnica(TS). Integración de producto(PI). Verificación(VER) Validación(VAL)	Gestión de proyecto integrada(IPM). Gestión de riesgos(RSKM).	Foco en proceso organizativo(OPF). Definición de proceso organizativo(OPD). Entrenamiento organizativo(OT).	Análisis de decisiones y soluciones(DAR).
2	Gestión de Requierimientos(RM)	Planificación de Proyecto(PP). Seguimiento y control de Proyecto(PMC). Gestión de acuerdos con proveedores(SAM).		Gestión de la configuración(CM). Aseguramiento de la calidad de producto y proceso(PPQA). Medición y análisis(MA).

Tabla 4: Vista de la representación del modelo CMMI por nivel y categoría

Leyendo la tabla 4 de forma horizontal, tenemos la representación escalonada de CMMI, para cada nivel las áreas de proceso necesarias. Si miramos la tabla de manera vertical, tenemos la representación continua, cada categoría cuales son las áreas de proceso necesarias para un determinado nivel.

2.11 Por qué se aplica CMMI

Se aplica CMMI porque:

- Es un modelo que aplica estándares de calidad
- Provee un enfoque más efectivo e integrado a Ingeniería de Sistemas y de Software.
- Construye procesos desde un inicio, o sobre inversión previa usada con SW CMM.

- Provee un enlace o relación mas explícita entre la Gestión y la Ingeniería con los objetivos del negocio.
- Provee mayor visibilidad del ciclo de vida del producto; y las actividades de ingeniería ayudan a asegurar que los productos y servicios satisfacen las expectativas de los clientes.
- Incorpora lecciones aprendidas de otras áreas, de mejores prácticas e implanta prácticas de alta madurez más robustas.
- Incluye funciones organizacionales que son críticas para los productos y servicios.
- Soporta integración futura con otros modelos CMMI de disciplinas específicas.
- Muestra el camino a seguir en la gestión de proyectos, integrando de una manera ordenada los procesos y los productos.

Capítulo 3 - Aseguramiento de la calidad del software

3.1 Introducción

La calidad es el conjunto de propiedades inherentes a una entidad, que permiten juzgar su valor. Está cuantificada por el valor que se le da al conjunto de propiedades seleccionadas. De esta manera la calidad es subjetiva y, como dice James Bach [11], es circunstancial. Es subjetiva porque depende de los atributos elegidos para medirla y es circunstancial porque el conjunto de atributos elegidos puede variar en situaciones diferentes.

Cuando aplicamos el concepto de calidad al software, éste deja de ser subjetivo porque se determinan cuales son los atributos de calidad del software. Pero no deja de ser accidental ya que en ciertas situaciones, un determinado conjunto de características de calidad puede ser más importante que en ciertas otras.

Resumiendo, la calidad del *software* es medible y varía de un sistema a otro o de un programa a otro.

3.2 Calidad del software

Hablamos todo el tiempo de problemas relacionados con la calidad del software pero no tenemos una definición precisa de lo que ésta significa. Sin una definición clara, concisa y medible de lo que es la calidad del software, no podemos tomar buenas decisiones de negocio respecto del uso de los recursos, ni en que áreas mejorar la calidad, ni que herramientas y técnicas utilizar para mejorar la calidad.
Hay diferentes puntos de vista para definir calidad de software. Desde el punto de vista del cumplimiento de los requerimientos Roger Pressman define la calidad de software como:

> *"El cumplimiento de los requerimientos funcionales y de performance explícitamente definidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas esperadas del desarrollo de software profesional." [1]*

Desde el punto de vista del cliente o usuario Watts Humphrey dice:

"El foco principal de cualquier definición de calidad de software debería ser las necesidades del cliente. Crosby [6] al igual que Pressman [1] define la calidad como conformidad con los requerimientos. Mientras uno puede discutir la diferencia entre requerimientos, necesidades y deseos, la definición de calidad debe considerar la perspectiva de los usuarios. Entonces las preguntas claves son ¿Quiénes son los usuarios?, ¿Qué es importante para ellos? Y ¿Cómo sus prioridades se relacionan con la manera en que se construye, empaqueta y se da soporte al producto?[2]"

Al Davis define calidad del software como:

"La calidad no se trata de tener cero defectos o una mejora medible de la proporción de defectos, no se trata de tener los requerimientos documentados. No es mas ni menos que satisfacer las necesidades del cliente (por mas que las necesidades estén o no correctamente documentadas)[5]"

Finalmente, desde estas dos perspectivas el glosario de la IEEE para la ingeniería de software define la calidad del software como:

"*El grado con el cual un sistema, componente o proceso cumple con los requerimientos y con las necesidades y expectativas del usuario. [4]"*

Mas allá de cómo definamos la calidad del software, para que la definición tenga sentido esta debe ser medible. Para poder controlar la calidad del software es necesario, ante todo, definir los parámetros, indicadores o criterios de medición, ya que, como bien plantea Tom De Marco, "no se puede controlar lo que no se puede medir".

Para poder identificar los costos y beneficios del software se definieron los atributos de calidad. La intención es separar el software en atributos que puedan ser medidos o cuantificados (en términos de costo beneficio). Ejemplos de estos atributos son confiabilidad, adaptabilidad, usabilidad y funcionalidad.

Para clasificar los atributos de calidad del software se definieron varios modelos, uno ellos fue el modelo FURPS+. Este modelo fue desarrollado por Robert Grady y Deborah Caswell de Hewlett Packard

Bajo el acrónimo FURPS+, por sus siglas en inglés, se definen las siguientes características:

Sigla	Tipo de Requerimiento		Descripción
F	Funtional	***Funcional***	Características, capacidades y algunos aspectos de seguridad
U	Usability	***Facilidad de Uso***	Factores Humanos (interacción), ayuda, documentación
R	Reliability	***Fiabilidad***	Frecuencia de fallos, capacidad de recuperación de un fallo y grado de previsión
P	Performance	***Rendimiento***	Tiempos de respuesta, productividad, precisión, disponibilidad, uso de los recursos
S	Supportability	***Soporte***	Adaptabilidad, facilidad de mantenimiento, internacionalización, facilidad de configuración.
+	Plus	***Implementación***	Limitación de recursos, lenguajes y herramientas, hardware
		Interfaz	Restricciones impuestas para la interacción con sistemas externos (no es GUI)
		Operaciones	Gestión del sistema, pautas administrativas, puesta en marcha
		Empaquetamiento	Forma de distribución
		Legales	Licencia, derechos de autor, etc.

Tabla 5: Fuente: Adaptado de Craig Larman: "UML y Patrones" 2Ed. 2003.

El más(+) en el acrónimo FURPS+ nos permite especificar restricciones, incluyendo restricciones de diseño, implementación e interfaces.

La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

Cuando no se cumplen los estándares o procesos de la organización o del proyecto se dice que estamos frente a una no conformidad. Lo esperable es la ausencia de no conformidades (conformidad)

El costo de conformidad (calidad) no es despreciable pero es menor que el costo de su alternativa (Costo de no conformidad). Crosby describe el costo de la no-conformidad como aquel costo en el que se incurre porque el producto o servicio no se desarrolló de forma apropiada la primera vez.

Figura 3: Ecuación para obtener el costo de la calidad

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 o requerimientos, 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.

En el siguiente cuadro Pressman muestra el costo relativo de corregir un error en las distintas etapas del ciclo de vida.

Fase	Requisitos	Diseño	Código	Prueba (desarrollo)	Prueba (sistema)	Explotación, producción
Multiplicador Coste	1	3-6	10	15-40	30-70	40-1000

Tabla 6: Costo de un error en las diferentes etapas del proceso.

Como se puede observar en la tabla 5, el costo relativo de corregir un error en producción es cientos de veces más que en la etapa de requisitos. El 50% del costo de

corrección de defectos en las últimas fases se debe a defectos introducidos en las primeras.

Entonces uno de los propósitos que guían el aseguramiento de la calidad es que es mucho menos costoso corregir los problemas en sus fases iniciales que esperar hasta que un problema se manifieste a través de las quejas del usuario.

Los principios básicos del concepto de calidad del software son:

- No alcanza en pensar en calidad a la hora hacer las revisiones y pruebas sino que debe ser una preocupación durante todo el ciclo de vida del software.
- Sólo se alcanza con la contribución de todas las personas involucradas.
- La calidad debe ser planificada y gestionada con eficacia.
- Dirigir esfuerzos a prevención de defectos.
- Reforzar los sistemas de detección y eliminación de defectos durante las primeras fases.
- La calidad es un parámetro importante del proyecto al mismo nivel que los plazos de entrega, costo y productividad.
- Es esencial la participación de la dirección, que ha de propiciar la calidad.

Hasta aquí, lo que define el modelo FURPS+ son los atributos de calidad de los productos de software, pero también los procesos que se siguen en su desarrollo cuentan con atributos de calidad como por ejemplo que estén bien definidos, que estén documentados, que sean prácticados y medidos. La calidad depende de los procesos, debe estar orientada al cliente y es de mejora continua.

3.3 Aseguramiento de la calidad del software

3.3.1 Surgimiento de SQA (Software Quality Assurance)

En los años 50, el software comenzó a encontrar su camino dentro de los sistemas del DoD (del inglés Deparment of Defense of USA). Usualmente estos proyectos estaban muy alejados de la planificación, se pasaban del presupuesto y tenían muchos problemas técnicos. Frecuentemente no funcionaban como se esperaba y muchos proyectos eran cancelados antes de ser entregados. Durante este periodo los contratistas

para el desarrollo de software a menudo hacían estimaciones muy optimistas sobre el estado del desarrollo del software. El DoD normalmente no era notificado de los problemas en la planificación, en la gestión del presupuesto y de problemas técnicos hasta muy avanzado el proyecto, cuando ya no eran capaces de entender los problemas ni de evaluar el impacto de éstos.
Para intentar resolver este problema se estableció la Verificación y Validación Independientes(IV&V del inglés Independent Verification and Validation), un proceso de ingeniería que empleaba metodologías rigurosas para evaluar la correctitud y calidad del software a lo largo de su ciclo de vida.

El primer software en usar IV&V fue el programa del misil atlas a finales de los años 50. Desde el proyecto atlas se ha recolectado mucha información que indica que los proyectos con IV&V se realizan o ejecutan mucho mejor que los proyectos sin IV&V. Con el tiempo el rol del IV&V se convirtió critico.
La actividad que llamamos SQA evoluciona directamente de la Verificación y Validación Independientes(IV&V), muchas de las tareas que asociamos con SQA son originarias de IV&V.

Luego durante los años 70 la actividad de desarrollo de software comenzó a expandirse y las compañías de desarrollo de software fueron experimentando los mismos pobres resultados que las agencias gubernamentales(DoD, NASA etc.) en las décadas tempranas. Las compañías tenían dificultad para entregar el software dentro de los plazos, presupuesto y calidad planificados. Varios proyectos desarrollados entre 1980 y 1990 fueron desastrosos, muchos excedían ampliamente el presupuesto y la planificación o entregaban software de baja calidad que no se podía usar.
Durante los 80 esta experiencia se convirtió en lo que conocemos como crisis del software, el tiempo consumido en el mantenimiento excedía el tiempo insumido en la construcción de nuevos productos de software.

Luego de la crisis del Software en los años 80, SQA evoluciono hacia una herramienta que las compañías de desarrollo de software utilizaban para identificar de forma temprana los problemas de calidad en el proceso de desarrollo. Mientras SQA era visto como un pequeño paso dentro del proceso del desarrollo del software, muchos jefes de

proyectos vieron beneficios cuantificables a partir de integrar SQA dentro del proceso de desarrollo de software.

En los 90 varias compañías de software ya tenían funciones de SQA dentro de sus organizaciones.

3.3.2 Definición de SQA (Software Quality Assurance)

Al igual que ocurrió con la definición de calidad hay varios puntos de vista desde donde se puede definir el aseguramiento de la calidad del software.
Desde el punto de vista de la evidencia, la IEEE define el aseguramiento de la calidad como

"Una guía planificada y sistemática de todas las acciones necesarias para proveer la evidencia adecuada de que un producto cumple los requerimientos técnicos establecidos.
Un conjunto de actividades diseñadas para evaluar el proceso por el cual un producto es desarrollado o construido." [4]

Daniel Galin define SQA como

"Un conjunto, sistemático y planificado, de acciones necesarias para proveer la evidencia adecuada de que el proceso de desarrollo o mantenimiento de un sistema de software cumple los requerimientos técnicos funcionales tan bien como los requerimientos gerenciales para cumplir la planificación y operar dentro del presupuesto confinado" [7].

Desde el punto de vista de la visibilidad, el SEI define SQA como

"El aseguramiento de la calidad del software provee claro control del proceso que está siendo usado por el proyecto y del producto que se está construyendo."[8]

Desde el punto de vista del aseguramiento, Don Reifer define SQA como

"El aseguramiento de la calidad del software es el sistema de métodos y procedimientos usados para asegurar que el producto de software alcanza sus requerimientos. El sistema involucra la planificación, estimación y monitoreo de las actividades de desarrollo realizadas por otros."[9]

Desde el punto de vista de la capacidad de uso Schulmeyer y McManus definen SQA como

"Las actividades sistemáticas que proveen evidencia de la capacidad o disponibilidad de uso del producto de software total."[10]

Para certificar madurez de procesos, hay que evidenciar que uno aplica un cierto proceso y para esto se deben registrar las distintas actividades de tal proceso de desarrollo, como éste es el objetivo que persigue el software a desarrollar como parte de esta tesis, elegiremos la definición que da la IEEE desde el punto de vista de la generación de evidencia adecuada que muestre que se cumple con el proceso que se dice seguir y con los requerimientos establecidos.

3.3.3 SQA no es lo mismo que SQC (Software Quality Control)

Generalmente cuando le preguntamos a un profesional de sistemas que es lo que entiende por aseguramiento de la calidad del software, inmediatamente comienza a hablar de testing, algunos de ellos incluyen a la validación y verificación y luego empiezan a hablar de revisiones, las cuales son sólo extensiones del testing. Es decir, a menudo hay una confusión entre SQA y el testing (el cual actualmente forma parte del área de control de calidad del software SQC).

Haciendo sólo testing y revisiones no aseguramos la calidad de los productos, sino aseguramos el cumplimiento de especificaciones tanto funcionales como técnicas. En el desarrollo de software la diferencia entre SQC y SQA no esta clara y estos términos a menudo se confunden, SQA se encarga de controlar el cumplimiento del proceso, mientras que SQC son aquellas acciones del aseguramiento de la calidad que proporcionan un medio para controlar y medir las características de un elemento, proceso o facilidad respecto a los requisitos establecidos.

La siguiente tabla expone sintéticamente las diferencias entre control de calidad y aseguramiento de la calidad.

Control de calidad	Aseguramiento de la calidad
Detecta problemas en los productos de trabajo.	Asegura la adherencia a los procesos, estándares y planes.
Verifica que los productos de trabajo cumplan con los estándares de calidad especificados en el plan de proyecto.	Evalúa que los procesos, planes y estándares utilizados en el proyecto cumplan con los estándares organizacionales.
Revisa el contenido del producto	Revisa procesos

Tabla 7:Control de calidad vs. Aseguramiento de la calidad

En conclusión, el rol del SQA es auditar que los distintos equipos de la organización, inclusive el de SQC siguen los procedimientos, estándares y procesos establecidos. El equipo de SQA debería establecer métricas para medir la efectividad del proceso. Como complemento el rol de SQC es tomar una actitud activa de verificación y validación del resultado o salida del proceso implementado.

3.3.4 Funciones generales del SQA

Describir los diferentes roles que puede jugar el equipo de SQA en una organización nos dará una visión clara de las funciones que puede llevar a cabo.

"Como policía del proceso": el trabajo del equipo de SQA es asegurar que el desarrollo sigue el proceso establecido. Entre sus funciones en este rol se encuentran:

- Auditar los productos del trabajo para identificar deficiencias.
- Determinar el cumplimiento del plan de desarrollo del proyecto y del proceso de desarrollo de software.
- Juzgar el proceso y no el producto.

“Como abogado del cliente”: el trabajo del equipo de SQA es representar al cliente. Entre sus funciones en este rol se encuentran:

- Identificar la funcionalidad que al cliente le gustaría encontrar.
- Ayudar a la organización a sensibilizarse con las necesidades del cliente.
- Actuar como un cliente de prueba para obtener una alta satisfacción del cliente.

“Como analista” el trabajo del equipo de SQA es recabar información. Entre sus funciones en este rol se encuentran:

- Juntar muchos datos sobre todos los aspectos del producto y del proceso.
- Con esta información ayudar a mejorar los procesos y los productos.

“Como proveedor de información” el trabajo del equipo de SQA es revisar qué es lo que esté hecho y decir cuáles objetivos técnicos realmente están cumplidos para que la gerencia pueda tomar mejores decisiones de negocios. Entre sus funciones en este rol se encuentran:

- Proveer información técnica objetiva para que la gerencia pueda usarla para tomar mejores decisiones.
- Proveer información apropiada de las clases de productos y de los riesgos asociados con estos.
- Concentrarse más en la reducción de los riesgos que en el cumplimiento del proceso.

“Como responsable de la elaboración del proceso” el trabajo del equipo de SQA es participar en la definición de los planes, procesos, estándares y procedimientos para asegurar que se ajustan a las necesidades del proyecto y que pueden ser usados para realizar las evaluaciones de QA y cumplir los requerimientos del proyecto y las políticas de la organización. Para cumplir este rol el aseguramiento de la calidad debería comenzar en las fases tempranas del proyecto.

Aquí conviene aclarar que no necesariamente las personas que definen la metodología a seguir pertenecen al equipo de QA. Definir la metodología puede llegar a ser o no una actividad del equipo de QA. Una estructura posible en el proceso de mejora del software puede ser contar con un SEPG (Software Engineering Process Group)

totalmente independiente del equipo de QA, encargado de definir la metodología mientras que el equipo de QA se limita a verificar que se cumpla dicha metodología.

3.3.5 Consideraciones

Para ser efectivo, el equipo que realiza SQA debe ser independiente de la organización de desarrollo. Aunque tener un grupo de auditoría independiente es difícil de aplicar en organizaciones chicas donde hay pequeños ambientes de desarrollos. Pero si la organización es madura y tiene una cultura orientada a la calidad, la función de SQA puede estar embebida en el proceso. Cuando el equipo de SQA esta embebido en el proceso, se deben resolver varios inconvenientes para garantizar la objetividad:

- Todo aquel que realice actividades de aseguramiento de la calidad debería estar entrenado en el aseguramiento de la calidad.
- Las actividades de aseguramiento de la calidad realizadas para un producto deberían ser separadas de aquellas involucradas en el desarrollo y mantenimiento del mismo.
- Debe estar disponible un canal de comunicación independiente en el nivel apropiado de la gerencia para poder escalar las no conformidades cuando sea necesario.

El equipo de SQA provee a la gerencia de información fehaciente, objetiva en el momento adecuado. La clave aquí esta en que el grupo de SQA provee a la gerencia de información técnica objetiva. La gerencia necesita ver a la gente de SQA como una fuente de información significativa que puede ayudarla a tomar decisiones difíciles. La Gerencia usa esta información para tomar decisiones de negocio apropiadas.

La objetividad en la evaluación de calidad de los procesos y productos es crítica para el éxito del proyecto. La objetividad se logra con independencia del equipo de SQA y sentido común o criterio.

Hay diferentes maneras de realizar evaluaciones objetivas, entre las que se incluyen:

- Auditorías formales realizadas por un área de SQA independiente de la organización.
- Revisiones de a pares que pueden se realizadas con distintos niveles de formalidad.
- Revisiones rigurosas en el lugar de desarrollo.
- Revisiones distribuidas y comentarios del producto.

Teniendo en cuenta estas consideraciones podemos decir que la tarea del equipo de SQA es un conjunto planificado de tareas, actividades y acciones ejecutadas independientemente de la organización que desarrolla software, que provee a la gerencia del proyecto información fehaciente en un momento preciso que puede ser usada para tomar decisiones de negocio apropiadas.

3.5 Aseguramiento de la calidad de procesos y productos en CMMI (PPQA).

3.5.1 Introducción

Como ya mencionamos anteriormente la calidad de un producto esta determinada en gran medida por la calidad del proceso utilizado para desarrollarlo y mantenerlo, por lo que se sigue trabajando en mejorar la forma en la que las organizaciones afrontan los proyectos de construcción de software.

Entre las áreas de proceso, en la categoría de soporte para el nivel 2 de CMMI se encuentra PPQA (Process and Product Quality Assurance) – Aseguramiento de la calidad de producto y proceso. Esta es el área de proceso principal para el aseguramiento de la calidad del software dentro de CMMI.

3.5.2 Metas y prácticas.

El propósito del Aseguramiento de la Calidad de Proceso y Producto es proporcionar a los diferentes equipos y a la gerencia una visión objetiva de los procesos y productos asociados. El objetivo fundamental de PPQA es garantizar que los procesos definidos están siendo respetados en la organización, así como poder detectar deficiencias en la forma de trabajar establecida.

Las metas de esta área de proceso son:

- Evaluar objetivamente la ejecución de los procesos, los elementos de trabajo y servicios en contraste a los procesos, estándares y procedimientos definidos.
- Identificar y documentar no conformidades. Recordemos que definimos una no conformidad como el incumplimiento de un estándar o proceso de la organización o del proyecto.
- Proporcionar información a las personas que están usando los procesos y a la gerencia del proyecto, de los resultados de las actividades del aseguramiento de la calidad.
- Asegurar de que las no conformidades son tratadas.

Las prácticas en éste área de proceso aseguran que los proceso establecidos son implementados mientras que las prácticas en el área de proceso de verificación aseguran que se cumplen los requerimientos especificados. En ciertas ocasiones estas dos áreas de proceso trabajan sobre el mismo producto pero desde dos perspectivas diferentes.

El siguiente es el listado de funcionalidades típicas para satisfacer este área de proceso:

- Completar las plantillas de chequeo del cumplimiento de los procesos definidos en la organización dentro del sistema de control de documentación.
- Generar la convocatoria de una auditoría de calidad (personal participante y objetivos) dentro de la agenda de eventos compartida.
- Dejar registro de la ejecución de la auditoría de calidad dentro del sistema de control de documentación (por ejemplo rellenando las plantillas de chequeo predefinidas).
- Registrar las no conformidades dentro del sistema de gestión de incidencias.
- Generación automática de informes de auditorías en base al rellenado de las plantillas de chequeo y notificación a las personas implicadas
- Posibilidad de generar un cuadro de mando que visualice las no conformidades encontradas en las auditorías de calidad de cada proyecto
- Registro del tiempo incurrido en aseguramiento de calidad, así como una estimación del ahorro de tiempo obtenido al encontrar no conformidades con

anticipo, de manera que se pueda analizar si el tiempo en QA es una buena inversión para la pequeña empresa

El cumplimiento de estas metas y prácticas son el camino para incrementar la madurez de la organización en el desarrollo de software conforme a CMMI-SW, la guía para este objetivo es el equipo de QA compuesto por personal experimentado, entrenado en los procesos y actividades de QA que asegurará un nivel de calidad estándar de productos y servicios para los clientes. El progreso en este camino puede ser medido a través de evaluaciones que certifiquen el grado de concordancia de los procesos y procedimientos con respecto a las metas y prácticas especificadas.

Capítulo 4 – Certificación CMMI-SW

4.1 Introducción

Luego de la implementación del modelo de CMMI en una organización se debe recoger evidencias para comprobar si la organización ha alcanzado el nivel de madurez deseado. Muchas organizaciones miden su progreso llevando a cabo una evaluación (Appraisal) y ganando una clasificación del nivel de madurez. Este tipo de evaluaciones son realizadas normalmente por las siguientes razones:

- Para determinar qué tan bien los procesos de la organización se comparan con las mejores prácticas CMMI y determinar que mejoras se pueden hacer.
- Para informar a los clientes externos y proveedores acerca de qué tan bien los procesos de la organización se comparan con las mejores prácticas CMMI.
- Para cumplir los requisitos contractuales de uno o más clientes.

La evaluación se enfoca en identificar oportunidades de mejora, y comparar los procesos de la organización con las mejores prácticas CMMI. Los equipos de evaluación usan métodos conformes a los requerimientos para certificar CMMI (ARC-Appraisal Requirements for CMMI) [12] para guiar su evaluación y reporte de conclusiones. Los resultados de la evaluación son usados para planear mejoras en la organización.

Hay tres clases de métodos de evaluación. Clase A, B, C. El Standard CMMI Appraisal Method for Process Improvement (SCAMPI) es un método de evaluación que cumple todos los requerimientos ARC.

4.2 Requerimientos para la certificación de CMMI

4.2.1 Introducción

Los requerimientos de certificación de CMMI (Appraisal Requirements for CMMI, ARC) definen las características esenciales de los métodos de certificación, consisten en un conjunto de criterios de diseño de alto nivel para desarrollar, definir y usar métodos de certificación basados en el modelo CMMI.

Los requerimientos para las evaluaciones CMMI sirven de guía para el desarrollo de métodos de certificación, definen tres clases de métodos de certificación, las clases reflejan maneras comunes de usar éstos métodos. Los métodos de clase A, son los más rigurosos, se centran en cómo se ha realizado la implementación y se examina el nivel de institucionalización de las prácticas desplegadas. Los de clase B, son métodos iniciales e incrementales, son métodos de autoevaluación de los procesos. Finalmente los de clase C, son métodos que sirven para tener una mirada rápida de los procesos.

Los equipos de evaluadores usan modelos de CMMI como base para identificar los puntos fuertes y débiles de los procesos examinados durante la evaluación.

Los resultados de la evaluación pueden ser usados para:

- Planificar una estrategia de mejora para la organización
- Generar mediciones del nivel de madurez o capacidad
- Guiar la toma de decisiones.
- Mitigar los riesgos para la adquisición, desarrollo y monitoreo de los productos.

*4.2.2 Principios y conceptos fundamentales de ARC (***Appraisal Requirements for CMMI***).*

Los principios de evaluación ARC son:

- Comenzar con un modelo de referencia de certificación.
- Usar un proceso de certificación formalizado.

- Involucrar a la alta gerencia como patrocinador de la evaluación.
- Centrar la certificación sobre los objetivos de negocio del patrocinador
- Observar estrictamente la confidencialidad y no las particularidades de los datos.
- Abordar la evaluación de forma colaborativa.
- Enfocar las actividades a seguir y basar las decisiones a tomar sobre los resultados de la certificación.

Los elementos fundamentales de ARC son:

- **El proceso de certificación:** una revisión o examen de uno o más procesos realizado por un equipo de profesionales entrenados usando un modelo de referencia de certificación como fundamento para determinar los puntos fuertes y las debilidades.
- **El Modelo de referencia de Certificación**: el modelo CMMI para el cual un equipo de certificación relaciona, entrelaza las actividades del proceso implementado.
- **La Unidad organizacional:** el área de una organización que es objeto de una certificación (también conocida como alcance de la evaluación o certificación). Una unidad organizacional implementa uno o más procesos que tienen un contexto coherente y operan dentro de un conjunto de objetivos de negocio coherentes. Una unidad organizacional es generalmente parte de una organización mayor, aunque para pequeñas organizaciones la unidad organizacional puede ser la organización entera. Ejemplos: – un proyecto o un conjunto de proyectos – una unidad dentro de una organización enfocada en una fase. – parte de la organización que se responsabiliza en el desarrollo de un producto.
- **Instanciación:** Las prácticas de CMMI son abstracciones las cuales son implementadas y hechas realidad cuando son aplicadas por los proyectos y las organizaciones. El contexto dentro del cual la práctica es aplicada genera la implementación.

  Las instancias incluyen: Detalles de implementación y Contexto en el que la práctica es implementada. Pueden haber a nivel organizacional o a nivel del proyecto.

- **Muestras e instancias dentro del proceso de certificación:** Se identifica dentro de la unidad organizacional una muestra representativa de las prácticas instanciadas, por ejemplo una práctica es dejar evidencia de que se sigue el proceso definido. El grado en el cual la práctica está implementada en la muestra se usa para indicar el grado en el cual la práctica esta implementada en la unidad organizacional.
- **Evidencia Objetiva:** Información cualitativa o cuantitativa, registros o documentación directamente relacionada con las características del producto o servicio o con la existencia e implementación de un elemento del proceso, los cuales están basados en la observación, medición o prueba y que son verificables. Los métodos de certificación CMMI están basados sobre las consideraciones de la evidencia objetiva así como en la formulación de las debilidades, fortalezas y mediciones.
- **Fuentes de evidencia objetiva(el elemento más importante para esta tesis):** las fuentes de evidencia objetiva incluyen:
  -*instrumentos*: información escrita relacionada con la implementación en la unidad organizacional.(Cuestionarios, encuestas o mapeo organizacional de las prácticas del modelo a sus procesos correspondientes)
  -*presentaciones*: Información preparada por la organización y entregada visualmente o verbalmente al equipo evaluador para describir los procesos organizacionales y la implementación de las prácticas del modelo. Incluye normalmente resúmenes de visión general y demostraciones de herramientas o capacidades.
  -*documentos*: Artefactos que reflejen la implementación de una o más prácticas del modelo; políticas de organización, procedimientos y artefactos a nivel de implementación. Los documentos pueden estar disponibles en hardcopy o softcopy.
  -*entrevistas*: interacción cara a cara con aquellos que implementan o usan los procesos dentro de la unidad organizacional. Típicamente las entrevistas se realizan con varios grupos o individuos tales como líderes de proyecto, gerentes y practitioners. Pueden realizarse una combinación de entrevistas o preguntas exploratorias desarrolladas para elicitar la información necesaria.

Las Premisas para certificar CMMI son:

- El objetivo logrado es una función de la extensión de las prácticas correspondientes que están presentes en los procesos planificados e implementados de la organización.
- La implementación de las prácticas a nivel de unidad organizacional es una función del grado de implementación de la práctica en el nivel de instanciación.
- El agregado de la disponibilidad de evidencia objetiva para el equipo de certificadores es usada como base para determinar la implementación de la práctica.
- El equipo de certificadores son obligados a buscar y considerar la evidencia objetiva de múltiples tipos para determinar la extensión de la implementación de la práctica.

Las 4 funciones principales de las evaluaciones son:

1.- Analizar: Se analiza como una organización trabaja en comparación a un modelo de referencia. Las evaluaciones estabilizan el proceso y priorizan el cambio.

2.- Motivar: : Las evaluaciones sirven como soporte al cambio, producen cambios involucrando y motivando a las organizaciones en esfuerzos de auto análisis..

3.- Transformar: Haciendo que personas diferentes vean las mismas cosas de la misma forma. Ayudando los esfuerzos de unificación de la gerencia. Suavizando una cultura de censura, permitiendo al personal la libertad de pensar acerca de qué se hace de forma equivocada y cómo corregirlo. Consolidando las mejoras y ayudando a institucionalizar el análisis riguroso.

4.- Educar: Exponiendo a las organizaciones las mejores prácticas mundiales. Las Evaluaciones educan proporcionando a las personas en las organizaciones un amplio conocimiento de su propia compañía y estimulando a las organizaciones a contemplar las mejores prácticas de la industria y compararlas con la organización.

4.2.3 Tipos de indicadores de implementación de prácticas

Los indicadores de implementación de las prácticas (PII´s Practice Implementation Indicators) son "huellas", evidencia de la implementación de una práctica de CMMI. Se utilizan para verificar la implementación de prácticas. La verificación consiste en revisar Evidencia Objetiva para determinar si una práctica está implementada.

Artefacto directo Productos tangibles resultado directo de la implementación de una práctica genérica o específica. Parte integral de la verificación de la implementación de una práctica. Puede estar explícitamente declarado o implícito en la declaración de la práctica o material asociado: Materiales típicos enunciados (typical workproducts) – productos típicos que responden a "establecer y mantener" – documentos, productos entregables, materiales de entrenamiento, etc. Se necesitan uno o varios artefactos directos para verificar la implementación del modelo. Para ser recolectadas previo a la evaluación.

Artefacto indirecto Son una consecuencia de ejecutar una práctica específica o genérica o de hacer tangible su implementación, pero no son el objetivo para el cual fueron realizados. Este tipo de indicador resuelve las dudas de si es que los objetivos de una determinada práctica se cumplen. Por ejemplo: Minutas de reunión, revisión de resultados, reportes de status, métricas de performance. Para ser recolectadas previo a la evaluación.

Afirmación – Declaraciones verbales (en entrevistas) o escritas, confirmando o apoyando la implementación de una práctica específica o genérica. Usualmente las generan los que implementan la práctica o los stakeholders: clientes internos, externos, gerentes,, pueden incluir por ejemplo entrevistas presenciales, video conferencia,., para ser recolectadas durante la evaluación

Durante una evaluación, todos estos indicadores se utilizan para verificar la evidencia objetiva. La organización proporciona un conjunto de evidencias objetivas de prácticas que satisfacen los objetivos del CMMI al inicio de la evaluación, el equipo verificará si es que realmente fueron implementadas. Para análisis a nivel de proyecto, se debe confirmar que cada proyecto implementa dichas prácticas. Para prácticas a nivel

organizacional, el equipo debe observar implementación a nivel organizacional así como implementación de las mismas en las actividades del proyecto.

El equipo debe confirmar:

- La validez de artefactos directos, indirectos y afirmaciones (Ej. Documentos, actas de reunión, declaraciones).
- Los artefactos directos, corroborados por los indirectos y que las afirmaciones verifiquen la implementación de cada práctica de CMMI necesaria.
- Que cada práctica del modelo dentro del alcance de la evaluación pueda ser demostrada, al menos 50% de las prácticas de la organización deben implementar las metas especificas y genéricas CMMI

*4.2.4 Estructura de los diferentes métodos de ARC (*Appraisal Requirements for CMMI*).*

La estructura de clases identifican los requerimientos adecuados para los métodos de evaluación específicamente diseñados para tres tipos de aplicaciones típicas. No todos los métodos de evaluación de CMMI se ajustan completamente a los requisitos de ARC y no se requiere que un método de evaluación de CMMI se ajuste exactamente a una clase de método.

Los atributos que diferencian las clases de evaluaciones incluyen:

- El grado de confianza de los resultados de las evaluaciones
- Generación de métricas.
- Costo y duración de la evaluación

Los métodos de clase A deben satisfacer todos los requerimientos de ARC y por ahora son los únicos métodos considerados aptos para obtener un nivel de certificación. Para los métodos de clase B se requiere que cumplan con un subconjunto de los requerimientos ARC. Muchos de los requerimientos para los métodos de clase A son opcionales para los métodos de clase B. Para los métodos de clase A y B se requiere dos tipos de evidencia objetiva. Los de clase B no necesitan generar mediciones. Este tipo

de evaluaciones son recomendadas para revisiones iniciales en organizaciones que recién han comenzado en el uso del modelo CMMI en las actividades de mejora de los procesos. También provee un medio efectivo para realizar evaluaciones intermedias entre certificaciones de clase A

Para los métodos de clase C se requiere que cumplan un subconjunto de los requerimientos que necesitan los de clase B. Sólo un tipo de evidencia objetiva de los dos que requiere el método de clase A es requerido por el método de clase C. Este tipo de evaluaciones debería ser utilizado cuando se necesita una mirada rápida de los procesos o para auto evaluaciones periódicas de los proyectos y de la organización.

Características	Clase A	Clase B	Clase C
Evidencia Objetiva	Alta	Media	Baja
Obtiene Valoración (Certificación)	SI	NO	NO
Utilización de Recursos	Altos	Medios	Bajos
Tamaño del Equipo de Evaluación	Grande	Medio	Reducido
Responsable de la Evaluación	Asesor Certificado	Asesor Formado y Experim.	Asesor Formado
Duración	2-3 meses	2-3 sem.	2-3 días

Tabla 8: Características de las Clases de Evaluación del modelo CMMI

Las clases de evaluación sirven de base para quienes quieran desarrollar métodos alternativos de evaluación para los modelos de CMMI. Como mencionamos la clase A define los métodos que hacen una comprobación exhaustiva de todas las características de CMMI.

Los métodos de certificación pueden ser aplicados para diferentes propósitos, inclusive como herramientas para mejoras de procesos internos y para evaluaciones de capacidad para seleccionar proveedores y para monitorear procesos.

El SEI solo reconoce las evaluaciones con el método SCAMPI como válidas a la hora de determinar el nivel de madurez de una organización.

4.3 SCAMPI - Método Estándar de Evaluación de CMMI para Mejora de Procesos

4.3.1 Introducción

Para llevar a cabo la evaluación basada en CMMI el SEI ha diseñado el Método Estándar de Evaluación de CMMI para Mejora de Procesos (Standard CMMI Appraisal Method for Process Improvement, SCAMPI), que consiste en una serie de métodos formales para la evaluación del modelo, que pueden usarse para evaluar:

- Si los procesos tal y como están definidos son adecuados según los requisitos de CMMI
- Cómo esos procesos se están desplegando en la organización
- Cómo los procesos están institucionalizados en la organización

El uso de SCAMPI nos permite:

- Comprender mejor el nivel de competencia en ingeniería de una organización, identificando los puntos fuertes y débiles de sus procesos actuales.
- Relacionar esos puntos fuertes y débiles con el modelo CMMI.
- Priorizar planes de mejora.
- Centrarse en las mejoras más importantes que haya que acometer según el nivel de madurez de la organización y de los recursos disponibles.
- Obtener para la organización su clasificación en uno de los niveles del modelo.
- Identificar riesgos de desarrollo y adquisición relativos a las limitaciones de la organización.

4.3.2 Objetivos

Los objetivos de SCAMPI son:

- Proveer un método de certificación común e integrado capaz de soportar certificaciones en el contexto de mejoras de procesos internos, selección de proveedores y monitoreo de procesos.
- Proveer un método eficiente de certificación capaz de ser implementado dentro de restricciones razonables de performance.

Para poder cumplir con el método de evaluación SCAMPI, el trabajo se debe organizar en tres fases

1. Planificar y preparar la certificación: lleva de 3 a 5 meses
2. Conducir la certificación: ejecución de la evaluación y reportes los resultados preeliminares
3. Reportar los resultados de la certificación: reportes de los resultados finales.

Las tareas a realizar dentro del SCAMPI son:

1. Desarrollar un plan de certificación.
2. Determinar los indicadores de implementación de las prácticas(PII´s)
3. Entrevistas, con los empleados, gerencia y demás participantes.
4. Seleccionar y preparar el equipo de certificación.
5. Obtener y analizar evidencia objetiva preeliminar
6. Preparar una colección de evidencia objetiva.
7. Examinar la evidencia
8. Verificar y validar la evidencia
9. Documentar la evidencia
10. Generar reportes de los resultados de la evaluación
10. Publicar los resultados de la evaluación
11. Empaquetar y archivar los instrumentos de certificación.

Adelantándonos al próximo capítulo, podemos decir que la herramienta a construir podrá ser usada como apoyo para ejecutar varias de estas actividades, como por ejemplo para examinar la evidencia, documentar la evidencia, generar reportes etc.

4.3.3 Clases de Métodos

En función de su grado de adaptación y rigurosidad se distingue entre:

- SCAMPI-C: Mide la idoneidad de los procesos, mediante entrevistas o revisión documental. Es el método idóneo para poder obtener una foto rápida del estado de los procesos en una organización para comenzar un programa de mejora de procesos.
- SCAMPI-B: Permite evaluar la idoneidad y el grado de despliegue de los procesos, mediante entrevistas o revisión documental. Es recomendable para hacer auditorías de los procesos de una organización antes de afrontar el proceso de certificación con la evaluación formal.
- SCAMPI-A: Es el más formal que mide la idoneidad, despliegue e institucionalización de los procesos. Es el necesario para poder obtener un certificado de un determinado nivel de madurez. El SEI publica los resultados de los SCAMPI-A llevados a cabo a nivel internacional en [13].

El método formal de evaluación SCAMPI-A tiene una serie de requisitos añadidos:

- Debe ser realizado por una persona acreditada por el SEI como SCAMPI Leader Appraiser
- Se debe formar un equipo de evaluación (Assessment Team Members) de al menos 4 personas, y todos sus miembros deben haber pasado el curso oficial de introducción a CMMI.
- El equipo de evaluación debe tener una experiencia mínima (6 años de experiencia media y 25 años en total en desarrollo de software, 10 años en gestión) en las disciplinas que son objeto de la evaluación
- Para garantizar la objetividad de las evaluaciones, las personas que participan como equipo de evaluación no pueden tener responsabilidad sobre los proyectos seleccionados y personas a entrevistar.

A pesar de que el método SCAMPI "Clase A" cumple con todos los requerimientos definidos por el SEI para esta clase de métodos en el documento "Appraisal Requirements for CMMI, (ARC)" algunos casos de estudio han demostrado que el uso de este método de evaluación involucra altos costos y consume mucho tiempo para poder obtener resultados [Zubrow,16]. Por tanto, no es factible para muchas organizaciones emplear una evaluación "Clase A", sobre todo en pequeñas organizaciones, por lo que para estos casos una evaluación "Clase B o C" es la más adecuada.

Todos los SCAMPIs deben ser supervisados por agentes autorizados del SEI, inclusive C y B para garantizar interpretaciones correctas y "autorizadas".

4.4 SCAMPI Leader Appraiser

Las evaluaciones de las organizaciones se llevan a cabo por supervisores de evaluación externos que tienen la autorización del SEI. Estos supervisores han recibido la formación necesaria y tienen acceso a métodos de evaluación, materiales de formación, asistencia técnica y actualización formativa proporcionados por el SEI. A través de su participación en evaluaciones de organizaciones y de los mecanismos de realimentación previstos en los métodos de evaluación, los supervisores de evaluación contribuyen a la mejora continua de la tecnología de evaluación del SEI.

Para que un profesional tenga la consideración de supervisor de evaluación SCAMPI debe estar en posesión del informe favorable que acredite que ha superado el plan formativo para supervisores de evaluación diseñado por el SEI. Para acceder a esta formación son necesarios los siguientes requisitos:

1. El SEI debe haber aceptado como asociada para servicios de evaluación SCAMPI a la organización a la que el profesional pertenezca.
2. Completar con éxito el proceso de selección, acreditando los conocimientos mínimos requeridos. Se exige haber formado parte de un equipo de evaluación SCAMPI en al menos dos evaluaciones en los dos años inmediatamente anteriores a la solicitud.
3. Aprobar un curso de introducción a CMMI.
4. Aprobar un curso de conocimientos intermedios de CMMI.

Capítulo 5 - Presentación del sistema

5.1 Introducción

El software a desarrollar, se construirá teniendo en mente que será utilizado por el equipo de QA dentro de una organización, por lo que, para comenzar daremos un breve pantallazo de las tareas que realizan los miembros del equipo.

El equipo QA generalmente esta compuesto por expertos de QA y revisores de QA (analista de QA), los expertos dan soporte a los revisores durante las revisiones. Los revisores de QA son personal experimentado, entrenado en los procesos y actividades de QA, idealmente son externos (independientes) al proyecto que relevarán.

Para cada proyecto el equipo de QA debe pasar por tres etapas. La primera de ellas, es planificar las revisiones de QA, es decir, planificar las actividades de QA, las distintas auditorías que se realizarán sobre el proyecto en los distintos estadios del mismo.
Luego, la segunda etapa, consiste en ejecutar las revisiones de QA, es decir, realizar las auditorías, por ejemplo entre las tareas de esta fase se encuentra:

1. Asegurar la disponibilidad de la documentación y que la misma esté actualizada con las últimas revisiones.
2. Verificar que la documentación generada cumpla con los estándares y planes de proyecto.
3. Documentar el resultado de las revisiones de QA.
4. Definir un plan de acción para resolver las no conformidades.
5. Dar a conocer los resultados de las revisiones de QA y el plan de acción a ejecutar.

Por último, la tercera etapa, consiste en hacer un seguimiento de las actividades de QA, para poder

1. Verificar el cumplimiento del plan de acción definido.
2. Hacer seguimiento de los planes de acción y resolución de no conformidades.
3. Detectar fortalezas y debilidades de los procesos definidos.

Para cada proyecto las responsabilidades del revisor de QA son:

- Liderar la revisión de QA.
- Revisar la documentación.
- Moderar las entrevistas
- Completar los resultados.
- Colaborar activamente con el coordinador del proyecto para crear el plan de acción.
- Seguir el plan de acción de QA asegurando que se resuelvan las no conformidades detectadas.

5.2 Objetivo del sistema

El objetivo del sistema que propone ésta tesis será ayudar a alcanzar la certificación de nivel 2 para el modelo CMMI, a través del seguimiento de las auditorías y del registro y tratamiento de las no conformidades, requeridos por el área de PPQA para este nivel dejando evidencia de la aplicación de control de los procesos definidos.

5.3 Funcionalidad

La funcionalidad mínima que deberá proveer el sistema, es aquella necesaria para cubrir los requerimientos del área de soporte PPQA, que ya fueron detallados en el capítulo 3. Recordemos que las metas de esta área de proceso son:

- Evaluar objetivamente la ejecución de los procesos, los elementos de trabajo y servicios en contraste a los procesos, estándares y procedimientos definidos.
- Identificar y documentar no conformidades. Recordemos que definimos una no conformidad como el incumplimiento de un estándar o proceso de la organización o del proyecto.
- Proporcionar información a las personas que están usando los procesos y a la gerencia del proyecto, de los resultados de las actividades del aseguramiento de la calidad.
- Asegurar de que las no conformidades son tratadas.

Como ya mencionamos en la introducción, la idea es realizar un sistema que ayude en la tarea diaria a la persona encargada de realizar QA. Su función consistirá, principalmente, en generar evidencia sobre la forma de trabajar de una organización.

Esto es, generar evidencia que muestre que la organización realmente aplica el proceso que dice aplicar y que cuando no lo hace, permita registrar tales fallas como no conformidades y poner en funcionamiento los mecanismos que sean necesarios para corregirlas. Por lo que el sistema deberá permitir, registrar la ejecución de auditorías sobre los distintos proyectos de la organización, registrar las no conformidades encontradas en las auditorías realizadas, debe permitir el seguimiento y control de las no conformidades encontradas, notificando al nivel de autoridad que corresponda. El sistema deberá también facilitar la realización de auditorías sobre los proyectos permitiendo agendar las distintas auditorías y controlando su vencimiento.

Para ver en detalle la funcionalidad que vamos a implementar a continuación definimos nuestros casos de usos. Para su mejor comprensión se recomienda leer el apéndice de esta tesis.

5.3.1 Casos de uso

El objetivo de esta sección es ayudar a comprender cual es la funcionalidad de nuestro sistema. A fin de no extendernos demasiado y con la certeza de que no se va a ver comprometido el entendimiento, para los casos de ABM solo nos limitamos a especificar las altas, que pensamos que son mas interesantes respecto de la interacción con el usuario. Creemos que las bajas y modificaciones no aportan mucho a la comprensión.

Antes de comenzar a describir los casos de uso, algunas aclaraciones para que se entiendan mejor:

- **Auditoría pendiente:** entendemos como auditoría pendiente, a aquella auditoría asociada a un proyecto que tiene tareas en estado “no iniciada” o “abierta”.
- **No conformidad**: para nosotros una no conformidad es una tarea abierta. Un tarea define un aspecto a verificar como por ejemplo “Que el documento cumpla con el template”.
- **Proyecto en gestión**: un proyecto en gestión es aquel que no se encuentra en el estado finalizado.

- **RN**: son reglas de negocio, nosotros aquí las utilizamos como notas del caso de uso para la aclaración de cuestiones que ayudan a entender el funcionamiento del caso de uso pero que exceden la interacción del usuario con el sistema.

Diagrama de los casos de uso mas importantes

Vamos a presentar tres diagramas, cada uno de ellos agrupa los casos de uso de una vista del sistema, el panel de control, la agenda y por último la configuración,

Panel de Control.

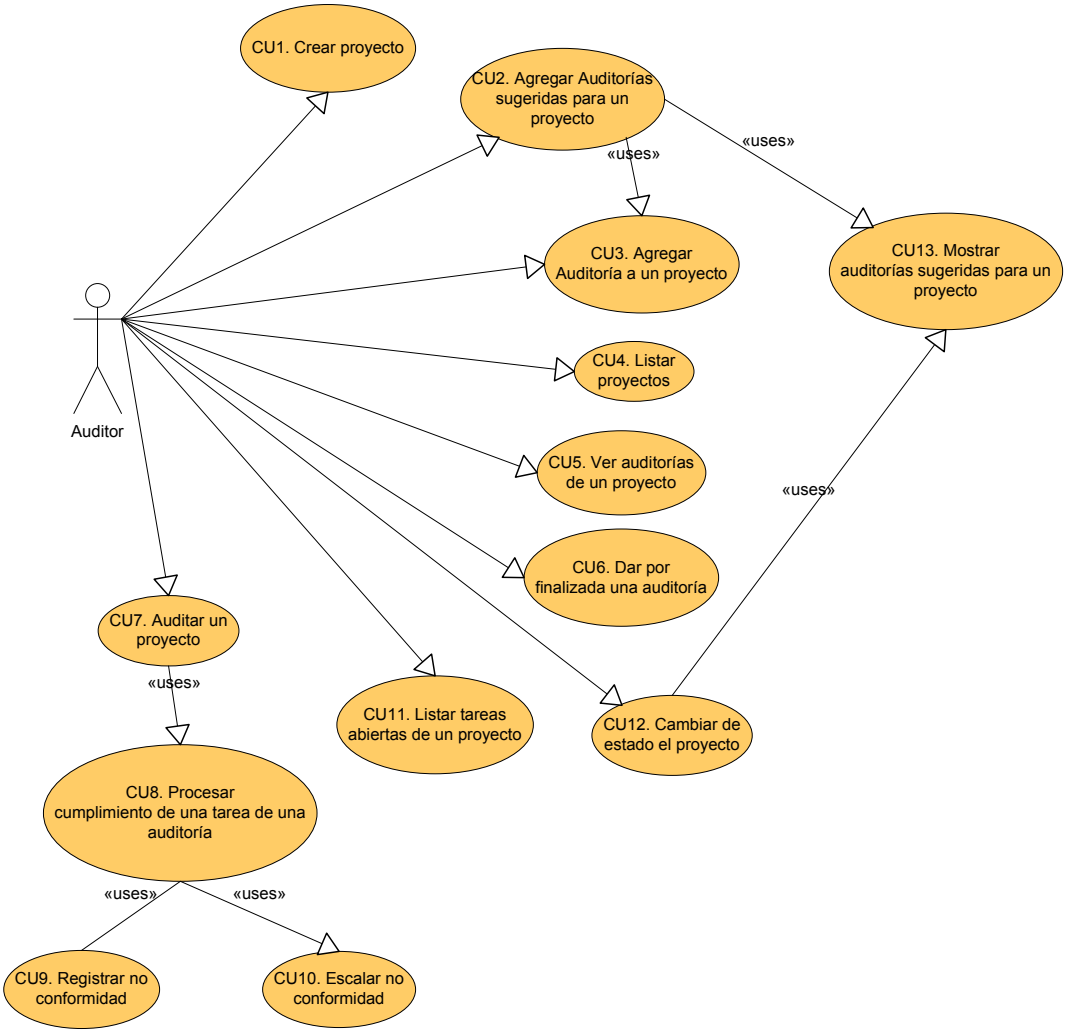


Figura 11: Diagrama de los casos de usos del panel de control.

Agenda

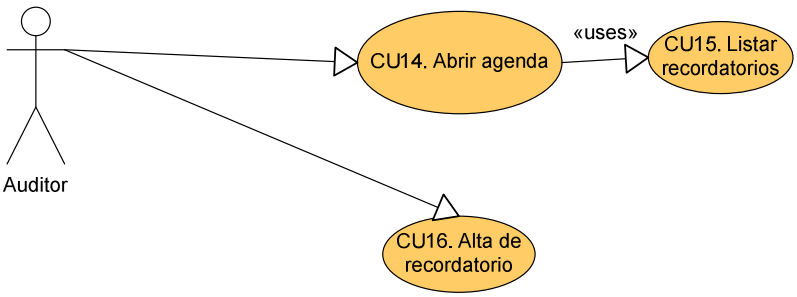


Figura 12: Diagrama de los casos de usos de la agenda.

Configuración

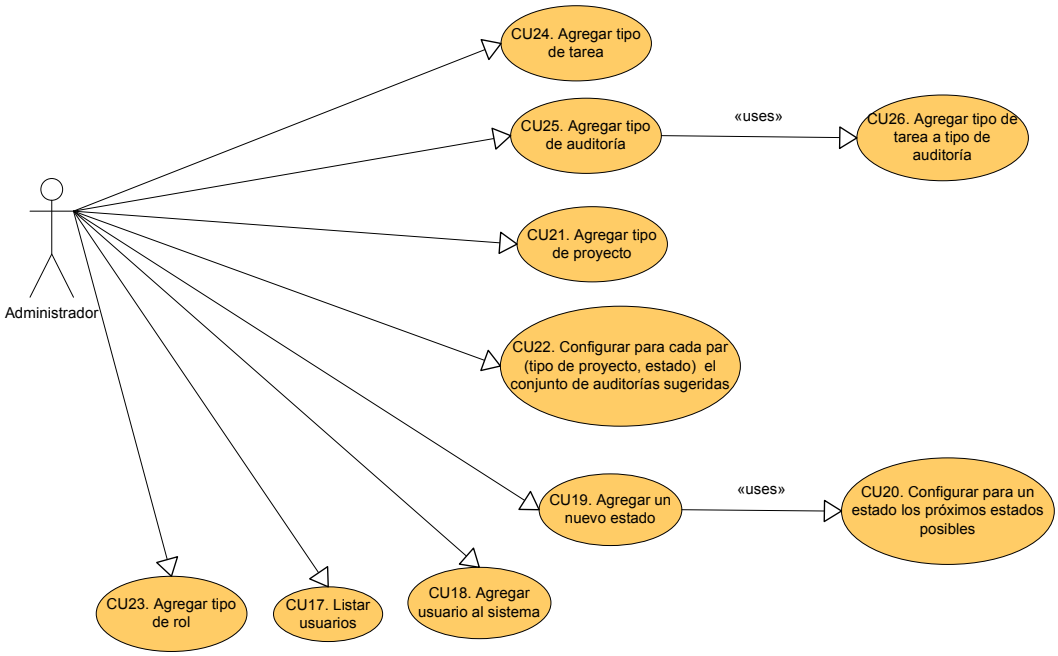


Figura 13: Diagrama de los casos de uso de la vista de configuración

Detalle de los casos de uso mas importantes

CU 1_Crear proyecto

Objetivo: Agregar un nuevo proyecto al sistema.
Contexto de uso: Cuando llega un nuevo pedido, se da de alta en el sistema el proyecto correspondiente.
Precondiciones:

Poscondiciones:

1. El proyecto se dio de alta en el sistema.
2. El proyecto esta en gestión.
3. Si eligió aceptar las auditorías sugeridas, las auditorías sugeridas están dadas de altas en estado pendientes.

Escenario principal:

00_Proyecto dado de alta

1.El usuario solicita dar de alta un nuevo proyecto.
2.El sistema solicita los datos requeridos, nombre del proyecto, tipo de proyecto y que usuarios van a cumplir los distintos roles.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
4.En caso que existan auditorías para el tipo de proyecto seleccionado, el sistema muestra una lista de auditorías sugeridas.
5.El usuario puede seleccionar las auditorías sugeridas para el tipo de proyecto.
6.Invocar **2_Agregar Auditorías sugeridas para un proyecto.**
7.El sistema completa el estado del proyecto.
8.El sistema informa que el proyecto ha sido dado de alta.
9.FIN

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no esta completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos proyectos con el mismo nombre.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 El sistema vuelve al paso 2 para solicitar que se modifiquen los datos inválidos.
3.b.3 FIN.

Escenario Alternativo:

A1_El usuario no desea crear las auditorías sugeridas para el tipo de proyecto
4.a Condición: El usuario no desea crear las auditorías sugeridas para el tipo de proyecto.
4.a.1 va al paso 7 para completar el alta del proyecto.
4.a.2 FIN.

CU 2_Agregar Auditorías sugeridas para un proyecto.

Objetivo: Agregar a un proyecto las auditorías que se sugieren en función de su tipo y estado.

Contexto de uso: Cuando se da de alta un nuevo proyecto o cuando a uno existente se le modifica su estado se pueden asociarle las auditorías sugeridas por el sistema para el tipo de proyecto en el estado actual.
Precondiciones:

1. El proyecto se encuentra en estado en gestión.

Poscondiciones:

1. Las auditorías han sido agregadas al proyecto.

Escenario principal:

00_Auditorías agregadas al proyecto.

1.Invocar **13_ Mostrar auditorías sugeridas para un proyecto**
2.El usuario selecciona los tipos de auditorías a agregar.
3.El sistema configura al proyecto según RN02.
4.FIN.

RN02
El sistema crea las auditorías seleccionadas por el usuario y las agrega a la lista de auditorías pendientes del proyecto.

CU 3_Agregar Auditoría a un proyecto

Objetivo: Agregar a un proyecto una nueva auditoría.
Contexto de uso: En un momento posterior al alta del proyecto el usuario decide agregarle una nueva auditoría a realizar.
Descripción resumida:
Precondiciones:

1. El proyecto se encuentra en estado en gestión.

Poscondiciones:

1. La auditoría ha sido agregada al proyecto elegido.

Escenario principal:

00_Auditoría agregada exitosamente
1.El usuario solicita agregar una auditoría a un proyecto
2.El usuario selecciona un proyecto en estado en gestión.
3.El sistema lista los distintos tipos de auditorías.
4.El usuario selecciona el tipo de auditoría que desea agregarle al proyecto.
5.El sistema crea una nueva auditoría del tipo elegido.
6.El sistema actualiza el proyecto según RN01.
7. El sistema informa el éxito de la operación.
8.FIN

RN01
Luego de crear la nueva auditoría el sistema la agrega a la lista de auditorías pendientes del proyecto seleccionado.

CU 4_Listar proyectos

Objetivo: Ver la lista de los proyectos en gestión y finalizados.
Contexto de uso: El usuario debe tener acceso a los proyectos cargados en el sistema para poder auditarlos.
Descripción resumida: Cuando el usuario ingresa al sistema se le muestra la lista de proyectos para que pueda empezar a operar. Para administrar los proyectos se deben mostrar para cada proyecto de la lista, entre otra información, el estado en el que se encuentra.
Precondiciones:

1. Existen proyectos para mostrar.

Poscondiciones:

1. Se listan los proyectos.

Escenario principal:

00_Lista de proyecto
1.El usuario solicita ver la lista de proyectos.
2. El sistema mostrará la lista de proyectos mostrando para cada proyecto sus datos más relevantes e información de su estado.
3.FIN.

CU 5_Ver auditorías de un proyecto

Objetivo: Poder ver para un proyecto las auditorías asociadas.
Contexto de uso: una vez listados los proyectos el usuario puede ver el estado de las auditorías realizadas a un proyecto seleccionado.
Descripción resumida: El usuario podrá ver las auditorías pendientes y finalizadas. También vera las tareas abiertas de las auditorías realizadas sobre el proyecto.
Precondiciones:

1. Hay un proyecto seleccionado.

Poscondiciones:

1. Se listan las auditorías asociadas al proyecto, junto con sus tareas abiertas y cerradas.

Escenario principal:

00_Lista de auditorías
1. El usuario solicita ver las auditorías asociadas a un proyecto.
2. El sistema muestra la lista de las auditorías asociadas al proyecto.
3. El sistema muestra las tareas asociadas a las auditorías del proyecto.
4. El sistema indica el estado de las auditorías listadas, si están pendientes o finalizadas.
5. FIN.

CU 6_Dar por finalizada una auditoría

Objetivo: Dar por cumplida una auditoría
Contexto de uso: hay situaciones en las que se requiere dar por cumplida una auditoría pendiente de un proyecto.

Descripción resumida: Dar por finalizada una auditoría implica cerrarla y pasarla a la lista de auditorías finalizadas.
Precondiciones:

1. El sistema verifica que haya una auditoría seleccionada.
2. El sistema verifica que la auditoría seleccionada este en estado pendiente.

Poscondiciones:

1. La auditoría fue finalizada.

Escenario principal:
00_Auditoría finalizada.
1. El usuario solicita dar por finalizada una auditoría pendiente (con tareas no iniciadas o abiertas.)
2. El sistema pregunta al usuario si desea dar por cumplida la auditoría.
3. El usuario responde que quiere dar por cumplida la auditoría.
4. El sistema cierra la auditoría y deja las tareas en el estado que en se encuentran para que puedan ser consultadas.
5. El sistema pasa la auditoría del proyecto de pendiente a finalizada.
6. El sistema informa el éxito de la operación.
7.FIN.

Escenario alternativo:

A1_El usuario responde que no quiere cerrar la auditoría
3.1Condicion: El usuario responde que no quiere cerrar la auditoría
3.1.a FIN

CU 7_Auditar un proyecto.

Objetivo: Realizar una auditoría sobre un proyecto
Contexto de uso: Este caso se de uso se dispara cuando un usuario quiere realizar una auditoría sobre un proyecto
Descripción resumida: Permite auditar un proyecto en gestión
Precondiciones:

1. El proyecto a auditar esta en gestión.

Poscondiciones:

1. El proyecto esta auditado

Escenario Principal:

00_Audita proyecto

1.El usuario solicita realizar auditoría sobre un proyecto.
2.El sistema lista las auditorías pendientes del proyecto.
3.El usuario selecciona una auditoría.
4.Invocar **8_ Procesar cumplimiento de una tarea de una auditoría**
5.El sistema actualiza los datos de la auditoría, fecha de actual y quien la realizó.
6.El sistema informa éxito de la auditoría.
7.FIN.

Escenario alternativo

A1 La auditoría a realizar no está entre las pendientes del proyecto
3.1 Condición: La auditoría a realizar no esta entre las pendientes del proyecto
3.a.1 Invocar **3_Agregar Auditoría a un proyecto.**
3.a.2 El sistema vuelve al paso 2 para que el usuario seleccione una auditoría.
3.a.3 FIN.

CU 8_Procesar cumplimiento de una tarea de una auditoría

Objetivo: Verificar el cumplimiento de una tarea.
Contexto de uso: El realizar una auditoría sobre un proyecto implica controlar el cumplimiento de las tareas requeridas por la misma.
Descripción resumida: Permite abrir o cerrar una tarea de las requerida por la auditoría en cuestión.
Precondiciones:

1. La tarea elegida no esta en estado "Cerrada".

Poscondiciones:

1. La tarea no esta en estado "no iniciada"

Escenario Principal:

00_Tarea verificada

1. El sistema lista todas las tareas para cumplir la auditoría seleccionada.
2. El usuario selecciona la tarea a verificar.
3. El usuario marca la tarea como cumplida.
4. El sistema pasa a estado "Cerrada" a la tarea
5 El sistema procesa la auditoría según RN03
8.FIN

Escenario alternativo

A1_El usuario marca la tarea como no cumplida.
3 Condición: El usuario marca la tarea como no cumplida.
3.a.1 La tarea esta en estado abierta
3.a.2 Invocar **10_Escalar no conformidad**.
3.a.3 FIN.

Escenario alternativo

A1A1 La tarea esta en estado no iniciada
3.a.1.a Condición: La tarea esta en estado no iniciada
3.a.1.a.1 Invocar **9_Registrar no conformidad**.
3.a.1.a.2 FIN

RN03
Si todas las tareas de la auditoría están en estado "cerrada" entonces el sistema da por cumplida la auditoría. Esto implica sacar la auditoría de las lista de auditorías pendientes del proyecto y agregarla a la lista de auditorías finalizadas.

CU 9_Registrar no conformidad

Objetivo: Registrar una no conformidad.
Contexto de uso: Cuando se está verificando el cumplimiento de una tarea, si esta no se cumple se debe registrar una no conformidad.
Descripción resumida: Permite asociar no conformidades que surgen al auditar un proyecto en gestión.
Precondiciones:

1. La tarea esta en estado "no iniciada"

Poscondiciones:

1. La tarea esta en estado "abierta".

Escenario Principal:

00_Registra no conformidad

1. El sistema pone a la tarea en estado "abierta".
2. El sistema solicita una descripción.
3. El sistema completa información de estado de la tarea, auditor, fecha y nivel de escalamiento.
4.El sistema informa de la no conformidad al nivel correspondiente.
5.El sistema informa el éxito de la operación.
7.FIN

CU 10_Escalar no conformidad

Objetivo: Informar a las autoridades del proyecto que hay una no conformidad abierta.
Contexto de uso: Cuando se está auditando un proyecto y se encuentra que una tarea esta abierta luego del plazo establecido se debe escalar la no conformidad
Descripción resumida: Permite escalar o notificar a los responsables del proyecto cuando una tarea abierta esta sin solución durante el lapso que tiene de acuerdo al tipo de auditoría.
Precondiciones:

1. La tarea esta en estado "abierta"

Poscondiciones:

1. La no conformidad fue escalada.
2. La tarea tiene información actualizada.

Escenario Principal:

00_No conformidad escalada
1. El usuario solicita al sistema escalar una no conformidad.
2. El sistema informa al próximo nivel de escalamiento.
3. El sistema actualiza la información de la tarea, fecha de actual y auditor.
4. El sistema informa el éxito de la operación.
5.FIN.

CU 11_Listar tareas abiertas de un proyecto

Objetivo: Ver todas las tareas abiertas sobre un proyecto.
Contexto de uso: Cuando el usuario ingresa al sistema puede solicitar ver cuales son las no conformidades asociadas a un proyecto, es decir, todas las tareas abiertas.
Descripción resumida: Se mostrará la lista de tareas abiertas de l proyecto seleccionado, indicando a que auditoría pertenece , el motivo y la fecha en la que se reportó.
Precondiciones:

1. Hay un proyecto seleccionado.
2. Existen tareas abiertas asociadas al proyecto elegido.

Poscondiciones:

1.Se muestran las tareas abiertas del proyecto.

Escenario Principal:

00_Lista de tareas abiertas.
1.El usuario solicita ver la lista de no conformidades de un proyecto.
2.El sistema lista las tareas abiertas del proyecto elegido.
3.FIN.

CU 12_ Cambiar de estado el proyecto.

Objetivo: Poder indicar el estado de un proyecto
Contexto de uso:
Descripción resumida: Permite cambiar de estado a un proyecto.
Precondiciones:

1. Hay un proyecto seleccionado.

Poscondiciones:

1. El proyecto cambio de estado.

Escenario Principal:

00_Cambio de estado

1. El usuario solicita cambiar el estado de un proyecto
2. El sistema muestra los posibles estados destino en función del estado origen.
3. El usuario selecciona un estado destino.
4. El sistema cambia de estado al proyecto.
5. El sistema muestra la lista de auditorías sugeridas.

Invocar **13 Mostrar auditorías sugeridas para un proyecto**

6. El sistema informa el éxito de la operación.
7.FIN.

CU 13_ Mostrar auditorías sugeridas para un proyecto

Objetivo: Consultar cual es el próximo tipo auditoría sugerida en función del estado y del tipo del proyecto.

Contexto de uso: Un usuario puede pedirle al sistema que le sugiera cuales son los tipos de auditorías que se pueden realizar en el estado y tipo del proyecto sobre el que se encuentra trabajando.
Descripción resumida: Permite consultar el próximo tipo de auditoría sugerida para un proyecto. El sistema sugiere la próxima auditoría en función del tipo del proyecto seleccionado y de su estado.
Precondiciones:

1. Hay un proyecto seleccionado.

Poscondiciones:

1. Lista de auditorías sugeridas.

Escenario Principal:

00_Lista de auditorías sugeridas
1.El usuario solicita ver los próximos tipos de auditorías sugeridas.
2. El sistema lista los tipos de auditorías sugeridas.
3.FIN.

CU 14_Abrir agenda.

Objetivo: Mostrar las auditorías y tareas abiertas que hay agendadas por el usuario para los distintos proyectos.
Contexto de uso: Cuando el usuario ingresa al sistema puede consultar la agenda para agendar o listar las auditorías o tareas abiertas. Las auditorías y tareas que se agenden luego a su debido tiempo se le recordaran al usuario cuando ingrese al sistema.
Descripción resumida: El sistema muestra al usuario una bandeja de entrada donde se le muestran las auditorías o tareas abiertas que tiene agendadas.
Precondiciones:
Poscondiciones:

Escenario Principal:
00_Agenda
1.El usuario solicita abrir la agenda.
2. Invocar al **CU 15 Listar recordatorios.**
3.FIN.

CU 15_Listar recordatorios

Objetivo: Acceder a la lista de todas las tareas abiertas y auditorías pendientes agendadas por el usuario.
Contexto de uso: Cuando el usuario realiza una auditoría sobre un proyecto y encuentra no conformidades puede que éste desee agendar la no conformidad para no olvidarse de chequearla mas tarde. También existe la posibilidad de agendar la realización de la auditoría en cuestión o de una próxima.
Descripción resumida: Cuando el usuario ingresa a la aplicación se le muestran los recordatorios vencidos o próximos a vencer que tiene agendados. También puede acceder a la agenda del sistema.
Precondiciones:

1. Existen recordatorios asociados al usuario para mostrar.

Poscondiciones:

1. Se muestran todas las tareas abiertas y las auditorías agendadas por el usuario logueado.

Escenario Principal:

00_Lista de recordatorios
1.El usuario solicita ver todas las auditorías por él agendadas.
2.El sistema muestra la lista de recordatorios de auditorías.
3. FIN.

Escenario Alternativo:

A1_El usuario solicita ver todas las no conformidades por el agendadas.
1.1 Condición: El usuario quiere ver todas las tareas abiertas agendadas-
1.a.1 El sistema lista los recordatorios de no conformidades agendados.
1.a.2 FIN.

CU 16_ Alta de recordatorio

Objetivo: Agendar una auditoría a realizar sobre un proyecto o una no conformidad a verificar del mismo para que el sistema le recuerde el evento en el momento señalado.
Contexto de uso: el usuario selecciona una auditoría pendiente de un proyecto o una tarea abierta de una auditoría para agendarla.
Descripción resumida: El usuario puede indicarle al sistema que le recuerde en una determinada fecha o con cierta frecuencia que le tiene que realizar una determinada auditoría a un proyecto o que tiene que rechequear una tarea abierta. Por ejemplo agendarse la próxima auditoría a realizar sobre el proyecto.
Precondiciones:
1. La auditoría o tarea a agendar no esta cumplida.

Poscondiciones:
1. La auditoría o tarea ha sido agendada.

Escenario Principal:

00_Recordatorio creado

1.El usuario solicita agendar una auditoría.
2.El sistema lista las auditorías pendientes de los proyectos.
3 El usuario selecciona una auditoría de la lista.
4 El usuario completa una descripción e indica la fecha y/o la frecuencia para el recordatorio.
5.El sistema guarda el recordatorio
6. FIN.

Escenario Alternativo:

A1_El usuario solicita agendar una tarea abierta o no conformidad.

1.1 Condición: El usuario quiere agendar una tarea abierta.
1.a.1 El sistema lista las no conformidades de los proyectos.
1.a.2 El usuario selecciona la no conformidad.
1.a.3 El sistema vuelve al paso 4 para que el usuario completa la información del recordatorio.

CU 17_Listar usuarios

Objetivo: Consultar los usuarios del sistema
Contexto de uso: Para operar sobre los usuarios el administrador debe ver la lista de todos los usuarios.
Descripción resumida: Permite ver todos los usuarios del sistema.
Precondiciones:

1. El usuario es administrador.

Poscondiciones:

1. Se listan los usuarios del sistema.

Escenario Principal:

00_Lista de usuarios

1. El usuario solicita ver la lista de usuarios.
2. El sistema lista todos los usuarios del sistema
3.FIN.

CU 18_Agregar usuario al sistema

Objetivo: Dar de alta en el sistema a un nuevo usuario.
Contexto de uso: Durante la configuración del sistema el usuario administrador debe dar de alta todos los usuarios del mismo.
Descripción resumida: Permite agregar un nuevo usuario del sistema
Precondiciones:

1. El usuario es administrador.

Poscondiciones:

1. El nuevo usuario ha sido agregado.

Escenario Principal:

00_Alta de usuario
1. El usuario solicita dar de alta un nuevo usuario
2. El sistema solicita los datos requeridos, nombre, usuario ,clave, email, rol etc.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
4. El sistema almacena el nuevo usuario.
5. FIN.

Escenario de Excepción:

E1_Los datos requeridos no están completos

3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos usuarios con el mismo nombre de usuario.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 FIN.

CU 19_Agregar un nuevo estado.

Objetivo: Agregar un estado posible para un proyecto.
Contexto de uso: Durante la configuración del sistema se deben especificar los estados por los que puede pasar un proyecto.
Descripción resumida: Permite agregar un nuevo estado.
Precondiciones:

1. El usuario es administrador

Poscondiciones:

1. El estado ha sido dado de alto.

Escenario Principal:

00_Alta de estado.

1. El usuario solicita dar de alta un nuevo estado.
2. El sistema solicita los datos requeridos, nombre.
3. Invocar al **CU 20_Configurar para un estado los próximos estados posibles.**
3. El sistema verifica que los datos requeridos estén completos y sean válidos.
4. El sistema almacena el nuevo estado
5.FIN.

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son validos.
3.b Condición: Algunos de los campos solicitados no son validos.
El sistema controla que no existan dos estados con el mismo nombre.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 El sistema vuelve al paso 2 para solicitar que se modifiquen los datos inválidos.
3.b.3 FIN.

CU 20_Configurar para un estado los próximos estados posibles

Objetivo: Configurar las transiciones de estados
Contexto de uso: Durante la configuración del sistema se deben especificar las transiciones de estados
Descripción resumida: Permite configurar las transiciones de estados.
Precondiciones:

1. Hay un estado origen seleccionado.
2. El usuario es administrador.

Poscondiciones:

1. El estado origen tiene asociados posibles estados destino.

Escenario Principal:

00_Configurar las transiciones de estados.

1. El usuario selecciona un estado para configurar las transiciones.
2. El sistema muestra la lista de estados disponibles.
3. El usuario selecciona los estados destino.
4. El sistema da de alta las transiciones.
5. FIN.

CU 21_Agregar tipo de proyecto

Objetivo: Dar de alta un nuevo tipo de proyecto.
Contexto de uso: En la etapa de configuración del sistema se deben dar de alta los diferentes tipos de proyectos.
Descripción resumida: Permite que se configuren los distintos tipos de proyectos válidos
Precondiciones:

1. El usuario es administrador

Poscondiciones:

1. El nuevo tipo de proyecto se ha agregado al sistema.

Escenario Principal:

00_Alta de tipo de proyecto

1.El usuario solicita dar de alta un nuevo tipo de proyecto.
2.El sistema solicita los datos requeridos, nombre y descripción.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
7.El sistema guarda el nuevo tipo de proyecto.
8.FIN

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.

3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos tipos de proyecto con el mismo nombre.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 El sistema vuelve al paso 2 para solicitar que se modifiquen los datos inválidos.
3.b.3 FIN

CU 22_ Configurar para cada par (tipo de proyecto, estado) el conjunto de auditorías sugeridas

Objetivo: Configurar las auditorías sugeridas para un tipo de proyecto en los diferentes estados por los que puede atravesar.
Contexto de uso: Cuando se edita un tipo de proyecto se le pueden agregar las auditorías sugeridas para el tipo de proyecto en determinados estados.
Descripción resumida: se agrega al tipo de proyecto un tipo de auditoría sugerida para el mismo.
Precondiciones:

1 Hay un tipo de proyecto seleccionado.
2. El usuario es administrador.

Poscondiciones:

1.Se agrego un tipo de auditoría al tipo de proyecto.

Escenario Principal:

00_Tipo de auditoría agregada al tipo de proyecto.

1. El usuario solicita agregar un tipo de auditoría al tipo de proyecto.
2. El usuario selecciona el estado para el que quiere sugerir un tipo de auditoría.
3. El sistema lista los tipos de auditorías existentes.
3. El usuario selecciona los tipos de auditorías a sugerir cuando los proyecto de este tipo se encuentren en el estado seleccionado.
4. El sistema asocia al tipo de proyecto y al estado con los tipos de auditorías seleccionados.
5. FIN.

CU 23_Agregar tipo de rol

Objetivo: Dar de alta un nuevo tipo de rol en sistema.
Contexto de uso: En el despliegue del sistema, el administrador deberá crear los distintos tipos de roles que tendrán los proyectos y para cada uno de ellos el nivel de escalamiento.
Descripción resumida:
Precondiciones:

1. El usuario es administrador.

Poscondiciones:

1. Se dio de alta el nuevo tipo de rol.

Escenario Principal:

00_Alta de tipo de rol
1. El usuario administrador solicita dar de alta un nuevo tipo de rol
2. El sistema solicita los datos requeridos, nombre y nivel de escalamiento.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
4. El sistema almacena el nuevo usuario.
5. FIN.

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos tipos de rol con el mismo nombre y que no existan dos tipos de rol con el mismo nivel de escalamiento.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 FIN.

CU 24_Agregar tipo de tarea

Objetivo: Configurar los tipos de tareas que determinan los aspectos a controlar que permiten detectar una no conformidad.
Contexto de uso: Cuando se instala el sistema inicialmente se deben indicar todos los tipos de tareas . El administrador dará de alta los distintos tipo de tareas que sirven para encontrar no conformidades. También se podría agregar un tipo de tarea durante la operatoria del sistema.
Descripción resumida: permite crear los tipos de tareas que indican los aspectos para describir las no conformidades que pueden tener los proyectos.
Precondiciones:
1. El usuario es administrador.

Poscondiciones:
1. El nuevo tipo de tarea se ha agregado al sistema.

Escenario Principal:
00_Alta de tipo de tarea.

1.El usuario solicita dar de alta un nuevo tipo de tarea.
2.El sistema solicita los datos requeridos, nombre y descripción.
3.El sistema verifica que los datos requeridos estén completos y sean válidos.
4.El sistema guarda el nuevo tipo de auditoría.
7.FIN.

Escenario de Excepción:

E1_Los datos requeridos no están completos
3.a Condición: Los campos obligatorios no están completos.
3.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
3.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.
3.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
3.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos tipos de tareas con el mismo nombre.
3.b.1 El sistema informa al usuario cuales campos son inválidos.
3.b.2 El sistema vuelve al paso 2 para solicitar que se modifiquen los datos inválidos.
3.b.3 FIN.

CU 25_Agregar tipo de auditoría

Objetivo: Configurar las diferentes auditorías que se pueden realizar sobre los proyectos..
Contexto de uso: cuando se instala el sistema inicialmente se deben indicar los distintos tipos de auditorías que se podrán realizar sobre los proyectos. Solo los administradores podrán realizar esta operación. También se podría agregar un tipo de auditoría durante la operatoria del sistema.

Descripción resumida: permite crear los tipos de auditorías que sirven para auditar los proyectos.
Precondiciones:
1 El usuario que realiza la operación es administrador.
Poscondiciones:
1. El nuevo tipo de auditoría se ha agregado al sistema.

Escenario Principal:

00_Alta de tipo de auditoría

1.El usuario solicita dar de alta un nuevo tipo de auditoría.
2.El sistema solicita los datos requeridos, nombre y la lista de tareas que conforman la auditoría.
3.El usuario invoca **26 Agregar tipo de tarea a tipo de auditoría**.
4.El sistema guarda el nuevo tipo de auditoría.
5.FIN

Escenario de Excepción:

E1_Los datos requeridos no están completos
4.a Condición: Los campos obligatorios no esta completos.
4.a.1 El sistema informa al usuario cuales son los datos obligatorios que no están completos.
4.a.2 El sistema vuelve al paso 2 para solicitar los campos obligatorios incompletos.

4.a.3 FIN.

E2_Los datos solicitados por el sistema no son válidos.
4.b Condición: Algunos de los campos solicitados no son válidos.
El sistema controla que no existan dos tipos de auditoría con el mismo nombre.
4.b.1 El sistema informa al usuario cuales campos son inválidos.
4.b.2 FIN

CU 26_Agregar tipo de tarea a tipo de auditoría

Objetivo: Configurar un tipo de auditoría
Contexto de uso: Cuando se crea un nuevo tipo de auditoría se le deben agregar las tareas a realizar para cumplir con la auditoría. Este caso también se puede ejecutar al momento de modificar un tipo de auditoría existente.
Descripción resumida: se agrega al tipo de auditoría un nuevo tipo de tarea a realizar para cumplir con una auditoría de este tipo.
Precondiciones:

1. Hay un tipo de auditoría seleccionado.
2. El usuario es administrador.

Poscondiciones:

1.Se agrego un tipo de tarea al tipo de auditoría.

Escenario Principal:

00_Tipo de tarea agregado a la auditoría.

1. El usuario solicita agregar un tipo de tarea al tipo de auditoría.
2. El sistema lista los tipos de tareas existentes que el tipo de auditoría aún no tiene.
3. El usuario selecciona los tipos de tarea a agregar.
4. El sistema agrega al tipo de auditoría los tipos de tareas seleccionados.

5.4 Arquitectura y diseño de alto nivel.

5.4.1 Arquitectura.

La arquitectura de la aplicación está basada en capas. Cuando pensamos un sistema en términos de capas imaginamos un subsistema principal con forma de torta de varias capas, donde cada capa se apoya en la capa inferior. En este esquema las capas superiores hacen uso de servicios definidos en las capas inferiores, pero las capas inferiores no tienen conocimiento de las capas superiores. Cada capa esconde a las superiores las capas inferiores.

Cuando se habla de capas de arquitecturas, a menudo hay una confusión entre los términos capas(layer) y niveles o pisos(tier). A veces son usados como sinónimos, pero para la mayoría de la gente los niveles(tier) implican una separación física. Los sistemas cliente/servidor en general son descritos como sistemas de dos niveles y la separación es física. Usamos el termino capa para enfatizar que no necesariamente tenemos que correr las capas en diferentes máquinas.

Por su naturaleza, ésta es una aplicación cliente/servidor y usamos una arquitectura de cuatro capas primarias , la capa de presentación, la capa de servicios, la capa de dominio y la capa de fuente de datos. La capa de presentación provee la comunicación con el usuario permitiéndole invocar los servicios y mostrándole la información del resultado. La capa de servicios define la forma y el medio de comunicación del cliente con la lógica de dominio. La capa de dominio se encargará de la lógica del sistema y la capa de fuente de datos, se encarga de la comunicación con la base de datos.

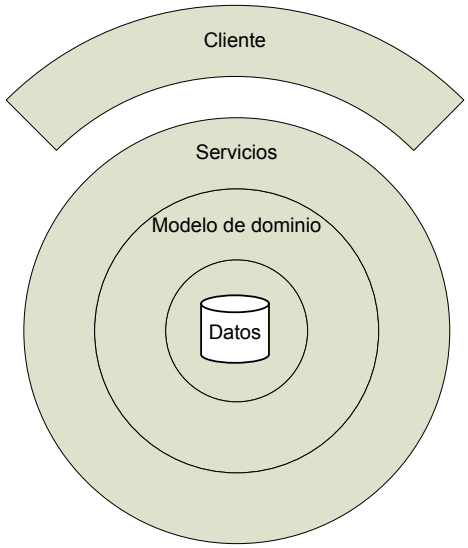


Figura 4: Modelo de las capas de arquitectura

Como ya mencionamos vamos a tener una arquitectura orientada a servicios (SOA), definiendo una fachada del servidor.

5.4.1.1 Tecnologías seleccionadas para implementar la arquitectura

Para dar soporte a la arquitectura diseñada, utilizamos en las diferentes capas una serie de tecnologías que a continuación detallamos.

Introducción

A la hora de definir la arquitectura del sistema decidimos hacer una aplicación WEB en FLEX con java y motor de persistencia hibernate. Tenemos un modelo de objetos java, una fachada del servidor y comunicación java- flex / flex-java con objetos DTO, para dar soporte a esta comunicación utilizamos BlazeDS. Para la fachada de servicios utilizamos como middleware Spring. La base de datos relacional utilizada es MySQL 5 y como servidor elegimos usar Tomcat 6.
Lo novedoso de las tecnologías usadas es uno de los puntos fuertes de este sistema. Todas ellas son tecnologías *open source*.

Adobe Flex 3

Para la capa de presentación decidimos realizar el cliente de esta aplicación con la herramienta Flex 3.
La plataforma incluye:

- Un lenguaje basado en XML llamado MXML que soporta la programación declarativa de componentes GUI para captar a los diseñadores.
- El lenguaje orientado a objetos standard, ActionScript 3.0, basado en la última especificación ECMAScript.
- Integración con el lado del servidor vía Flex Data Services (FDS) dando a las aplicaciones cliente acceso transparente al mundo J2EE.
- Componentes de charting, acceso a controles multimedia, etc.
- Un IDE basado en Eclipse con despliegue automatizado, debugging, y facilidades de traza.

La plataforma Flex 3 es fácilmente extensible y se integra bien con la parte del servidor, ya sea Java, ColdFusion, PHP, Ruby, ASP, o parecidos.
Pueden hacerse aplicaciones Flex más avanzadas incluyendo interacción con sistemas de la parte del servidor a través de Flex Data Services, que proveen acceso remoto a los objetos Java y componentes Java EE del lado del servidor, soporte de mensajería

extenso (incluyendo integración JMS), sincronización con información persistida, la integración con otras tecnologías persistentes.

Java

Optada ya la tecnología para la capa de presentación, y en base a la misma, parece lógico y natural inclinarse por tecnologías Java para la capa de Modelo. Es bien conocida la buena sintonía entre Flex y Java y su uso conjunto está cada vez más extendido en el ámbito empresarial.
Además de ser el lenguaje de programación mas popular hoy en día, java es libre y es el que mas dominamos por estar trabajando diariamente con él. También posee una integración optima, a parte de la que tiene con Flex , con las demás tecnologías elegidas para el desarrollo del producto.

BlazeDS

La capa de servicios la conectamos a Flex usando los llamados *Data Services*. Estos servicios son un componente del lado del servidor que traduce las peticiones Flex a una tecnología servidor – en nuestro caso a Java – y las respuestas de estos servicios los traduce de vuelta a un formato binario de Adobe (AMF) de esta forma para Flex es transparente lo que se esté usando en el servidor.

Adobe comercializa una versión de los Data Services llamada Adobe LiveCycle(LCDS), además existen alternativas opensource como GraniteDS. Sin embargo, junto con Flex 3.0 se liberó BlazeDS: la versión opensource y gratuita de LiveCycle y la que usaremos en nuestra tesis. Al ser gratuita no tiene todas las características de la versión que se comercializa (LCDS), pero creemos que para el alcance de esta tesis cubre los requerimientos. BlazeDS no cuenta con el módulo de Data Management que provee sincronización de datos en tiempo real y carga lazy o paginado.

BlazeDS se despliega como un conjunto de archivos JAR como parte de la aplicación Web ejecutándose en un servidor de aplicaciones Java EE o en un contenedor de servlets.

BlazeDS consiste en tres servicios clave:

- El Servicio Remoto permite a las aplicaciones Flex invocar directamente métodos de objetos Java desplegados en el servidor de aplicación. Esto es lo que utilizamos en nuestra aplicación.
- El Servicio de Mensajes provee una infraestructura publicar/suscribir que permite a las aplicaciones Flex publicar mensajes y suscribirse a un destino de mensajes, permitiendo el desarrollo de data push en tiempo real así como de aplicaciones colaborativas.
- El Servicio Proxy permite a las aplicaciones Flex hacer peticiones de servicio cross-domain de forma segura y controlada. En otras palabras, permite a las aplicaciones Flex acceder a un servicio disponible en un dominio diferente al dominio de donde la aplicación fue descargada (sin tener que desplegar un archivo con política cross-domain.xml sobre el dominio en cuestión).

BlazeDS es la capa intermedia que actúa a modo de *pegamento* entre los clientes Flex y los sistemas de backend.

Spring

Para el backend del sistema utilizamos el Framework Spring. Básicamente es un contenedor liviano de objetos que soporta inversión de control y aspectos (Spring in action [26]), sirve para múltiples propósitos y para esta aplicación aprovechamos las siguientes característica:

- Su potente gestión de configuración basada en JavaBeans, aplicando los principios de Inversión de Control (*IoC*). Esto hace que la configuración de aplicaciones sea rápida y sencilla. Ya no es necesario tener *singletons* ni archivos de configuración, una aproximación consistente y elegante. Esta factoría de *beans* puede ser usada en cualquier entorno, desde *applets* hasta contenedores J2EE. Estas definiciones de *beans* se realizan en lo que se llama el contexto de aplicación.

- Una capa genérica de abstracción para la gestión de transacciones, permitiendo gestores de transacción enchufables (*pluggables*), y haciendo sencilla la demarcación de transacciones sin tratarlas a bajo nivel. Se incluyen estrategias genéricas para JTA y un único JDBC DataSource. En contraste con el JTA simple o EJB CMT, el soporte de transacciones de Spring no está atado a entornos J2EE.
- Integración con Hibernate, JDO e iBatis SQL Maps en términos de soporte a implementaciones DAO y estrategias con transacciones. Especial soporte a Hibernate añadiendo convenientes características de *IoC*, y solucionando muchos de los comunes problemas de integración de Hibernate. Todo ello cumpliendo con las transacciones genéricas de Spring y la jerarquía de excepciones DAO.

Acegi

Para la autenticación y autorización del sistema, utilizamos el Framework Acegi (actualmente llamado Spring Security) y sus principales características son:

- Facilidad de uso.
- Integración completa en Spring, utiliza los mecanismos de configuración de Spring. Configuración mediante el contexto de aplicación de Spring o basada en atributos.
- Seguridad a nivel de instancia de objetos del dominio.
- En muchas aplicaciones es deseable definir listas de control de acceso (Access Control Lists o ACLs) para instancias de objetos del dominio individuales. Proporciona un completo paquete ACL con características que incluyen máscaras de bits, herencia de permisos, un repositorio utilizando JDBC, caché y un diseño pluggable utilizando interfaces.
- Configuración no intrusiva, mantiene los objetos libres de código de seguridad.
- La totalidad del sistema de seguridad puede funcionar en una aplicación web utilizando los filtros que proporciona. No hay necesidad de hacer cambios especiales o añadir librerías al contenedor de Servlets o EJB.
- Integración opcional con los contenedores

- Las características de autenticación y autorización que proporcionan los contenedores Servlet o EJB pueden ser usadas utilizando los adaptadores que se incluyen. Actualmente existe soporte para los principales contenedores: Catalina (Tomcat), Jetty, JBoss y Resin
- Muchas aplicaciones necesitan proteger datos a nivel de objeto basándose en cualquier combinación de parámetros (usuario, hora del día, autoridad del usuario, método que es llamado, parámetros del método invocado etc.). Acegi da esta flexibilidad sin necesidad de añadir código a los objetos de negocio.
- Protección de peticiones HTTP. Además de proteger los objetos, el proyecto también permite proteger las peticiones HTTP. Ya no es necesario depender de restricciones de seguridad definidas en el archivo web.xml. Lo mejor de todo es que las peticiones HTTP pueden ser protegidas por una serie de expresiones regulares o expresiones de paths como las utilizadas por Ant, así como autenticación, autorización y gestores de reemplazo de credenciales para la ejecución como otro usuario, todo ello totalmente pluggable.
- El sistema de seguridad puede redirigir automáticamente las peticiones a un canal de transmisión adecuado. Comúnmente esto se aplica para asegurar que las páginas seguras estarán sólo disponibles sobre HTTPS, y las páginas públicas sobre HTTP, aunque es suficientemente flexible para soportar cualquier tipo de requisitos de "canal". También soporta combinaciones de puertos no usuales y gestores de decisión de transporte pluggables.
- Esta autenticación es adecuada para aquellas aplicaciones que prefieren una simple ventana de login del navegador en lugar de un formulario de login. Acegi Security puede procesar directamente peticiones de autenticación HTTP BASIC siguiendo el RFC 1945.
- Librería de etiquetas. Proporciona una librería de etiquetas que puede ser utilizada en JSPs para garantizar que contenido protegido como enlaces y mensajes son únicamente mostrados a usuarios que poseen los permisos adecuados.
- Permite seleccionar el método utilizado para la configuración de seguridad del entorno, tanto a través del contexto de aplicación de Spring o utilizando atributos en el código fuente utilizando Jakarta Commons Attributes.
- Distintos métodos de almacenamiento de la información de autenticación

- Acegi incluye la posibilidad de obtener los usuarios y permisos utilizando archivos XML, fuentes de datos JDBC o implementando un interfaz DAO para obtener la información de cualquier otro lugar.
- Soporte para eventos. Utilizando los servicios para eventos que ofrece Spring, se pueden configurar receptores propios para eventos como login, contraseña incorrecta y cuenta deshabilitada. Esto permite la implementación de sistemas de auditoría y bloqueo de cuentas, totalmente desacoplados del código de Acegi Security.
- Utilizando EHCache u otra implementación propia se puede hacer caché de la información de autenticación, evitando que la base de datos o cualquier otro tipo de fuente de información no sea consultado repetidamente.

Hibernate

Como mapeador objeto/relacional usamos el Framework Hibernate, el cual es el ORM(Object relacional mapping) más popular en el mundo Java.

Estas herramientas hacen la programación más fácil y permiten una adhesión más cercana a los ideales de la programación orientada a objetos y a un desarrollo arquitectónico multi-capa.

Las principales características de Hibernate son:

- Persistencia transparente.
- Hibernate puede operar proporcionando persistencia de una manera transparente para el desarrollador.
- Modelo de programación natural.
- Hibernate soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el framework de colecciones de Java.
- Soporte para modelos de objetos con una granularidad muy fina.
- Permite una gran variedad de mapeos para colecciones y objetos dependientes.
- Sin necesidad de mejorar el código compilado (bytecode).
- No es necesaria la generación de código ni el procesamiento del bytecode en el proceso de compilación.
- Escalabilidad extrema.

- Hibernate posee un alto rendimiento, tiene una caché de dos niveles y puede ser usado en un cluster. Permite inicialización perezosa (lazy) de objetos y colecciones.
- Lenguaje de consultas HQL
- Este lenguaje proporciona una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.
- Soporte para transacciones de aplicación
- Hibernate soporta transacciones largas (aquellas que requieren la interacción con el usuario durante su ejecución) y gestiona la política optimistic locking automáticamente.
- Generación automática de claves primarias.
- Soporta los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos (secuencias, columnas auto incrementales) así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas.

Base de datos

En este punto se optó directamente por una base de datos libre tan consolidada como MySQL por su rapidez, interoperabilidad y estabilidad. Para el desarrollo utilizamos la versión 5. MySQL funciona sobre múltiples plataformas.

Servidor

Para soportar esta arquitectura puede usarse en cualquier servidor J2EE, pero para la funcionalidad de la herramienta producto de esta tesis nos alcanza con un contenedor de servlets, por lo que utilizaremos Tomcat 6.
Tomcat 6 es un servidor Java EE open source ampliamente utilizado.

Resumiendo

Fue un factor importante a la hora de elegir las tecnologías para el desarrollo el hecho de que muchas de ellas son libres y son las que utilizamos a diario en nuestros trabajos a excepción de Flex 3. Gráficamente la arquitectura de nuestro sistema cliente/servidor es la siguiente:

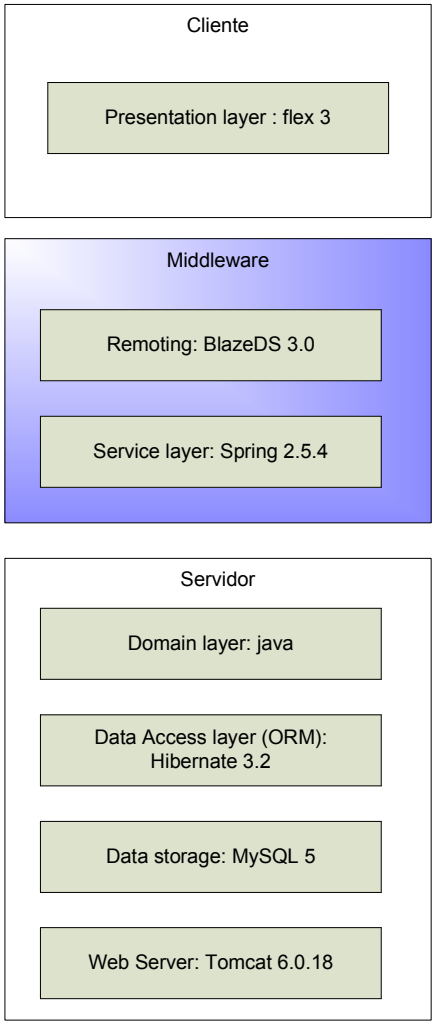


Figura 5:*Arquitectura del sistema.*

5.4.2 Diseño de alto nivel

Una vez establecida la arquitectura, hablaremos del diseño de alto nivel mostrando los distintos módulos o subsistemas de la aplicación y la interacción entre ellos.

Podemos tener varias vistas de la aplicación, una ella es la vista de módulos. Un módulo lo podemos ver como un agrupamiento lógico no necesariamente físico, de cierta funcionalidad que le da significado al mismo. Entre los módulos de este sistema se encuentran: el módulo de Seguridad, el módulo de manejo de la agenda, el módulo de gestión de auditorías, el módulo de gestión de los proyectos, el módulo de reportes,

módulo de configuración, el módulo AssemblerDTO, el módulo de Persistencia. Gráficamente:

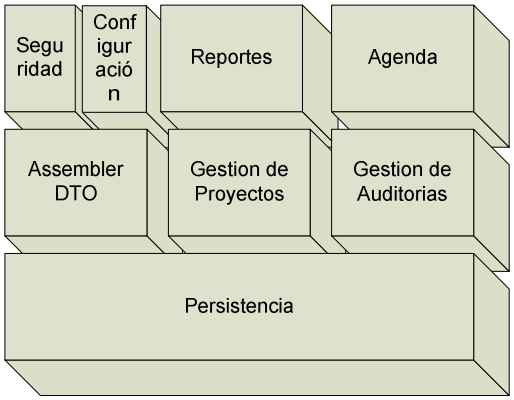


Figura 6: Vista de los módulos

- **El módulo de Persistencia:** este módulo se encarga de encapsular la interacción con hibernate , el Framework de persistencia que estamos usando. Su función es hacer transparente para la aplicación el ORM(Object Relational Mapping) e independizarnos de la base de datos.
- **El módulo AssemblerDTO:** este módulo se encarga de realizar la transformación de los objetos de dominios a DTO´s(Data transfer object) y viceversa, los cuales son utilizados para que puedan viajar al cliente, y el cliente enviarlos al servidor. En la próxima sección se explicarán los DTO con más detalle.
- **El módulo de Gestión de Proyectos:** este módulo se encarga del manejo y seguimiento de los proyectos, por ejemplo este módulo en colaboración con el de gestión de auditorías se encargaran de sugerir las próximas auditorías para un determinado proyecto. También se encarga del ABM de proyectos.
- **El módulo de Gestión de Auditorías:** se encarga del manejo de las auditorías realizadas sobre los proyectos. Se encarga también de controlar el estado de las tareas de las auditorías, del seguimiento y escala de no conformidades.
- **El módulo Agenda:** este módulo se encarga del manejo de la agenda que permite agendar las auditorías en el tiempo, controlar el vencimiento de las mismas y de las no conformidades.
- **El módulo de Reportes:** este módulo permitirá imprimir reportes sobre las auditorías pendientes de los proyectos y exportarlos a diferentes formatos.

También se van a poder sacar informes de las no conformidades registradas para un proyecto o de un grupo de proyectos.

- **El módulo de Seguridad:** se encargará de la autenticación y autorización de los usuarios. También manejará los ABM y demás operaciones sobre los usuarios.
- **El módulo de Configuración:** permitirá al administrador del sistema instanciar o desplegar la aplicación antes de que se pueda operar con ella.

5.5 Detalles de implementación

Introducción

En esta sección hablaremos de los detalles de implementación que consideramos más relevantes. También presentaremos el diagrama de clases surgido del diseño realizado para soportar los requerimientos, junto a una breve explicación del modelo.

Servicios.

Al tener una arquitectura SOA, la comunicación del cliente con el servidor es a través de servicios. En los servicios se han tratado de incluir las funciones de negocio.
Los hemos implementados como beans de Spring para que puedan ser invocados remotamente. Nuestros servicios no tienen estado, son stateless. Sus métodos son transaccionales.
Tratamos de agrupar los servicios en función de la entidad sobre la que actúan. Nosotros llamamos servicio a un método de un clase. La clase implementará una serie de métodos que operan sobre una misma entidad. Por ejemplo la clase ProyectoServiceImpl.java en la que se definen los métodos para realizar el ABM de proyectos.
Hemos implementado dos capas de servicios una remota (RemoteFacade, Fowler[17]) y otra local, la local trabaja sobre objetos de dominio y la remota sobre DTO´s(Data transfer object) de los mismos.
El tener un capa remota, nos permite ubicar en ella comportamiento adicional, por ejemplo , la creación de DTO´s(Data transfer object).

DTO

Flex al utilizar su propio algoritmo de serialización/deserialización (AMF), por problemas compatibilidad con hibernate, nos imposibilitaba poder utilizar objetos detachados para trabajar con ellos en el cliente. Como solución utilizamos DTO´s (Data transfer object) de los objetos de dominio, Fowler[17].

Al utilizar DTO en la comunicación con el servicio, tuvimos que implementar una clase que se encargue de la transformación de objetos de dominio a DTO y viceversa, para esto implementamos la clase AssemblerDTO.java. Por ejemplo para la clase proyecto, cuando cliente solicita listar los proyectos la secuencia es la siguiente:

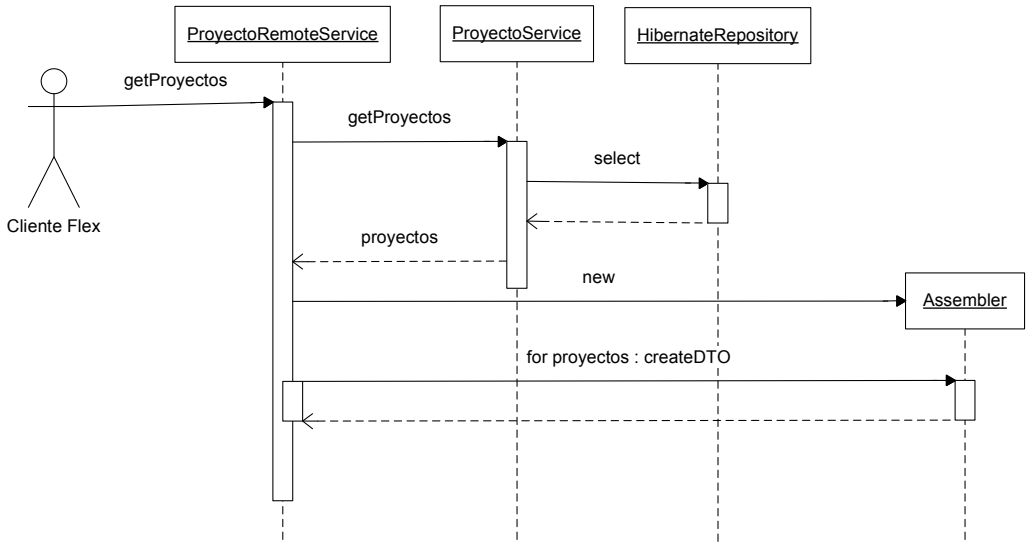


Figura 7: Diagrama de secuencia de la obtención de los proyectos.

De esta manera el cliente puede trabajar con los DTO´s. Cuando el cliente modifica el DTO, este se envía al servicio para que se pueda recuperar el objeto de dominio correspondiente y realizarle las modificaciones pertinentes.

Capa de persistencia.

Al tener un modelo orientado a objetos y una base de datos relacional necesitamos unir estos dos mundos con un ORM (Object Relational Mapping), por lo cual elegimos hibernate por su facilidad de uso y madurez.
Para encapsular la interacción con hibernate y que la lógica para la persistencia de datos no quede mezclada con la lógica del negocio implementamos una capa de persistencia representada a través de la clase HibernateRepository, de esta manera cuando la capa de lógica de negocio necesita interactuar con la base de datos, ésta utilizará los métodos ofrecidos por esta clase. Generalmente la clase de operaciones que ofrece la capa de datos se le conoce como CRUD (Create, Read, Update y Delete).

Hasta aquí los detalles de implementación mas importantes. Ahora veremos el modelo de objetos que soportan el sistema.

Diagrama de clase

A continuación exponemos el modelo de clases persistentes o de dominio que utiliza la aplicación que vamos a desarrollar.

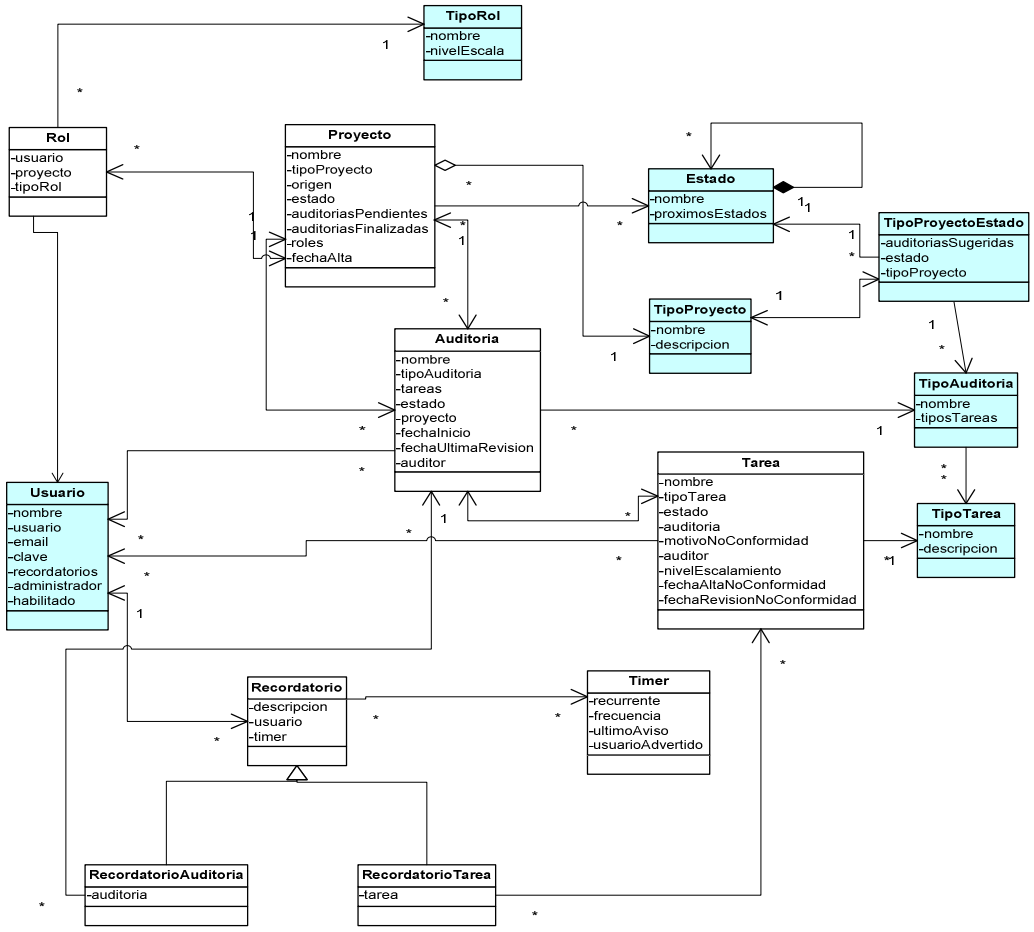


Figura 8: Diagrama de clases del sistema

Un proyecto es la unidad de trabajo del sistema, un proyecto además de los datos básicos que lo identifican, a lo largo de su ciclo de vida puede pasar por diferentes estados. Naturalmente hay proyectos de diversos tipos, el tipo de un proyecto junto con el estado en el que se encuentra en un momento dado permiten sugerir posibles auditorías a realizar. A medida que el proyecto comienza a transcurrir se le van asociando auditorías para realizarle, estas auditorías las llamamos pendientes. Cuando se realiza una auditoría sobre un proyecto y ésta es exitosa, se pasa a la lista de auditorías finalizadas el proyecto.

Los diferentes tipos de auditorías que se pueden realizar sobre un proyecto están conformadas por distintos tipos de tareas que se deben cumplir para poder dar por finalizadas las auditorías.

Inicialmente, cuando se asocia una auditoría a un proyecto, la auditoría y todas las tareas que la conforman se encuentran en estado "No iniciada". Al momento que se esta auditando un proyecto y nos encontramos con que cierta tarea no se cumple ésta pasa al estado "Abierta", si la tarea se da por cumplida pasa al estado "Cerrada". Si la auditoría tiene alguna de sus tareas en estado "Abierta" entonces su estado es "Abierta". La auditoría pasa a estado "Cerrada" cuando todas sus tareas están cerradas, en este momento se da por finalizada la auditoría.

La agenda que provee el sistema permite a los usuarios agendarse auditorías o no conformidades a realizar o revisar sobre los distintos proyectos de manera que a su debido tiempo se les recuerde. Los recordatorios pueden ser sobre auditorías o no conformidades y se puede configurar la frecuencia del aviso, es decir, si la alarma es recurrente, cada cierta cantidad de días o si solo se quiere que se le recuerde por única vez en una determinada fecha.

En un proyecto participan entre otras, personas de la organización con roles distinguidos dentro de cada proyecto como por ejemplo el líder del proyecto, el gerente etc.

En la configuración del sistema el usuario administrador deberá instanciar un conjunto de clases para que la aplicación quede operativa y pueda comenzar a funcionar. En el diagrama están en color celeste. Estas clases son de configuración, por ejemplo el administrador deberá cargar los usuarios del sistema, configurar los diferentes tipos de auditorías con su conjunto de tareas, además tendrá que definir cuales son los roles que tendrán los proyectos, los diferentes tipos de proyectos y los estados por los que podrán pasar.
Además si desea podrá indicar para cada par tipo de proyecto – estado los tipos de auditorías sugeridas.

5.6 Herramientas de desarrollo

Para el desarrollo de la aplicación utilizamos las siguientes herramientas

Eclipse

Como entorno de desarrollo utilizamos Eclipse, una plataforma de herramientas universal, un entorno integrado de desarrollo (IDE) abierto y extensible, para cualquier utilidad y no utilizamos ninguna extensión en particular.

Pese a que Eclipse esté escrito en su mayor parte en java(salvo el núcleo), se ejecute sobre una JVM (maquina virtual java) y su uso más popular sea como IDE para java, es una herramienta neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, XML, etc.

La característica clave de Eclipse es la extensibilidad. Eclipse es una gran estructura formada por un núcleo y muchos plugins que van conformando la funcionalidad final, así, los nuevos aportes se integran sin dificultad ni conflictos.

Flex builder 3

Para el desarrollo de las interfaces graficas del usuario en Flex, utilizamos el plugin para eclipse Flex builder, esta herramienta nos permite crear con rapidez y sencillez complejas pantallas, simplemente arrastrando y soltando sobre el área de trabajo los diferentes controles.

Maven 2

Para realizar la integración, hacer el deploy en el servidor, mantener un repositorio de librerías utilizamos una herramienta de gestión de información de proyectos llamada Maven(versión 2).

Maven tiene muchos objetivos, pero resumiendo Maven intenta hacer la vida del desarrollador sencilla proporcionando una estructura de proyecto bien definida, unos procesos de desarrollo bien definidos a seguir, y una documentación coherente que mantiene a los desarrolladores y clientes informados sobre lo que ocurre en el proyecto. Maven aligera en gran cantidad lo que la mayoría de desarrolladores consideran trabajo pesado y aburrido y les permite proseguir con las tareas más atractivas (típicamente codificación y modelado). Esto es esencial en proyectos open source donde no hay

mucha gente dedicada a la tarea de documentar y propagar la información crítica sobre el proyecto que es necesaria para atraer potenciales nuevos desarrolladores y clientes.
La ambición de Maven es hacer que el desarrollo interno del proyecto sea altamente manejable con la esperanza de proporcionar más tiempo para el desarrollo entre proyectos. Se puede llamar polinización entre proyectos o compartir el conocimiento sobre el desarrollo del proyecto.

Las principales características de Maven son:

- El modelo de objetos del proyecto POM es la base de cómo Maven trabaja. El desarrollo y gestión del modelo está controlado desde el modelo del proyecto.
- Un único conjunto de métodos son utilizados para todos los proyectos que se gestionan. Ya no hay necesidad de estar al tanto de innumerables sistemas de compilación. Cuando las mejoras se hacen en Maven todos los usuarios se benefician.
- Publicación del sitio web basado en el POM. Una vez que el POM es exacto los desarrolladores pueden publicar fácilmente el contenido del proyecto, incluyendo la documentación personalizada más el amplio conjunto de documentación generada por Maven a partir del código fuente.
- Publicación de distribuciones basada en el POM.
- Maven favorece el uso de un repositorio central de librerías, utilizando un mecanismo que permite descargar automáticamente aquellas necesarias en el proyecto, lo que permite a los usuarios de Maven reutilizar librerías entre proyectos y facilita la comunicación entre los mismos, para asegurar que la compatibilidad entre distintas versiones es correctamente tratada.
- Guías para la correcta disposición de los directorios. Maven contiene documentación sobre como disponer los directorios de forma que una vez que es aprendida se puede ver fácilmente cualquier otro proyecto que use Maven.

MySQL Query Browser , MySQL Administrator

Para crear la base de datos y posteriormente trabajar con ella utilizamos como front end las herramientas MySQL Query Browser y MySQL Administrator que fueron bajadas

gratuitamente del sitio oficial de MySQL.(url [25]) Con estas herramientas nos alcanzo de sobremanera para realizar éste trabajo.

SVN – Tortoise SVN

Actualmente existen dos grandes corrientes en cuanto al control de versiones se refiere: CVS y Subversion.

Por un lado, CVS es el sistema de control de versiones concurrentes más utilizado en el mundo del software libre, sin embargo es bastante arcaico (nació en 1985) y tiene bastantes carencias, como son la falta de versionado de directorios, de renombrado o de copias; su uso poco eficiente de la red o el que no almacene archivos binarios completos, solo los cambios entre versiones.

Subversion es un CVS de código abierto, bajo una licencia del tipo Apache/BSD, que nació como alternativa a CVS y que intenta solucionar estos problemas y carencias.

Subversion es la alternativa de este proyecto para el control de versiones, la herramienta concreta a utilizar será Tortoise SVN. Tortoise SVN es un cliente Subversion, implementado como una extensión al shell de Windows. Es software libre, liberado bajo la licencia GNU GPL.

Capítulo 6 - Caso de estudio

En esta sección vamos a mostrar la funcionalidad de la herramienta desarrollada configurándola para ser usada dentro del área de QA de una fábrica de software.

6.1 Definición del dominio y el alcance

La empresa cuenta entre otras, con una oficina para la gestión de los proyectos (PMO, Project Management Office) y una para el aseguramiento de la calidad (QAO, Quality Assurance Office).

En la empresa un proyecto se inicia cuando un usuario carga una petición mediante el envío de un mail. Una petición es simplemente un pedido de usuario, es el disparador que da origen a un proyecto. Este mail le llega entre otros a la PMO avisándole que hay un nuevo proyecto.

La PMO selecciona un JP y le avisa mediante un mail en el que se especifica el ID de la petición. Luego la petición queda en estado "Pendiente de aceptación".

El JP seleccionado accede a la petición para analizarla y tomar una decisión, si la acepta, la petición queda pendiente de Estimación. Si el JP rechaza la petición, ésta queda en estado Rechazada.

Una vez aceptada la petición, como primera medida, el PMO solicita a la gente de infraestructura vía e-mail la creación del repositorio, copiando a la QAO. El mail contiene la siguiente información: ID de Petición, Código de petición, Descripción y Jefe de proyecto. Mas tarde infraestructura informa sobre la efectiva creación del repositorio respondiendo el e-mail recibido.

Luego de la creación del repositorio se desencadena la primera actividad de control. El equipo de QA, 96 horas posterior a la aceptación de la petición, verifica que se haya creado el repositorio. Además verifica que éste cumpla con las reglas establecidas. Si no encuentra evidencia de la creación del repositorio de Proyecto o no cumple con las reglas, envía un e-mail al área de infraestructura (INF) informando la falta.

Luego de aceptada una petición el JP debe estimarla, para lo que crea en el repositorio el documento de estimación. [IdProyecto]_Estimacion.doc y lo completa. Al terminar la estimación el JP cierra la etapa de estimación, avisando de tal hecho a la PMO y la petición pasa a estar en estado "Pendiente de aceptación de Estimación".

Luego se documentan los requerimientos correspondientes a la petición. El JP crea y completa el documento [IdProyecto]_Requerimientos_de_Proyecto.doc y lo da de alta en el repositorio.

El equipo de QA después de ser notificado del cierre de la etapa de estimación, revisa que se hayan creado el documento [IdProyecto]_Estimación.doc y el documento [IdProyecto]_Requerimiento_de_Proyecto.doc, que los mismo se encuentren en el repositorio creado a tal fin. Si no existe alguno de los documentos, no respetan el template o están incompletos, se envía un e-mail al Jefe de Proyecto, informándolo de la no conformidad.

Luego de que la estimación es aceptada, el JP crea y completa el documento de planificación [IdProyecto]_Plan_de_Proyecto.doc, cierra la etapa de planificación e informa vía e-mail a la PMO y a la QAO que la etapa de planificación ha sido cerrada.

Después de notificado del cierre de etapa de planificación, la QAO revisa que se haya creado el documento [IdProyecto]_Plan_de_Proyecto.doc y revisa que el mismo se encuentre en el repositorio creado a tal fin. Si no existe el documento, no respeta el template o está incompleto, se envía un e-mail al Jefe de Proyecto, informándolo de la no conformidad.

La QAO, a 96 horas posterior a registrar una no conformidad, revisa el estado de la misma. Si no se encuentra evidencia de corrección de la no conformidad se procede a escalar, se envía un e-mail al responsable asignado copiando a un nivel de jerarquía superior informando la falta de solución de la no conformidad.

Si en la revisión encuentra evidencia de corrección de la no conformidad se procede a cerrar la misma.

Para seguir el ciclo de vida del proyecto identificando oportunidades de mejora en la gestión, la PMO genera un informe quincenal con información de gestión de la petición. Este informe de seguimiento se publica en la Wiki y se informa de su publicación vía e-mail a los Jefes de Proyectos y / o Gerentes con copia a la QAO.

Luego el equipo de QA realiza la auditoría de seguimiento de proyecto. Revisa que los proyectos evidencien seguimientos, que se hayan generado los informes en tiempo y forma. Si no evidencia seguimiento carga una no conformidad.

Los primeros días de cada mes una persona asignada por la QAO debe ingresar a revisar las auditorías correspondientes a los proyectos en curso. Por cada proyecto ingresado

analiza la gestión realizada, la cantidad de auditorías que tiene pendientes y las que están finalizadas. Si no evidencia gestión de auditorías eleva una no conformidad.

A medida que avanza el desarrollo del proyecto, para cumplir con los compromisos asumidos se empiezan a entregar versiones de los artefactos según lo estipulado en documento de plan de proyecto, donde figuran las fechas y los artefactos a entregar**.**

Para cada entrega planificada el JP realiza la nota de entrega correspondiente y la sube al repositorio.

Mas tarde, el equipo de QA, en función del documento de planificación realiza auditorías para determinar si se cumplieron las entregas pautadas y si estas tienen la nota de entrega asociada.

Finalmente un día el jefe de proyecto, da por finalizada la petición, dejándola en estado finalizada. Entonces avisa a la PMO y la QAO a través de un email que el proyecto terminó.

Al recibir el email de cierre de proyecto, la QAO identifica en el plan de proyecto los compromisos correspondiente a artefactos entregables y verifica que los mismos se encuentren en el directorio salida, según el documento plan de proyecto.

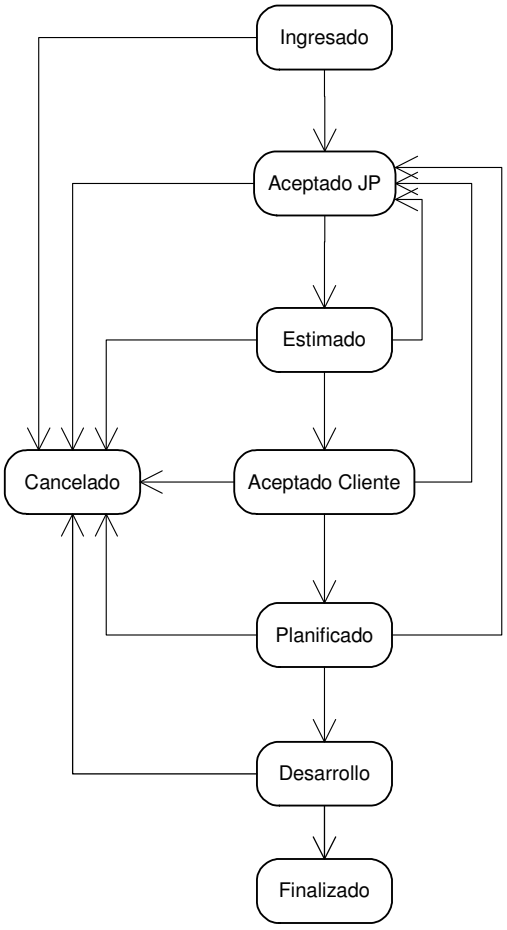
6.2 Aplicación de la herramienta al caso de estudio

Vamos a utilizar la herramienta desarrollada en esta tesis para dar soporte al flujo de trabajo que describimos arriba.

6.2.1 Estados

Entonces los estados por los que puede pasar un proyecto son:

1. Ingresado.
2. Aceptado JP.
3. Estimado.
4. Aceptado Cliente.
5. Planificado.
6. Desarrollo.
7. Finalizado.
8. Cancelado.



Comienzan con estado "*Ingresado"*, cuando lo da de alta la PMO, luego lo toma el jefe de proyecto y si lo acepta lo pasa al estado *"Aceptado JP*".

Una vez que el JP acepta el proyecto, lo debe estimar, finalizada esta tarea queda en estado *"Estimado"*.

Obviamente, el cliente debe aprobar la estimación, si acepta la estimación el proyecto pasa a estado *"Aceptado cliente"*. Si no la acepta, el proyecto puede volver al estado *"Aceptado JP*".

Luego de que el cliente acepta la estimación. El próximo paso es realizar la planificación del desarrollo. Terminado esto, el proyecto se pasa al estado

"Planificado".De aquí comienza el desarrollo y durante este tiempo el proyecto se encuentra en estado *"Desarrollo"*.
En cualquier momento el proyecto se puede cancelar, por lo que de cualquiera de los estados anteriores se puede pasar al estado *"Cancelado"*.

6.2.2 Tipos de Proyectos

Una fábrica de software lleva adelante diferentes tipos de proyectos, para este ejemplo, vamos a suponer que contamos con las siguientes clases de proyectos:

1. Mantenimiento.
2. Nuevo desarrollo.
3. Capacitación.
4. Consultaría.

El tipo de un proyecto junto con el estado en el que se encuentra puede determinar los tipos de controles a realizar sobre el mismo, es decir, las auditorías a aplicar.
Una posible sugerencia, por ejemplo, es que cuando un proyecto esta en estado estimado se deben realizar la auditorías de estimación .

6.2.3 Tipos de Auditorías y sus tipos de tareas

El realizar una auditoría sobre un proyecto implica verificar que se cumplan ciertas condiciones, una condición no cumplida es un aspecto de no conformidad. Por ejemplo para la auditoría de estimación tenemos los aspectos:

- Falta el documento.
- El documento está vacío.
- El documento no cumple con el template.
- No figura el usuario aprobador.

Si al realizar una auditoría se encuentran aspectos de no conformidad, se genera una no conformidad y se avisa a quien corresponda.

A continuación vamos a enumerar los distintos tipos de auditorías que se pueden realizar sobre los proyectos. Para cada una de ellas enumeramos la lista de tareas a verificar para dar por terminada la auditoría.

1. **Auditoría de creación del repositorio.**
   a. Verificar existencia de la carpeta correspondiente al nuevo proyecto en el repositorio.
   b. Verificar que el nombre de la carpeta respeta la nomenclatura (PRO_[NRO], donde el [NRO] representa al ID del proyecto.).
   c. Verificar que la carpeta contenga los directorios \Otros y \Salida.

2. **Auditoría de estimación.**
   a. Verificar existencia del documento.
   b. Verificar que el nombre del documento respeta la nomenclatura ([IdProyecto]_Estimación.doc).
   c. Verificar que el documento se encuentre en el repositorio creado para tal fin.
   d. Verificar que el documento no se encuentre vacío.
   e. Verificar que se haya completado el apartado Control de cambios con versión del documento, cambios realizados y actor que lo realizó.
   f. Verificar que se hayan eliminado las instrucciones de cómo completar el documento.
   g. Verificar que estén completos todos los apartados.

3. **Auditoría de requerimientos del proyecto.**
   a. Verificar existencia del documento.
   b. Verificar que el nombre del documento respeta la nomenclatura
   c. Verificar que el documento no se encuentre vacío.
   d. Verificar que se haya completado el apartado objetivo del documento.
   e. Verificar que se hayan eliminado las instrucciones de cómo completar el documento.

4. **Auditoría de planificación.**

   a. Verificar existencia del documento.

   b. Verificar que el nombre del documento respeta la nomenclatura ([IdProyecto]_Plan_de_Proyecto.doc).

   c. Verificar que el documento no se encuentre vacío.

   d. Verificar que se haya completado el apartado Control de cambios con versión del documento, cambios realizados y actor que lo realizó.

   e. Verificar que se haya completado el apartado alcance del documento.

   f. Verificar que se hayan eliminado las instrucciones de cómo completar el documento.

   g. Verificar que este identificado el equipo del proyecto (roles y personas).

5. **Auditoría de seguimiento de un proyecto.**

   a. Verificar que se hayan generado los informes en tiempo y forma.

   b. Verificar que los informes representativos del seguimiento contengan referencias a los proyectos en gestión.

6. **Auditorías de entregas.**

   a. Ingresar al repositorio del proyecto y verifica que en el directorio salida existan los artefactos comprometidos a entregar según documento plan de proyecto.

7. **Auditoría de auditorías.**

   a. Verificar que la cantidad de auditorías corresponda al tiempo de vida del Proyecto en curso.

   b. Verificar el estado de gestión de las no conformidades.

   c. Verificar la existencia del informe mensual.

8. **Auditoría de cierre de proyecto.**

   a. Verificar que se hayan realizado todas las entregas estipuladas en el plan de proyecto.

6.2.4 Roles y Niveles de escalamiento de no conformidad

Un proyecto posee un conjunto de roles distinguidos, el jefe de proyecto, el Project Management, el auditor o analista de QA, el configurador y el gerente. Estos roles determinan los niveles de escalamiento de las no conformidades.

Cuando el auditor registra una no conformidad, en primera instancia se le avisa al jefe de proyecto. Si la próxima vez que se realiza la auditoría, aún continua la no conformidad, entonces ésta se escala, y se le avisa al PM.

El orden de escalamiento es configurable. Para el ejemplo, vamos a suponer el siguiente orden, en primera instancia se va a notificar al jefe del proyecto, luego al Project manager y en última instancia al gerente.

Capítulo 7 – Conclusiones.

7.1 Sobre el trabajo

En esta sección se listarán una serie de tareas realizadas en este trabajo junto con un resumen del aporte realizado durante la elaboración del mismo.

Como primera tarea, se realizó una investigación sobre CMMI, desde los conceptos más básicos, hasta los más complejos, viendo que con muchos de estos estábamos familiarizados y los aplicábamos diariamente, como por ejemplo cumplir un proceso.

Al profundizar sobre los temas generales de CMMI, nos enfocamos específicamente en el nivel 2, cuales son las áreas participantes, los requisitos necesarios para certificar y a partir de esto ajustar y acotar la definición de la herramienta que planeamos desarrollar.

Luego de entender las necesidades de la certificación, nos centramos en el área de PPQA, ésta fue nuestra puerta de entrada a SQA, para conocer los requerimientos del área y lo que ellos implicaban. Esta actividad nos permitió básicamente aprender con cierto nivel de profundidad el modelo CMMI-SW, sobre todo en la que nos centramos, PPQA.

De aquí sacamos los primeros capítulos de la tesis. Y nos permitió definir el objetivo de nuestro sistema.

Una consecuencia lógica de la investigación previa fue estudiar los requisitos para alcanzar el nivel 2 de CMMI, para esto estudiamos los distintos métodos de evaluación SCAMPI y el rol del certificador.

Paralelamente analizamos las posibles tecnologías que utilizaríamos para desarrollar el sistema, focalizándonos al seleccionar las mismas, en que sean open source, y por otro lado que el producto final sea lo mas amigable y fácil de utilizar por el usuario. Por este motivo uno de los grandes desafíos del trabajo de grado fue utilizar Flex para desarrollar un cliente RIA (*Rich Internet Applications*).

En el servidor decidimos utilizar la plataforma java, utilizando como middleware Spring y Spring Security para la autenticación. La comunicación de Spring con el cliente Flex nos las proveyó el framework open source de adobe BlazeDS. Para la persistencia utilizamos el framework hibernate y como motor de base de datos optamos por MySQL 5.

Como mencionamos en el capítulo 5, en la definición del objetivo del sistema desarrollado, nos hemos centrado solo en uno de todos los escalones a subir que implica certificar CMMI-nivel 2, cumplir con los requerimientos del área PPQA. Esto nos permitió darle un contexto a nuestra tesis y acotar el alcance del sistema.
Una vez que teníamos decididas las tecnologías a utilizar nos centramos en los requerimientos del área de PPQA, lo que nos permitió fácilmente advertir cual era la funcionalidad mínima que debería proporcionar nuestro sistema. A partir de acá, definimos los requerimientos y los casos de uso.
Las principales funcionalidades que aporta el sistema son:

- Realizar la configuración general para el seguimiento y control de cumplimiento de las auditorías.
- Poder crear un proyecto a partir de una plantilla predefinida.
- Agregar y agendarse auditorías.
- Verificar si las tareas de una auditoría se cumplen.
- Generar no conformidades
- Escalar no conformidades.
- Recordar al usuario de auditorías y no conformidades agendadas.

7.2 Generales

De la etapa de investigación de CMMI podemos concluir que certificar CMMI es un tema que desde hace relativamente poco tiempo esta en boga en Argentina. Esto ya lo vivió Chile un poco antes que nosotros y hace años fue la fiebre india, polaca etc. En la Argentina, las empresas en general tienen en la mira el nivel 2. Muchas de ellas por un tema puramente comercial. En la industria cada vez más se están pidiendo o prefiriendo empresas con CMMI (y no se aclara el nivel, pero se sobreentiende 2 o 3).

En el contexto de Argentina, hoy por hoy, estar reconocido como nivel 2 es un gran prestigio, un diferenciador y una ventaja competitiva. Pero no perdamos de vista que ser nivel 2 , por ejemplo en India es lo mismo que saber Visual Basic 1.0, debido al

progreso en temas de mejora que tiene el país. La mejora a nivel nacional trae un "rating inflation", que empuja los niveles de madurez hacia arriba. Acá esto recién comienza.

Lamentablemente son contadas las empresas que inician un proceso CMMI con un genuino interés por la mejora interna. En general, es una necesidad de exposición, de no perder competitividad o un requerimiento de un cliente o del mercado.

Las empresas pequeñas que están encarando estos procesos los encaran solas, con algo de asesoría externa y la herramienta construida en esta tesis podría ser de gran ayuda.

Para alcanzar el nivel de madurez 2 de CMMI se requieren grandes cambios en la empresa, los más importantes son en la gerencia de la misma. De hecho, esto es lo que muchas veces impide que las empresas lleguen a Nivel 2. La gestión de la calidad es responsabilidad de todos los niveles ejecutivos de una organización, pero debe estar guiada por la alta dirección. Su realización involucra a todos los miembros de la organización.

Para una empresa alcanzar un nivel de reconocimiento de CMMI, aunque sea el nivel 2, es un cambio cultural que implica contar con procesos, calidad de esos procesos, métricas, controles. Todo esto es visto, en principio y erróneamente, como un gasto que no genera beneficios.

7.3 Personales

A medida que avanzaba el desarrollo de la tesis apareció la duda de si el título que habíamos elegido para la misma, “Sistema para auditar el cumplimiento de CMMI-SW nivel 2”, era demasiado ambicioso. Luego de haber madurado la idea originaria, y habiendo decidido centrarnos en el área de soporte PPQA, quizás el talle del título que mejor le hubiera quedado a esta tesis es “Sistema para auditar el cumplimiento de los requerimientos de PPQA para CMMI-SW nivel 2”. Sin embargo, mirándolo desde otro punto de vista, más general, el de una persona que no conoce CMMI o que tiene una vaga idea del mismo y que no sabe lo que significa PPQA y que se encuentra con esta tesis, nos pareció apropiado ofrecerle el primer título.

La experiencia de haber realizado esta tesis nos ha brindado la posibilidad de desarrollar un proyecto de principio a fin. Desde la elección de las herramientas hasta su diseño interno y externo. Si bien ya habíamos trabajado con la gran mayoría de estas tecnologías, el proyecto nos ha permitido profundizar nuestro conocimiento en su uso y configuración. Nuestra mayor apuesta personal en cuanto a herramientas ha sido la de usar Flex para la capa de presentación, herramienta que nunca habíamos usado antes. Su aprendizaje y familiarización, e integración con las demás tecnologías usadas ha ocupado la mayor parte del tiempo durante el desarrollo del sistema.

También nos permitió conocer el modelo CMMI-SW, sus distintas áreas, sobre todo en la que nos centramos, PPQA.
Aportamos nuestro granito de arena, proveyendo una herramienta para mitigar los problemas mas comunes que tienen las organizaciones en las evaluaciones para alcanzar el nivel 2.

Podemos decir que hemos alcanzado el objetivo planteado para el sistema, creemos que nuestra modesta herramienta cumple con el objetivo delineado de ayudar a certificar CMMI-nivel 2, proveyendo un medio para cumplir parte de las metas que permiten incrementar la madurez de la organización en el desarrollo de software conforme a CMMI-SW. Al trabajar sobre una parte (PPQA) del todo requerido para certificar CMMI- nivel 2, el producto de esta tesis no es suficiente pero es de gran ayuda para alcanzar el nivel 2. Ya que si bien PPQA es un área de procesos, es cross a todas las demás áreas. Porque controlar y dejar registro de las no conformidades se debe hacer en todas las áreas. Esto en CMMI-SW se llama una práctica genérica.
A partir de los conocimientos adquiridos durante la realización del trabajo, creemos estar capacitados para dar soporte al área de QA de una empresa, participar en la definición de procesos y configurar la herramienta para que sea utilizada por las personas que participan en esta área y por la persona que certifica CMMI.
Para finalizar, creemos que lo que distingue esta herramienta de las que se encuentran en el mercado, es la posibilidad que provee de agendarse auditorías y no conformidades, poder escalar una no conformidad por el no cumplimiento de la misma, enviando un mail a un superior automáticamente, y lo que pensamos que es de mucha ayuda para quien desempeña estas tareas es el panel de recordatorios.

Capítulo 8 - Trabajos futuros

Los siguientes temas son aquellos en los que creemos que pueden existir puntos de interés para la ampliación de este trabajo.

Como venimos repitiendo en los capítulos precedentes, nos centramos en los requerimientos del área de PPQA para el nivel 2 de CMMI, esto deja la puerta abierta para desarrollar herramientas que complementen a la desarrollada, centrándose en las otras áreas involucradas en la certificación para éste nivel.

También se podrían desarrollar herramientas de soporte para guiar a los desarrolladores en el proceso definido por la organización, por ejemplo, que le ayude a éstos a crear los distintos artefactos respetando la nomenclatura definida por el equipo de ingeniería.

Por estar restringidos en tiempo, nos quedo funcionalidad afuera de esta primera versión del sistema. La herramienta se podría extender de la siguiente manera:

Extensiones futuras a la herramienta:

- Automatizar la ejecución de ciertas tareas. Por ejemplo aquellas que requieren chequear la existencia de algún documento.
- Reportes. Poder obtener más reportes de interés para los certificadores y exportación de éstos a otros formatos, por ejemplo Excel.
- Carga de proyectos desde una fuente externa al sistema, por ejemplo desde un xml, vía email etc.
- Poder cargar la configuración inicial del sistema a partir de un archivo XML, esto es, que se puedan especificar los tipos de proyectos, tipos de auditorías , tipos de tareas y las relaciones entre ellos en un archivo XML.
- Copiar estructura de auditorías entre dos proyectos del mismo tipo. Para poder satisfacer requerimientos del estilo “A éste proyecto quiero realizarle todas las auditorías que le hice a éste otro.”
- Crear un proyecto a partir de la copia de otro. Clonar un proyecto. Esta extensión incluye al punto anterior.

Capítulo 9 - Apéndice

9.1 Definición de Casos de uso

9.1.1 *Qué es un caso de uso*

Introducción

Los casos de uso modelan comportamiento, interacción. No tiene sentido usarlos si lo que se quiere modelar no es comportamiento interactivo.
El caso de uso comprende los pasos necesarios para alcanzar un objetivo de su actor principal. Debe proveer una especificación funcional completa, independiente de la tecnología.
El caso de uso no describe el procesamiento interno del sistema, sólo la interacción y los resultados de valor para el usuario.

Resumiendo un caso de uso es una interacción típica entre un usuario y un sistema de cómputo.

- El caso de uso capta alguna función visible para el usuario.
- El caso de uso puede ser pequeño o grande.
- El caso de uso logra un objetivo discreto para el usuario.

Una buena fuente para identificar los casos de uso son los eventos externos. Pensar en todos los eventos externos ante los cuales se quiere reaccionar.

Actores

Los actores o roles del sistema se obtienen normalmente en la etapa de relevamiento. Los actores llevan a cabo los casos de uso. Un mismo actor puede realizar muchos casos de uso.
Un actor primario es quien tiene la meta que cumple el caso de uso. Un actor secundario es quien realiza algo para cumplir esa meta a pedido del caso de uso.
Los actores pueden ser personas, representadas por roles, o sistemas externos.
Tener en cuenta que un actor hereda los roles que pueden jugar sus ancestros en los casos de uso.
Los actores del sistema solo se deben mostrar cuando sean ellos los que necesitan el caso de uso.

Nivel de abstracción

Para Cockburn[18] los casos de uso pueden presentar distintos niveles de abstracción según el propósito para el que se los utilice.

- *Nivel objetivo del usuario*: El objetivo se cumple mediante una tarea que realiza un único actor, en un mismo lugar, en un momento del tiempo, sin interrupción , y que deja al sistema en un estado consistente.
  Que sea un "único actor" no refiere a que haya una única persona responsable de su ejecución en el ámbito del negocio, sino a que toda vez que uno de los posibles actores inicia el CU éste será ejecutado completamente por el mismo actor.
  Para ver si un caso de uso tiene este nivel de abstracción puede sometérselo a los siguientes tests:
  -El usuario se siente satisfecho después de haber cumplido con esta meta.
  -La productividad del usuario puede ser medida según la cantidad de veces que cumplió este tipo de objetivo en el día.
  -Después de cumplido este objetivo, el usuario puede hacer una pausa para ir a tomar café.
- *Nivel resumen*: Los casos de uso de este nivel, también llamados casos de uso del negocio, cumplen con objetivos de mayor alcance que pueden llevar desde horas hasta años para concretarse.
- *Nivel subsunción*: No constituyen un objetivo autónomo y sólo sirven para cumplimentar objetivos de nivel superior. Ejemplo: Buscar producto. Los distintos niveles se articulan, de modo que los casos de uso de menor nivel indican cómo se concretan los objetivos de nivel superior y los casos de uso de nivel superior explican por qué se ejecutan los casos de uso de menor nivel.

Nos van a interesar los casos de uso de tipo objetivo del usuario.

9.1.2 Relaciones entre casos de uso

Incluye

Esta relación proviene de que un paso en el caso de uso incluyente es la invocación del caso de uso incluido.
Se recomienda utilizar casos de uso incluidos para factorizar comportamiento común entre dos o más casos de uso.

No se recomienda utilizarlos con el mero objeto de realizar una descomposición funcional de un caso de uso particular. Particionar casos de uso individuales tiende a desperdigar comportamiento importante, dificultar la comprensión y enmascarar problemas. Antes de optar por la descomposición funcional vale preguntarse por qué el caso de uso es tan largo y complejo.
Esto puede ser un síntoma de que:

- el caso de uso está intentando cumplir con más de un objetivo
- los pasos tienen demasiado nivel de detalle
- se describen detalles tecnológicos

El caso de uso incluido puede ser del mismo o de menor nivel que el caso de uso que lo incluye.

Ejemplo de caso de uso incluido del mismo nivel:

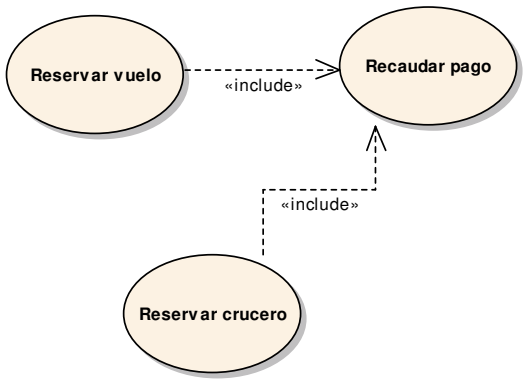


Figura 9: Diagrama de caso de uso

Ejemplo de casos de uso incluidos de menor nivel:

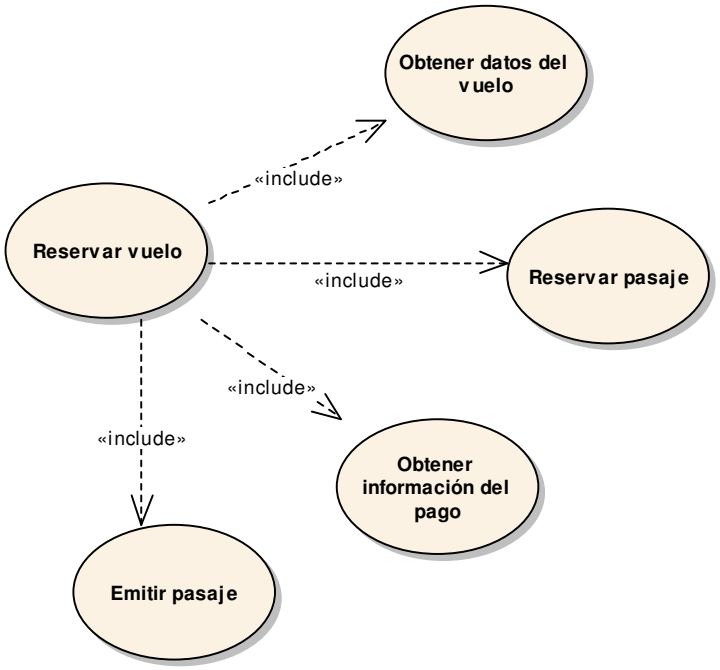


Figura 10: Diagrama de caso de uso

En los diagramas se representan normalmente casos de uso del mismo nivel.
Si queremos mostrar el comportamiento común de menor nivel en un diagrama, hagámoslo en un diagrama aparte.

Extiende

Es difícil ver la diferencia entre un caso de uso que extiende a otro y un escenario alternativo. Podemos decir que:
Un escenario alternativo es una forma de cumplir con el objetivo del CU utilizando otro camino.
Una extensión de un CU tiene objetivos propios que exceden a los del caso de uso.
No debe confundirse el objetivo con los productos que deja el escenario. Si un escenario alcanza el objetivo y además, lateralmente, produce efectos que no producía el escenario básico, el objetivo es el mismo y se modela como un escenario alternativo dentro del mismo caso de uso.
Si, por el contrario, el actor tiene un objetivo distinto al ejecutar este curso alternativo, entonces se trata de un caso de uso aparte, que es extensión del CU original.

Si el base y la extensión son usados por distintos actores, podemos estar seguros que se trata de CU diferentes y no de alternativas dentro de un CU.
El caso de uso extendido debe indicar cuál extiende. Los casos de uso extendidos no son independientes de su caso de uso base y deben referenciar los pasos del caso de uso base (tal como lo hacen los escenarios alternativos).

9.1.3 Como vamos a documentar nuestros casos de usos

Para este proyecto vamos a especificar los casos de usos según las recomendaciones hechas por Cockburn[18] y Bramble[19]. Los elementos de nuestros casos de uso y las reglas o buenas prácticas utilizadas para documentarlos son las siguientes:

Nombre
El nombre del caso de uso debe representar el objetivo del actor primario, expresado como Verbo+Objeto.
Ejemplo: Agregar Proyecto

Objetivo
Debe enunciar las metas o necesidades que el actor principal realiza a través del uso del sistema. La descripción del objetivo debe agregar información de valor adicional al nombre del caso de uso.

Contexto de Uso
Enunciar las condiciones en que habitualmente se utiliza este caso de uso.
Mencionar aquellos casos de uso desde los cuales es posible "navegar" hasta el caso de uso que se está describiendo.

Descripción Resumida
Descripción textual y breve que en pocas palabras explique el caso de uso, con foco en el escenario principal.

Precondiciones

Condición sobre el estado (del sistema) que se requiere antes de dar inicio al CU. (Esto implica que esta condición no necesita ser nuevamente verificada durante el caso de uso).

Esta condición debe poder ser garantizada por el sistema.

Los estados deben ser "observables " por el usuario. Las precondiciones de un CU derivan, en general, de algún otro caso de uso cuya "ejecución" se completó con éxito previamente. Incluirlas siempre que se considere que contribuyan al entendimiento del CU.

Poscondiciones

Describe el estado de cosas que el sistema debe garantizar en caso de finalización exitosa del CU. Se aplica para el escenario principal y los alternativos – los escenarios de excepción no garantizan las poscondiciones.

Declaración sobre el estado del modelo de dominio.

Escenario principal

El escenario principal representa el flujo exitoso más simple o habitual para el caso de uso.

El tamaño de este escenario es habitualmente de 3 a 9 pasos. Un escenario principal de más de 9 pasos puede indicar que o bien el caso de uso tiene múltiples objetivos o bien incluye demasiado nivel de detalle.

Si fuera necesario, puede desplegarse el caso de uso en subcasos de uso que lleven a cabo uno de los pasos del caso de uso principal.

Paso de un caso de uso

Cada paso se escribe como una oración sobre una meta que se cumple.

Preferentemente los pasos de un escenario estarán numerados para facilitar la referencia desde otros escenarios o CU relacionados.

Sintaxis:

<nro paso>. <Sujeto> <verbo activo en presente> <Objeto>

- Debe explicitarse quién realiza la acción.

- No escribir oraciones condicionales en los escenarios.

Escribir condicionales obliga a tratar la condición dentro del escenario y no queremos hacer esto. Cada bifurcación se tratará en un escenario alternativo.
Un paso de un caso de uso puede ser la invocación a otro caso de uso.

Sintaxis:
<nro paso>. Invocar a <XXX>

Pueden intercalarse comentarios fuera de la numeración referentes a:

- repetición de pasos. Ej: "el usuario repite los pasos 3-4 hasta que indica que finalizó la carga"
- orden indistinto de un conjunto de pasos. Ej: "los pasos 3-5 pueden realizarse en cualquier orden".
- Cualquier otra nota aclaratoria

Escenarios alternativos

Un escenario alternativo comienza con el número de paso del escenario principal en el que la bifurcación se produce y la condición que da lugar a la bifurcación. La condición debe ser una condición detectable por el sistema. Por ejemplo, no tiene sentido poner una condición como "Se cortó la luz" si el sistema no puede detectarla.
Como puede haber más de una ramificación para un paso del escenario principal se indica también una letra para distinguir esta alternativa de otras.
Se enuncian a continuación los pasos a seguir para manejar esta alternativa. Indicar luego en qué paso continúa o si, por el contrario, el escenario termina.
La numeración de los pasos debe ser única en el ámbito de todos los escenarios de un CU.

9.2 Guía de uso – Manual del usuario

En esta sección se muestra el funcionamiento de la herramienta desarrollada. Dado que en la aplicación tenemos dos tipos de usuarios, desdoblaremos esta parte en una guía para el administrador y otra para el usuario común.

9.2.1 Guía de uso del administrador

El administrador como es un usuario distinguido cuando ingrese a la aplicación se le presentará una opción adicional al panel de control y a la agenda, la de configuración, para poder dejar operativo el sistema acorde a la organización donde trabaja.

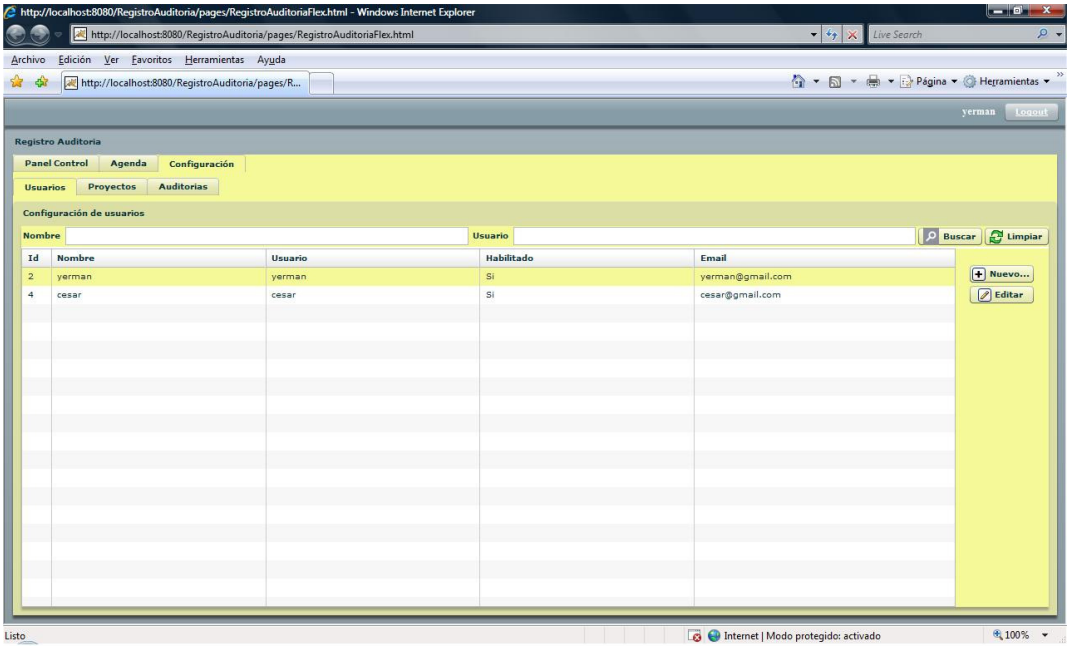


Figura 14 – Vista de administración de usuarios.

Lo primero que podrá realizar el usuario administrador desde esta perspectiva es trabajar con los usuarios, dar de alta nuevos usuarios o modificar los existentes. Una cuestión que vale la pena destacar es que en el sistema no existe la baja física de un usuario, si se quiere evitar el ingreso de un determinado usuario el administrador podrá deshabilitarlo desde la edición del usuario.

Además en nombre de usuario, es decir, el login deberá ser único en todo el sistema y no se podrá modificar.

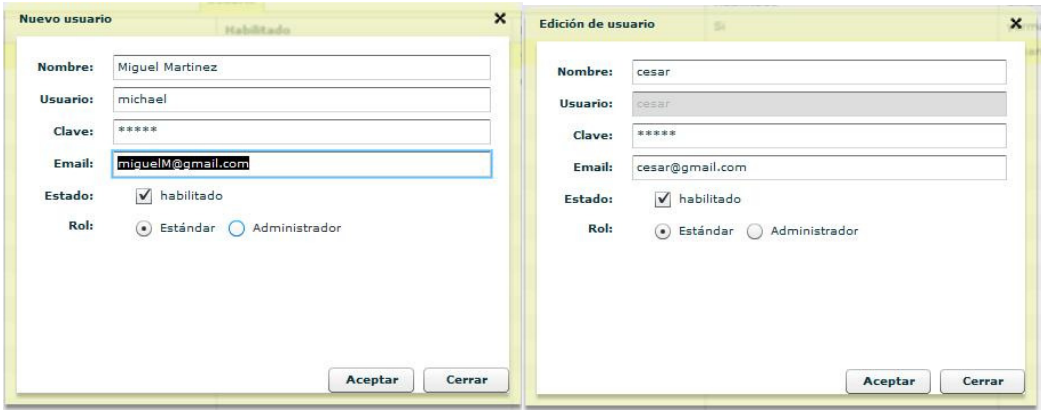


Figura 15 – Diálogos de creación y edición de usuarios.

La segunda opción de la solapa de configuración, "Proyectos" permite que el administrador cargue los diferentes tipos de proyectos que hay en la organización los estados por los que podrán pasar y los roles distinguidos de los proyectos.

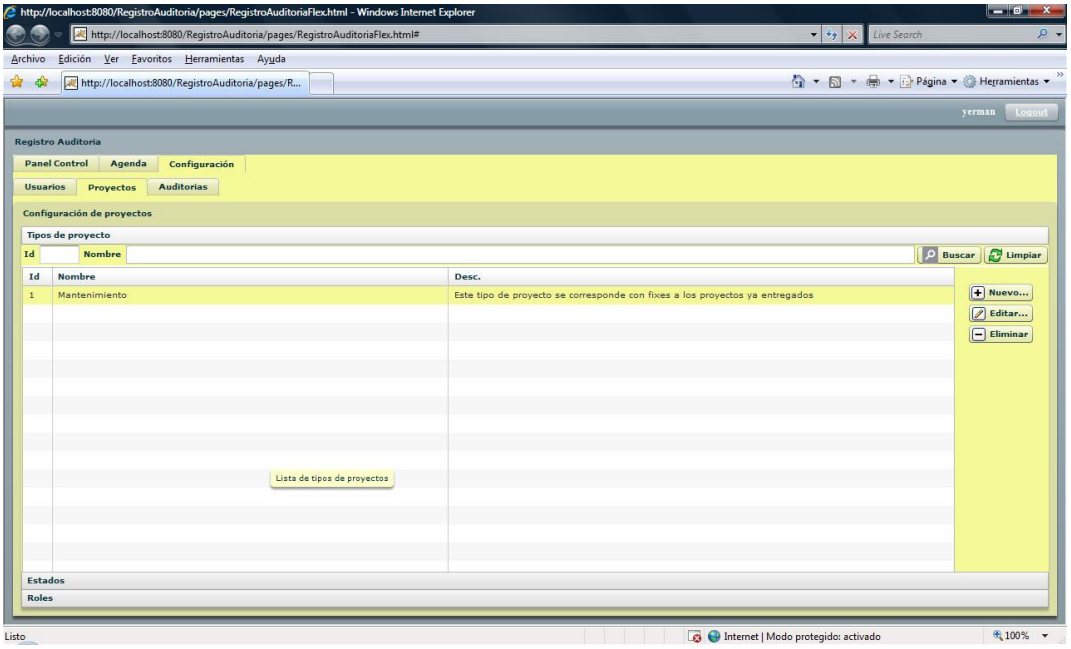


Figura 16 – Vista de administración de tipos de proyectos.

Particularmente en la edición de los tipos de proyectos se pueden configurar las auditorías sugeridas para el tipo de proyecto en cuestión en los diferentes estados por los que puede pasar.

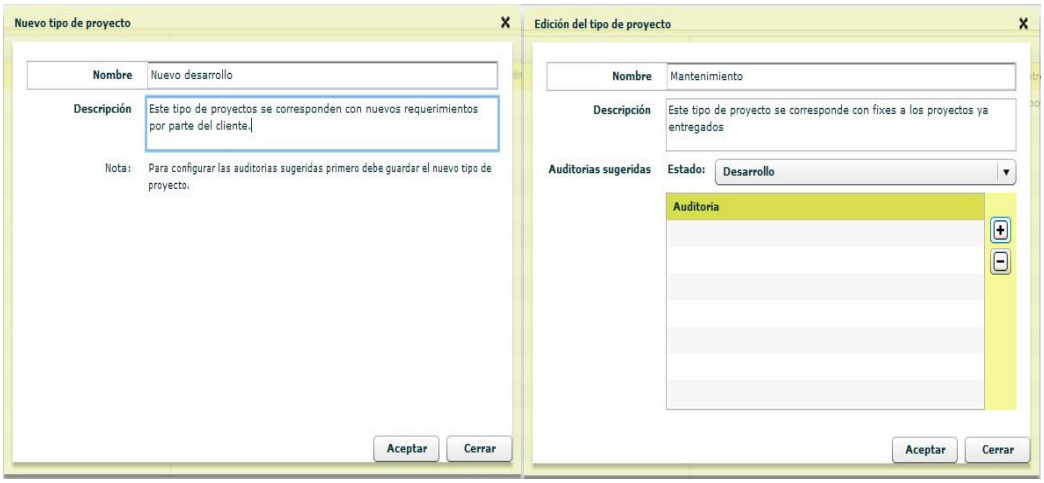


Figura 17 – Diálogos de creación y edición de tipos de proyectos.

En la sección de estados, podrá dar de alta nuevos estados y configurar el diagrama de transición de estados, es decir, para cada estado cuales son sus próximos.

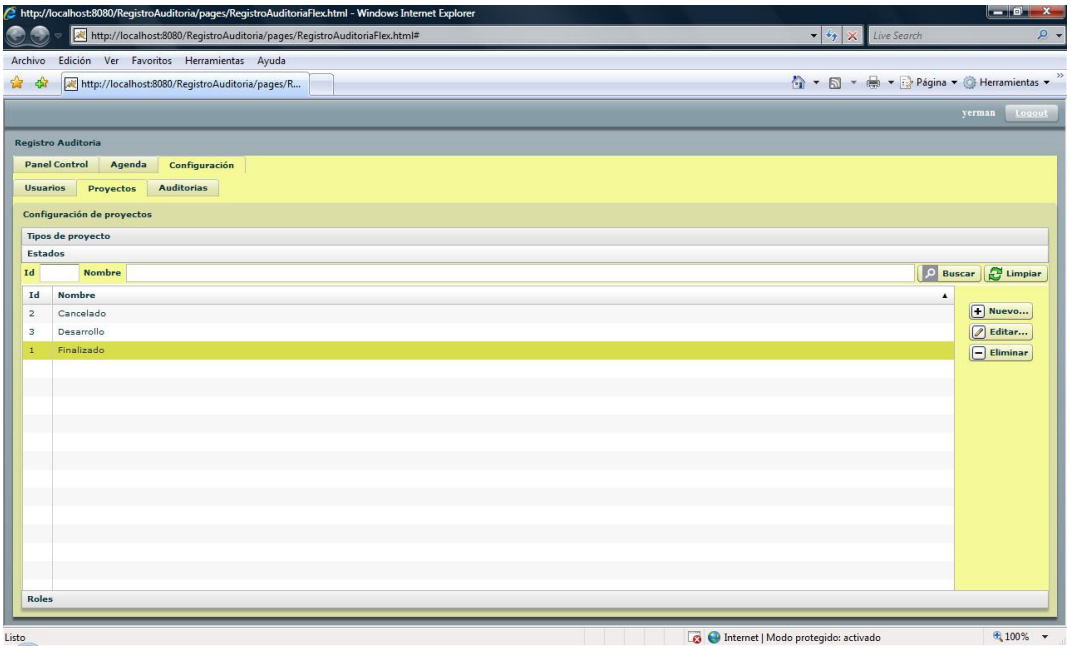
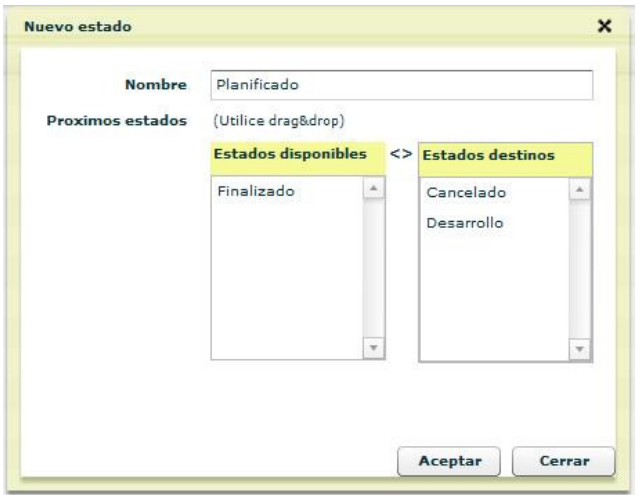


Figura 18 – Vista de administración de estados.

Para la creación o edición de un estado se muestra el siguiente diálogo:



En la lista de la izquierda aparecen todos los estados existentes en el sistema y en la lista de la derecha los estados destino, a donde se puede llegar desde el estado que se está editando o creando. En el ejemplo, un proyecto que esté en estado "planificado" puede pasar a "desarrollo" o a "cancelado".

Figura 19 – Diálogo de creación y edición de estados.

Por último en la solapa de Configuración/Proyectos tenemos la sección para administrar los roles y sus niveles de escalamiento de no conformidades.

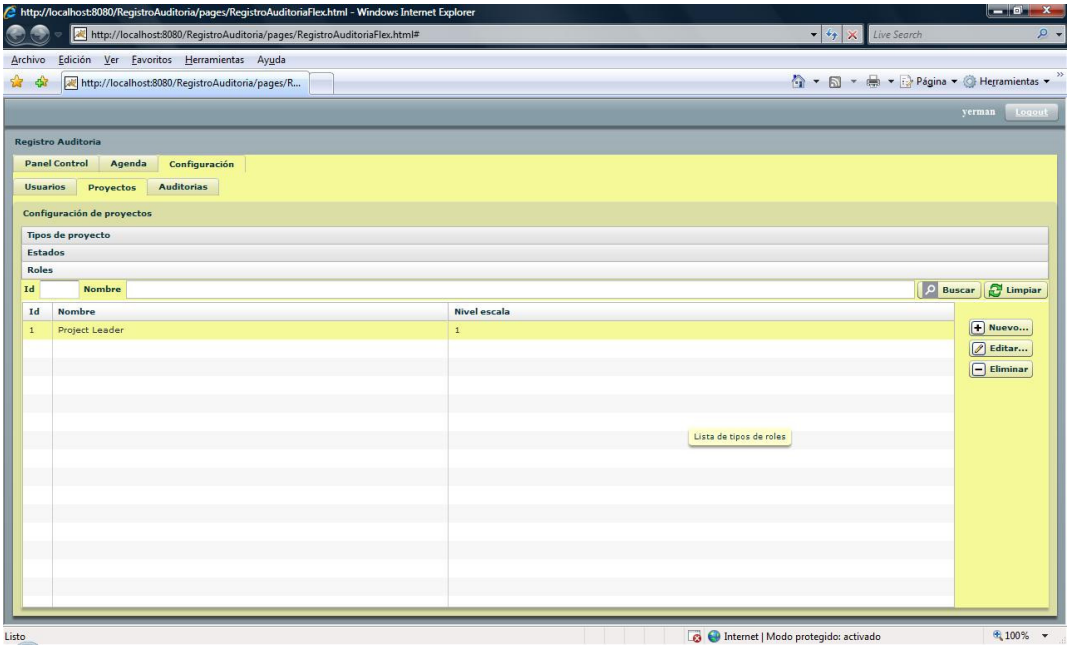


Figura 20 – Vista de administración de tipos de roles.

Antes de comenzar a operar con el sistema el administrador deberá dar de alta los diferentes roles que podrán tener los proyectos. Para cada uno de ellos se debe indicar el orden en el que serán notificados en caso de encontrarse no conformidades. Naturalmente ni el nombre del rol ni el nivel de escalamiento pueden estar repetidos.

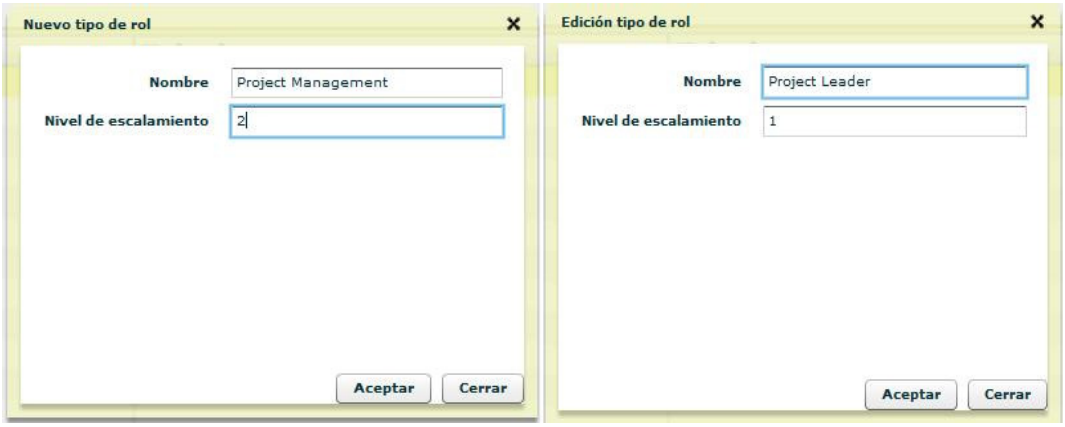


Figura 21 – Diálogos de creación y edición de tipo de rol.

Hay ciertos roles para los que no se requiere que se los notifique acerca de las no conformidades detectadas, por ejemplo para el auditor de un proyecto, ya que es él quien eleva las no conformidades. Para estos casos en el campo nivel de escalamiento se le debe indicar 0 (cero) como nivel.

Para completar la etapa de configuración antes de poder utilizar el sistema, se deberán cargar los diferentes tipos de auditorías a realizar sobre los proyectos, para cada una de ellas se especificará la lista de tareas a realizar para dar por cumplida una auditoría de ese tipo.
Entonces lo primero que se debe hacer es dar de alta todos los tipos de tareas a realizar, desde el menú configuración/Tipos de tareas

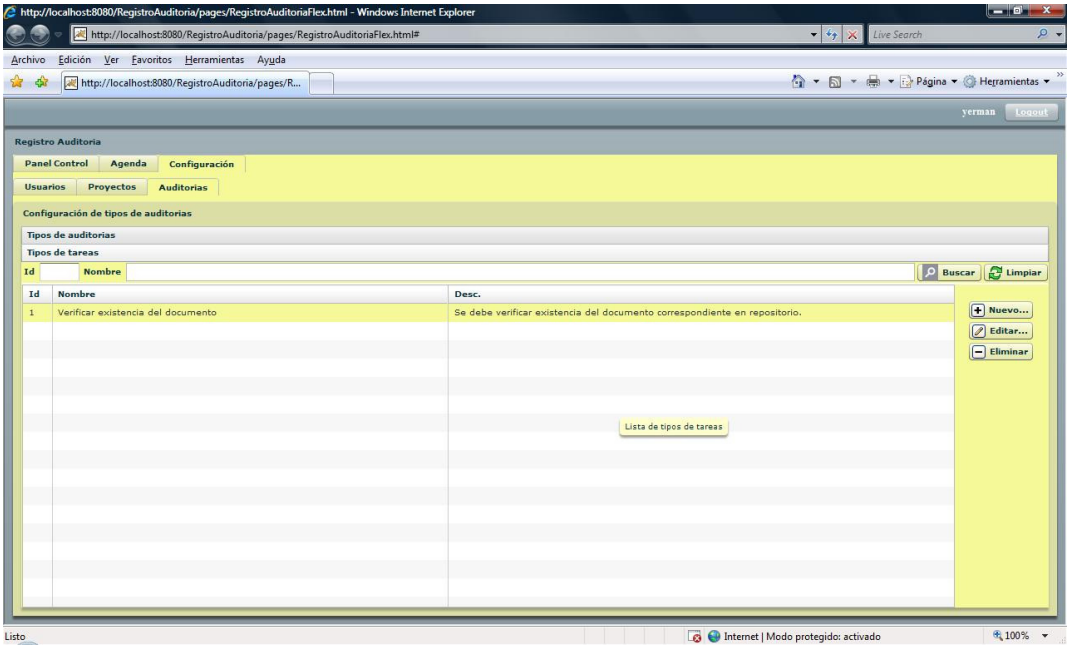
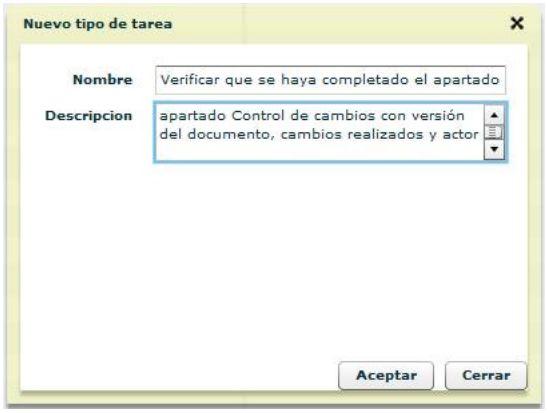


Figura 22 – Vista de administración de tipos de tarea.

Al presionar el botón nuevo en la sección de tipos de tareas se abrirá el siguiente diálogo:



Aquí se deberá completar el nombre de la tarea junto con una breve descripción de las acciones a realizar para cumplir con la misma.

Para la edición de la tarea, está disponible un diálogo como el de la figura.

Figura 23 – Diálogo de creación de tipo de tarea.

Una vez cargadas todas las tareas que determinan los aspectos de no conformidad, hay que crear los diferentes tipos de auditorías; un tipo de auditoría lo podemos ver como un agrupamiento de tipos de tareas. Para esto, desde la vista de administración de tipos de auditorías el administrador deberá dar de alta los nuevos tipos de auditorías.

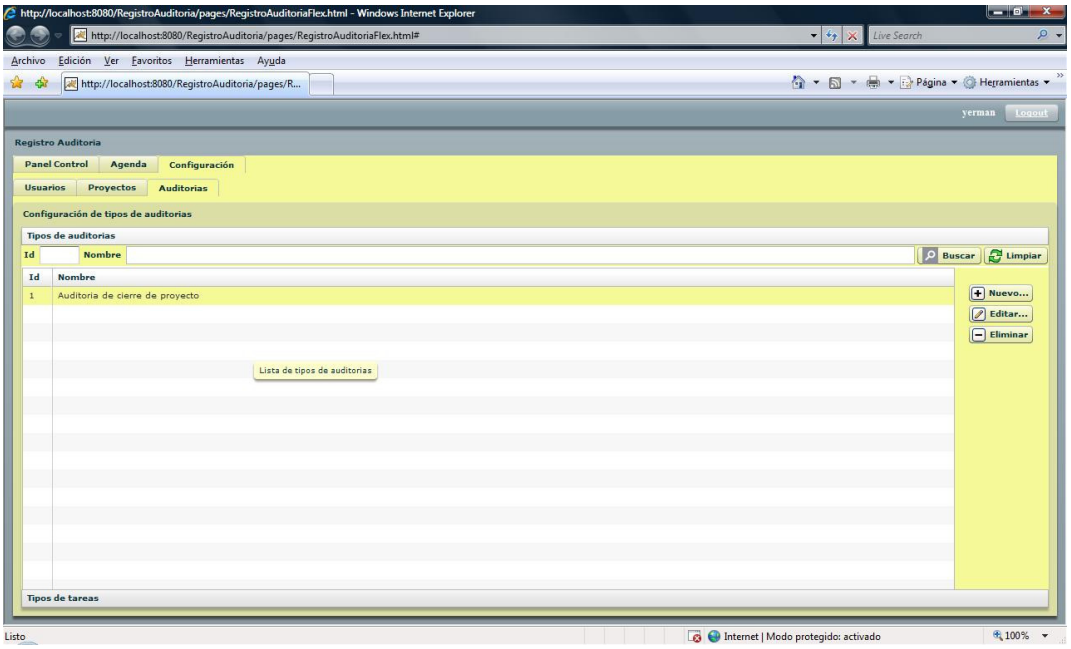


Figura 24 – Vista de administración de tipos de auditorías.

Luego para la creación de un nuevo tipo de auditoría se deberá definir el nombre y el conjunto de tareas que la conformarán.

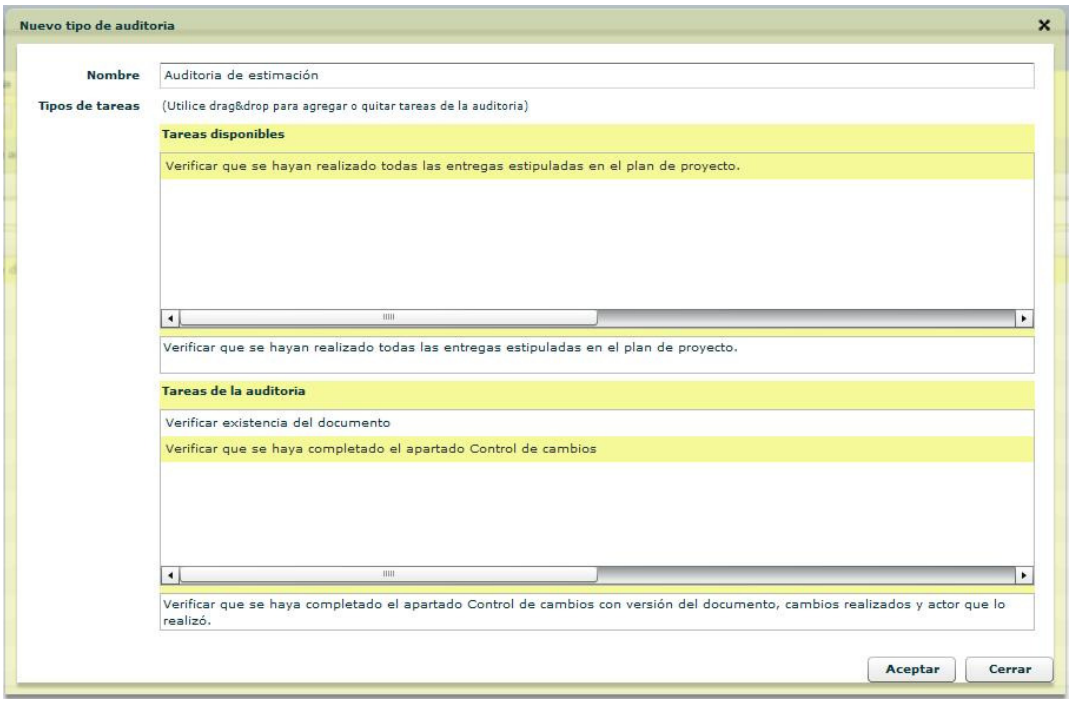


Figura 25 – Diálogo de creación de tipos de auditorías.

En el diálogo de creación o edición de tipos de auditorías se puede ver en la parte superior una lista de tareas disponibles para agregar al tipo de auditoría y en la parte inferior la lista de tareas que efectivamente conforman la auditoría.
Debajo de cada lista hay un recuadro en el cual se mostrará la descripción de tarea seleccionada.
Para pasar una tarea de una lista a la otra se debe utilizar drag and drop, es decir, arrastrar la tarea seleccionada y soltarla en la lista correspondiente.

Eliminación

Finalmente, para el administrador hay que aclarar que salvo para los usuarios en donde no existe una baja física, sino que se pueden deshabilitar; para las demás entidades, tipo de proyecto, estados, tipo de rol, tipo de tareas y tipos de auditorías, se pueden eliminar físicamente, siempre y cuando no estén asociadas a algún proyecto.

9.2.2 Guía de uso del usuario

Antes de comenzar con la guía de uso para los usuarios comunes debemos aclarar que tomamos al usuario administrador como un usuario común pero con la posibilidad de configurar el sistema. La funcionalidad que mostraremos en esta sección también está disponible para los usuarios administradores.

9.2.2.1 Agenda

Uno de los puntos por los que se distingue este sistema es por la posibilidad de agendarse las auditorías a realizar sobre los proyectos y la opción de hacernos recordar que debemos chequear una cierta no conformidad. Esto puede realizarse desde la agenda que provee la aplicación.

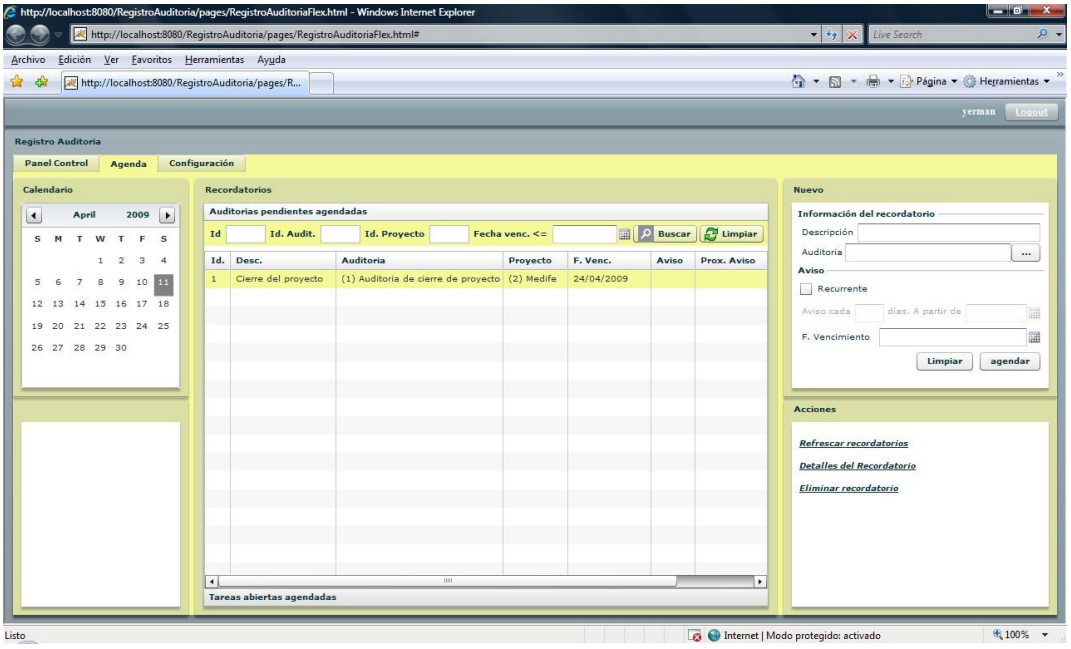


Figura 26 – Vista de la agenda

La agenda esta dividida en dos partes, la primera, la que permite agendar auditorías y la segunda la que permite agendar tareas abiertas o no conformidades.

En la parte superior del margen derecho, se ubica el formulario para poder cargar el recordatorio, para cualquiera de los dos casos, auditoría o tarea, para dar de alta un nuevo recordatorio se debe completar la descripción, seleccionar el elemento a agendar,

y luego se debe indicar si el aviso va a ser recurrente o si solo se alertará al usuario por una única vez en una determinada fecha. En el caso de que sea recurrente se debe indicar la frecuencia en cantidad de días y a partir de que fecha empieza a correr el plazo.

No existe la edición de recordatorios, si se ingresó mal algún dato se debe eliminar el recordatorio y darlo de alta nuevamente. Para eliminar un recordatorio se puede oprimir la tecla suprimir sobre el elemento a borrar o se puede ir a la opción "Eliminar recordatorio" del menú que se encuentra en la parte inferior del margen derecho.

Para ver la información completa del recordatorio agendado se puede hacer doble clic sobre él o ir a la opción "Detalles del recordatorio" del menú que se encuentra en la parte inferior del margen derecho. Se abrirá el siguiente diálogo:

Figura 27 – Diálogo donde se muestra la información completa del recordatorio

En esta pantalla se puede ver la información completa del recordatorio, además desde aquí se puede eliminar el recordatorio ejecutando la acción "Descartar".

Avisos

Cuando el usuario ingresa al panel de control (ésta es la vista principal y la que se abre por defecto) para comenzar a operar con los proyectos, en el margen izquierdo se le listan los recordatorios por él agendados. Separados según los siguientes criterios:

1. En primer lugar, se le listan los recordatorios vencidos, es decir, aquellos cuya fecha de vencimiento es menor a la fecha corriente o aquellos que por ser recurrentes ya se ha vencido el plazo de aviso.
2. Luego, se listan los recordatorios que vencen en el día actual.
3. Por ultimo, se listan los recordatorios que vencerán mañana.

De esta manera se le avisa al usuario cuando ingresa a la aplicación el estado de su agenda.

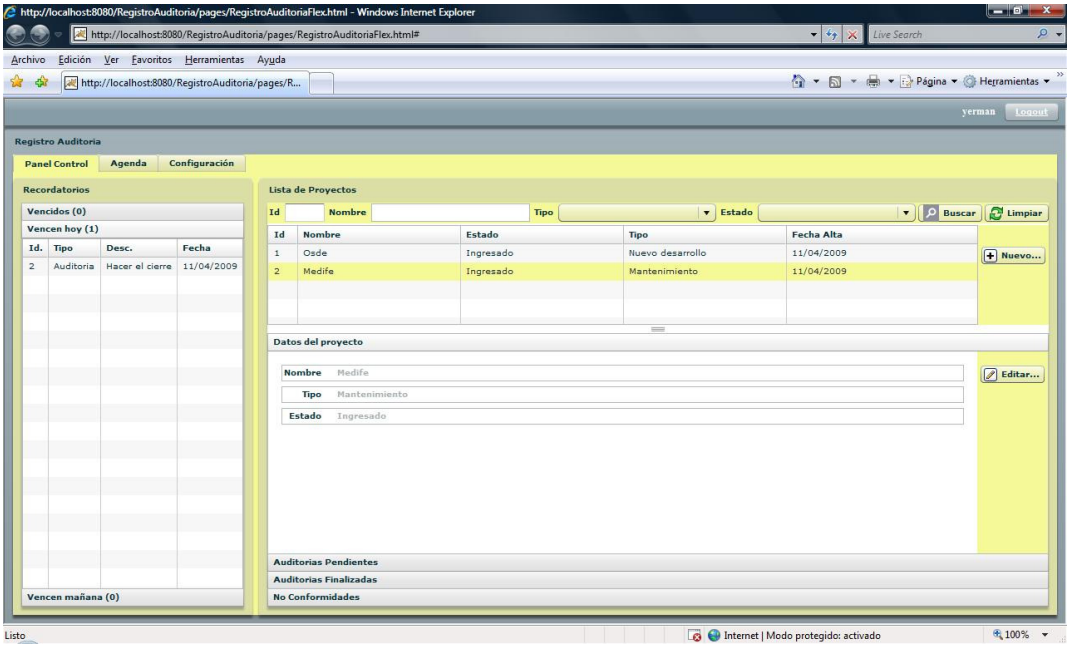


Figura 28 – Avisos para el usuario.

El usuario podrá desde el panel de control trabajar con los recordatorios, haciendo doble clic sobre ellos se le desplegará el mismo diálogo que si lo estuvieran haciendo desde la agenda(Figura 14) con la diferencia que desde aquí además de “Descartar” se podrá “Posponer” un recordatorio, esto significa para el caso de que el recordatorio sea recurrente, resetear el plazo de días para el próximo aviso, es decir, comienza la cuenta desde cero nuevamente. En el caso que el recordatorio no sea recurrente el posponerlo

lo saca de la vista del panel de control, ocultándolo hasta el próximo ingreso del usuario al sistema.

Figura 28 – Diálogo donde se muestra la información completa del recordatorio desde el panel de control.

9.2.2.2 Panel de control

En la solapa panel de control se encuentra la funcionalidad necesaria para poder auditar un proyecto a partir de las configuraciones cargadas previamente en la solapa de configuración.

Cuando un usuario quiere dar de alta un nuevo proyecto, el sistema le provee al auditor un conjunto de tipos de proyectos previamente configurados por el administrador. El auditor deberá entonces completar el nombre del proyecto, seleccionar de que tipo será y completar cada uno de los roles solicitados. El sistema mostrará un mensaje de error en los casos en que no se seleccione ningún tipo de proyecto, se ingrese un nombre de proyecto existente o no se seleccione un usuario para cumplir cada rol. Por defecto, el estado de un proyecto nuevo es “Ingresado”.

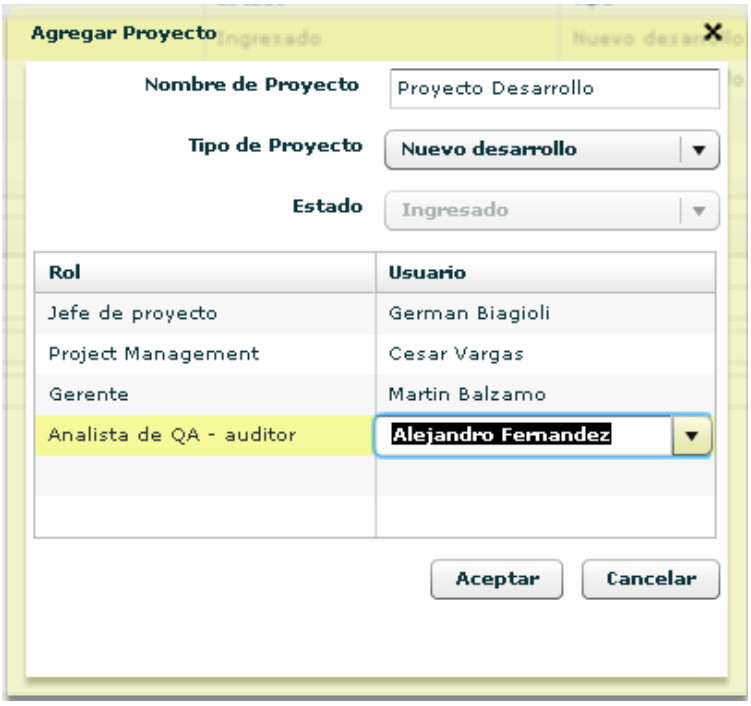


Figura 29 – Diálogo de ingreso de un nuevo proyecto a ser auditado.

En caso de que existan auditorías para ser sugeridas de acuerdo al tipo de proyecto seleccionado, el sistema mostrará la siguiente pantalla en la cual el usuario podrá seleccionar las auditorías sugeridas que desee agregar al proyecto creado.

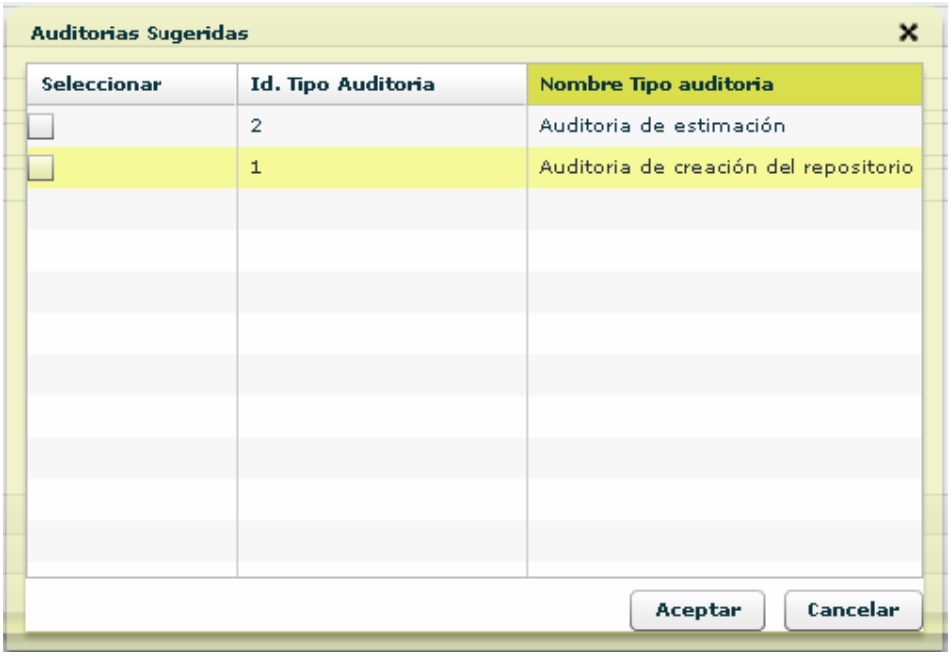


Figura 30 – Diálogo de auditorías sugeridas por el sistema.

Una vez creado el proyecto, y seleccionadas las posibles auditorías, el usuario podrá modificar el nombre y podrá cambiar el estado en el que se encuentra el proyecto.

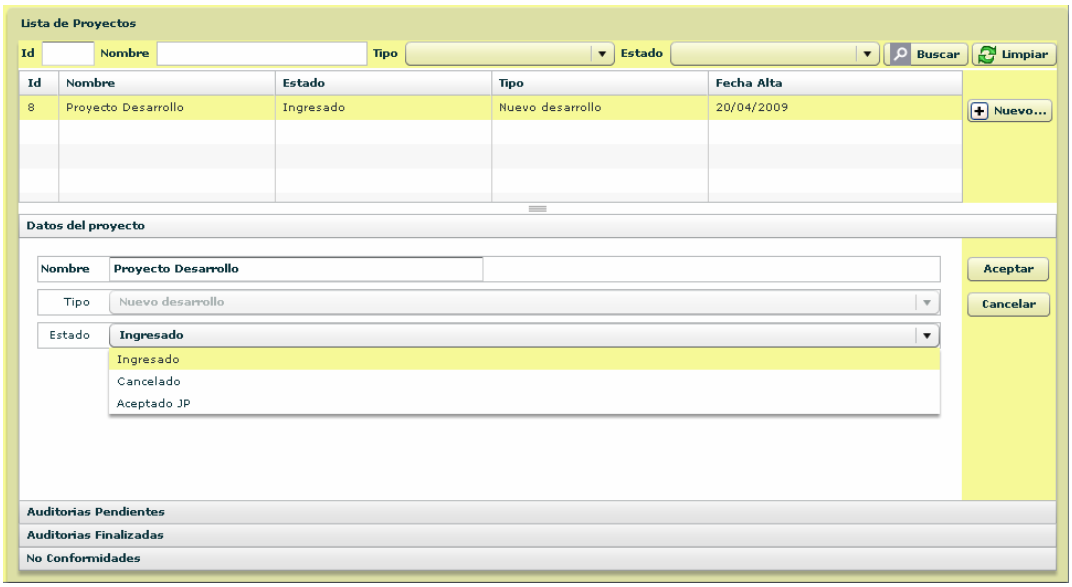


Figura 31 – Vista de modificación de un proyecto.

Durante el cambio de estado, en caso que el proyecto tenga auditorías abiertas, el sistema consultará si se quiere seguir de todos modos con el proceso, y en caso afirmativo se cerrarán cada una de dichas auditorías.

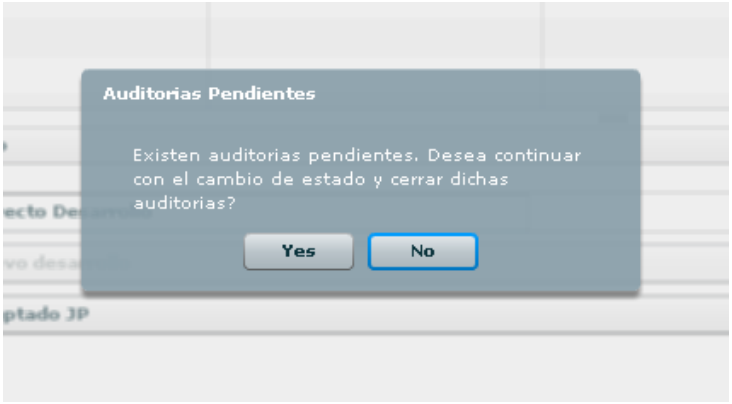


Figura 32 – Diálogo informativo de auditorías pendientes.

Dentro del panel de control, se encuentran las auditorías pendientes seleccionadas previamente por el auditor. En la vista Auditorías Pendientes, el usuario podrá

visualizar los datos de las auditorías a realizar o que se están realizando sobre el proyecto; podrá abrir la lista de tareas correspondientes a dicha auditoría y chequearlas como abiertas (No Conformidades) o Cerradas (Cumplidas).

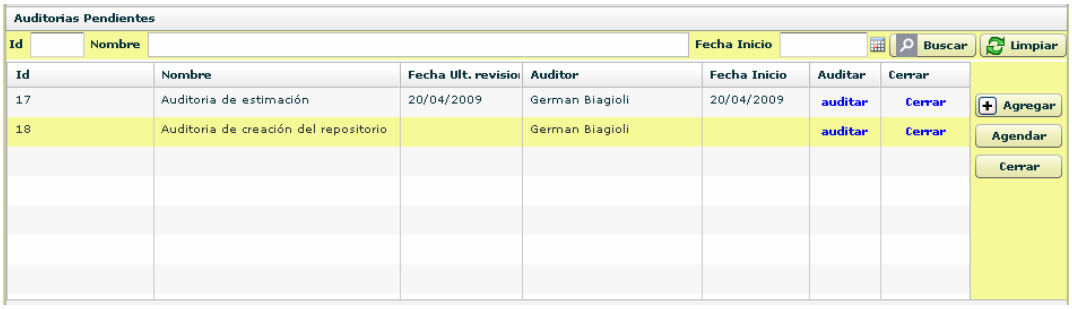


Figura 33 – Vista de auditorías pendientes.

En caso de que se quiera agregar una nueva auditoría, se le mostrará el diálogo de "auditorías sugeridas" previamente mencionado.

Desde esta vista, el usuario también podrá agendarse una auditoría para ser chequeada en el futuro, haciendo clic sobre el botón agendar se abrirá el diálogo

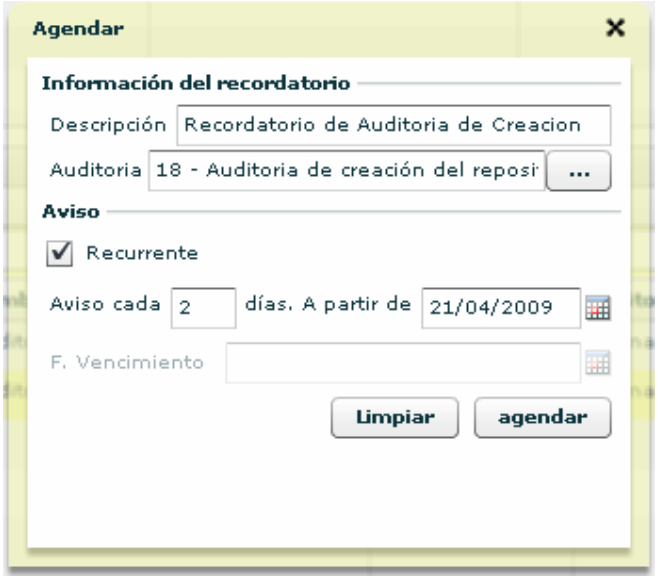


Figura 34 – Diálogo de recordatorio de auditorías.

Este diálogo es exactamente igual al utilizado para guardar un recordatorio en la agenda, el usuario deberá ingresar una descripción y tendrá la posibilidad de seleccionar entre un recordatorio recurrente (cada x cantidad de días) a partir de una fecha, o ingresar una fecha de vencimiento la cual servirá para recordar que se debe realizar la auditoría ese

día. Luego el elemento agendado aparecerá en los recordatorios. Para ver esta sección en detalle ir a: Avisos

A través de la opción "cerrar" de las auditorías pendientes el usuario podrá cerrar una auditoría aunque no se hayan chequeado todas sus tareas. En este caso se le consultará si está seguro de cerrarla y en caso afirmativo, se cerrará la auditoría y se pasará a la vista de auditorías finalizadas.

En caso de que el usuario haga clic sobre el link Auditar de una auditoría pendiente, el sistema mostrará una lista de todas las tareas, y se podrá chequear, generar una no conformidad o escalar una tarea abierta.

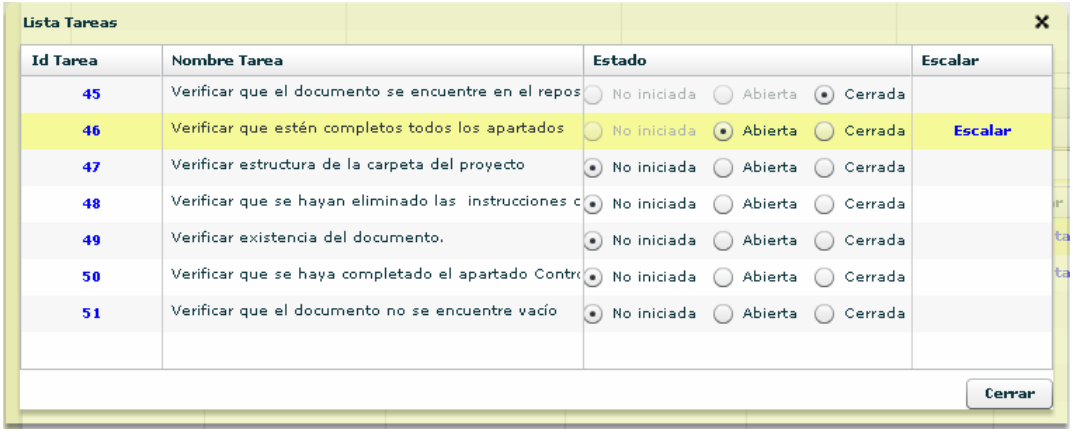


Figura 35 – Lista de tareas de una auditoría.

En el caso que se abra una tarea o se haga clic en Escalar, el usuario deberá ingresar un motivo de la no conformidad, y el sistema enviará un mail al usuario que cumpla con el rol del próximo nivel de escalamiento configurado. El link para escalar una no conformidad, aparecerá solamente para las tareas en estado "abierta".
Cuando se cierra la última tarea de la auditoría, el sistema automáticamente cierra la auditoría a la cual pertenece y la pasa a la lista de auditorías finalizadas.

Figura 36 – Lista de Auditorías Finalizadas.

Por otra parte, en caso de que se abra una tarea o se escale, dicha tarea aparecerá en la lista de No Conformidades.

Figura 37 – Lista de No Conformidades.

En esta pantalla el usuario podrá chequear como cumplida la No Conformidad (haciendo click en Auditar) o podrá escalar dicha tarea por el no cumplimiento luego de pasado el plazo para ser corregida(haciendo click en Escalar). Al darse por cumplida la tarea, la no conformidad es eliminada de la lista, y en caso de que sea la última tarea abierta de la auditoría, ésta será cerrada automáticamente y se mostrará en la lista de auditorías finalizadas.

De igual manera que en las auditorías pendientes, el usuario podrá agendarse una no conformidad, para ser chequeada en el futuro. El recordatorio aparecerá en la misma lista de recordatorios de las auditorías.

Referencias

Las siguientes referencias se han tomado como base para la elaboración de esta tesis.

[1] Pressman, R., **Software Engineering: A Practitioner's Approach**, McGraw-Hill, 4th edition, 1997.

[2] Humphrey, W., **A Discipline for Software Engineering**, Addison-Wesley, 1995.

[3] CMMI for Acquisition ***CMMI-ACQ***, version 1.2. Software Engineering Institute, http://www.sei.cmu.edu/pub/documents/07.reports/07tr017.pdf
(Último acceso:05/09)

[4] IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, 1990.

[5] Eickelman, N. and Hayes, J. eds., "**New Year's Resolutions for Software Quality**", *IEEE Software*, Jan-Feb 2004, pp. 12-13.

[6] Crosby, P., **Quality Without Tears**, McGraw-Hill Book Company, 1984, p.64.

[7] Galin, D., **Software Quality Assurance – From Theory to Implementation**, Pearson Education Limited, 2004.

[8] Paulk, M., et. al., **The Capability Maturity Model: Guidelines for Improving the Software Process**, Addison-Wesley, 1995.

[9] Reifer, D., **State of the Art in Software Quality Management**, Reifer Consultants, 1985.

[10] Schulmeyer, G. G. and McManus, J. I., eds. **Handbook of Software Quality Assurance**, 2nd ed, Van Nostrand Rheinhold, 1992.

[11] Bach. J., "**Good Enough Quality – Beyond the Buzzword**", *IEEE Computer*, August 1997.

[12] Appraisal Requirements for CMMI (ARC). Software Engineering Institute, http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr011.pdf
(último acceso:05/09)

[13] Publicación de los resultados de las evaluaciones CMMI - http://www.sei.cmu.edu/appraisal-program/appraisal-results.html
(Último acceso:05/09)

[14] Informe sobre CMMI, http://www.sei.cmu.edu/cmmi/adoption/pdf/cmmi-overview07.pdf
(Último acceso:05/09)

[15] Mike Phillips, CMMI Program Manager**. CMMI V1.1 and Appraisal Tutorial,** 2004, Carnegie Mellon University.

[16] Zubrow, D., Hayes, W., Siegel, J., and Goldenson, D. (1994), Maturity Questionnaire.Pittsburgh, PA.: Software Engineering Institute, Carnegie Mellon University.

[17] Martin Fowler, **Patterns of Enterprise Application Architecture**, 2007, Addison-wesley.

[18] Cockburn Alistair, **Writing Effective Use Cases**.,2000, Addison-Wesley.

[19] Bramble Adolph, **Patterns for Effective Use Cases,**2001.

[20] Craig Walls, **Spring in Action**,2007, 2da edicion, Manning.

[21] Christian Bauer,Gavin King, **Hibernate in Action**, 2005, Manning.

[22] Hibernate Reference Documentation, http://docs.jboss.org/ejb3/app-server/Hibernate3/reference/en/pdf/hibernate_reference.pdf
(Último acceso:05/09)

[23] Sitio de ayuda de Adobe Flex. http://livedocs.adobe.com/flex/3/html/help.html
(Último acceso:05/09)

[24] Sitio oficial de acegi http://www.acegisecurity.org/
(Último acceso:05/09)

[25] Sitio oficial de MySQL http://www.mysql.com/
(Último acceso:05/09)

[26] Sitio de Maven 2 http://maven.apache.org/index.html
(Último acceso:05/09)

[27] Sitio de Eclipse http://www.eclipse.org/
(Último acceso:05/09)

[28] Sitio oficial CMMI http://www.sei.cmu.edu/cmmi/
(Último acceso:05/09)

[29] Universidad Carnegie Mellon http://www.cmu.edu/index.shtml
(Último acceso:05/09)

[30] CMMI Overview http://www.sei.cmu.edu/cmmi/general/
(Último acceso:05/09)

[31] Capability Maturity Model for Software http://www.sei.cmu.edu/cmm/
(Último acceso:05/09)

[32] CMM-SW http://es.wikipedia.org/wiki/CMM-SW

(Último acceso:05/09)

[33] Systems engineering leading indicators for assessing program and technical effectiveness.

[34] ftp://ftp.cs.kuleuven.be/pub/Ada-Belgium/ase/ase02_01/bookcase/se_sh/cmms/ipd/IPDovrvw_lkd.pdf

(Último acceso:05/09)

[35] Sitio oficial CMMI http://www.sei.cmu.edu/cmm-p/version2/faq.html

(Último acceso:05/09)

[36] Sitio oficial CMMI http://www.sei.cmu.edu/arm/SA-CMM.html

(Último acceso:05/09)

[37] Sitio oficial SSE-CMM http://www.sse-cmm.org/index.html

(Último acceso:05/09)

[38] Sitio oficial CMMI http://www.sei.cmu.edu/pub/documents/95.reports/pdf/mm003.95.pdf

(Último acceso:05/09)

[39] Sitio oficial CMMI http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html

(Último acceso:05/09)

[40] Sitio oficial CMMI http://www.sei.cmu.edu/programs/acquisition-support/cmmi-acq.html

(Último acceso:05/09)

[41] Sitio oficial CMMI http://www.sei.cmu.edu/cmmi/models/CMMI-Services-status.html

(Último acceso:05/09)