UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 8

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

Plan de Aseguramiento de la Calidad para proyectos de software sobre la Arquitectura MDA (Model Driven Architecture)**.**

AUTOR: Ilidian Guzmán Santana.

TUTORES: Ing. Ramsés Delgado Martínez.
Ing. Armado Batista Piñeda.

Ciudad de la Habana.
Julio 2008
"Año de 50 de la Revolución".

"La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia."

John Ruskin

A mi mamá, a mi papiadelto,
A mi papá y mi hermanita,
A mis abuelos, especialmente a
mi abuelita Gladys.
A mi nenito con todo mi amor…
Ili

Agradecimientos

Al compañero Fidel por tan ingeniosa idea de construir la Universidad de las Ciencias Informáticas, esta casa de altos estudios, y darles el derecho a jóvenes de todo el país a crecer gratuitamente como profesionales en sus aulas y laboratorios.

A mi tutor Ramsés, gracias por dedicarme parte de su tan agitado tiempo, por tener tanta paciencia a la hora de enseñarme, por aguantar mis malcriadeces y mis temores después de cada corte de tesis, a lo mejor no he sido la mejor tesista pero siempre me acordare de él y le estaré eternamente agradecida por ser la personas que más me ha ayudado con mi tesis.

A mi tutor Armando, gracias por estar dispuesto a ayudarme en lo que hiciera falta, a pesar de no estar empapado del tema de la investigación.

A todos los profesores, gracias por tratar de inculcarme sus conocimientos.

A la UCI por ser mi hogar durante los últimos cinco años de mi vida.

A mi mamita del alma por ser mi todo, de no ser por ella no habría logrado tanto, por saber que esta allí para mí siempre, por brindarme tanto amor, por ser la luz de mi vida.

A mi papiadelto por darme tantos consejos y depositar tanta confianza en mí, por ser el responsable de todas mis malcriadeces desde pequeña, por estar siempre que lo necesito y por quererme tanto.

A mi nenito, gracias mi amor por estar siempre a mi lado, por convertirte también en mi compañero de tesis, por brindarme tu apoyo en todo lo que necesite y por regalarme tu amor.

A mi papito, por estar tan orgulloso de mí, por quererme tanto, por no olvidarme, por haberme brindado tanto apoyo en esta última y tan difícil etapa de mi vida.

A Vero por ayudarme y darme tantos consejos, por estar siempre pendiente a todo lo que me pasa.

A mi hermanita linda por quererme tanto y querer estar siempre a mi lado, por escucharme y hacerme tanto caso.

A mi nueva familia del Mónaco, a mi tío Raúl, a Gladys y a mis primitos lindos Alex y Carlitos, por estar tan dispuestos a ayudar, por estar siempre preocupados y al tanto de todo lo que sucede conmigo, por quererme tanto, por brindarme su hogar y convertirme en una más de su familia.

A mis suegros, muchas gracias por estar tan pendientes y preocupados siempre, por darle la vida a mi rey y por brindarme su apoyo en todo.

A mi padrino, muchas gracias por todo lo que ha hecho por mí, por estar tan pendiente siempre, por tratar de complacerme en todo, y aunque no pueda estar físicamente a mi lado en el día de mi graduación ha sido un gran partícipe en mi carrera.

Agradecimientos

A todos mis tíos, en especial a mi tío Silvio por venir a verme tantas veces y suministrarme siempre lo que necesité y por ayudarme cada vez que pudo, a mi tío Pucho nunca lo olvidaré, a él le dedico todo mi esfuerzo, aunque ya no esté entre nosotros.

A mis abuelos, por quererme tantos, a mi Aña por cuidarme tanto desde chiquita y brindarme tanto cariño, a mi abuelito Víctor por tratar de complacer siempre mis antojos, a mi abuelita Gladys, muchas gracias, a mi gordita linda que nadie se imagina cuanto la extraño, cuanto me duele que ya no esté, a ella le dedico todo mi esfuerzo, a ella que tanto quería que me graduara y que estuviera a su lado.

A todas mi amigas, Dayliana, Inés, Greilan, Yendy, Jenely, Mailen y Anagel, muchas gracias por ayudarme tanto y estar siempre a mi lado, por aguantar todas mis malcriadeces y por saber que puedo contar con ellas en lo que sea.

Al resto de mi familia, que por motivos de tamaño no puede ser mencionada cada persona en particular, pero que siempre me apoyaron y estuvieron al tanto de todo lo que ha acontecido en mi vida.

A mi Diosito, muchas gracias por darme fuerza para seguir antes las dificultades.

A todos mis compañeros de cinco años de estudio y de convivencia, por brindarme su amistad y compartir tantos momentos.

En fin, muchas gracias a todas las personas que me extendieron su mano y me ayudaron de una forma u otra…

Declaración de Autoría

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

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

_____
Ilidian Guzmán Santana
Firma del Autor

_____
Ramsés Delgado Martínez
Firma del Tutor

_____
Armado Batista Piñeda
Firma del Tutor

TUTOR: Ramsés Delgado Martínez.

Graduado de Ingeniero Informático en el Instituto Superior Politécnico "José Antonio Echeverría" (CUJAE) en el año 2006.

Especialista de la Dirección de Calidad de Software de la UCI.

Correo electrónico: ramsesd@uci.cu

Ubicación: Universidad de las Ciencias Informáticas (UCI), Cuba.

TUTOR: Ing. Armando Batista Piñeda.

Graduado de Ingeniero Informático en la Universidad de Holguín "Oscar Lucero Moya" en el año 2006.

Profesor del Departamento de Ingeniería y Gestión de Software.

Correo electrónico: armandobp@uci.cu

Ubicación: Universidad de las Ciencias Informáticas (UCI), Cuba.

RESUMEN

Actualmente en la UCI se genera un gran número de proyectos productivos y se hace compleja la tarea de asegurar la calidad de cada uno de ellos, por lo que es necesaria la aplicación de planes de Aseguramiento de la Calidad según las arquitecturas en las que se desarrollen. En el presente Trabajo de Diploma, se propone un Plan de Aseguramiento de la Calidad para proyectos de software sobre la Arquitectura MDA (Model Driven Architecture), para su elaboración, se abordan los aspectos teóricos que ayudan a tener un dominio del tema, se analizan cada uno de los elementos necesarios a tener en cuenta para su elaboración y adaptados a las características específicas que posee la Arquitectura MDA, y se valida la propuesta consultando los criterios de expertos para determinar la probabilidad de éxito de la propuesta.

PALABRAS CLAVES

Plan, Aseguramiento, Calidad, Arquitectura, MDA

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

Hoy día se vive una revolución informática y los paradigmas con los cuales se veía la economía hace algunos años atrás no son los mismos. Actualmente, la empresa más rentable del mundo es una de software al igual que la persona más rica del mundo es un empresario de software, situación impensable hace 40 años, cuando las empresas que integraban esas listas eran petroleras o siderúrgicas.

En ese sentido, el desarrollo de software constituye un sector de gran importancia mundial, se encuentra en el centro de todas las grandes transformaciones; sobre todo si se considera que los grandes temas del momento, como lo son la economía digital, la evolución de las empresas y la administración del conocimiento, se resuelven con software.

Para Cuba, como país en vía de desarrollo, el progreso de las tecnologías de información y la industria del software en particular, es un medio importante para incrementar su desempeño económico, atendiendo a que la industria del software requiere menos inversión que las restantes, tomando en consideración que la inversión principal ya se ha realizado a partir de la política educacional que ha seguido la Revolución desde 1959. El principal activo en la industria del software se encuentra en los técnicos y profesionales de carreras relacionadas con la informática que requieren, además de la especialización técnica, contar con una amplia visión para la creatividad y la innovación, elementos básicos en la generación de software. Incursionar en el mercado mundial del software requiere contar con un aval sólido que demuestre la calidad de los productos de software.

El aseguramiento de la calidad del software es en la actualidad uno de los tópicos de investigación más importante en el mundo dentro del área de ingeniería del software.

Para nuestro país ya es una realidad la idea de ocupar un puesto en la industria de software mundial, no es menos cierto que el camino es extenso pero no por eso dejara de ser transitable, si se opta por la correcta utilización y desarrollo de las tecnologías de la información y del capital profesional, y se enfatiza en asegurar la calidad de nuestros productos de software poco a poco se irán construyendo las bases hacia una economía sustentada por la industria del software como principal renglón para garantizar su solidez. Para alcanzar los objetivos trazados se prioriza la formación de profesionales, en universidades como el Instituto Superior Politécnico “José Antonio Echeverría” (CUJAE) y la nueva Universidad de las Ciencias Informáticas (UCI) que adquieren en su seno a estudiantes de todas partes del país**,** se prioriza la atención a tecnológicos de informática y se ha hecho una realidad la Universalización de la Carrera de Ingeniería Informática que permite de una forma viable que todas

aquellas personas logren superarse profesionalmente o dotarse de aptitudes y conocimientos básicos de informática**.**

En pos de alcanzar los objetivos trazados, se reconocen todas las fuerzas caminadas a asegurar la calidad de los productos y procesos, para elevar el prestigio de la Industria Cubana de Software. Los universitarios cubanos no están ajenos a este compromiso, unas de sus misiones fundamentales es el desarrollo de trabajos investigativos que contribuyen a mejorar la industria y la economía nacional.

Defendiendo estas ideas, en la UCI existe una dirección de calidad cuya misión es promover el desarrollo de procesos de formación, diagnóstico y certificación en el área de mejoramiento continuo de la calidad, para elevar la calidad de la producción de software en la UCI contribuyendo al aumento de la productividad y la calidad en los productos.

Este grupo realiza un seguimiento periódico de las actividades productivas de la UCI para definir estrategias correctas que aseguren la calidad de sus productos y entre sus objetivos principales está el de contribuir a la identificación, generación, promoción y adopción de estándares, normas y mejores prácticas relacionadas con la calidad en la Ingeniería de Software.

En aras de promover la investigación y la búsqueda de soluciones de los principales problemas en el área de la Calidad de Software, se arrojó la siguiente **situación problemática**:

Actualmente en la UCI se genera un gran número de proyectos productivos y se hace compleja la tarea de asegurar la calidad de cada uno de ellos, por lo que es necesaria la aplicación de planes de Aseguramiento de la Calidad según las arquitecturas en las que se desarrollen.

Debido a las ventajas que proporciona la utilización de la Arquitectura MDA ha aumentado mundialmente el número de proyectos de software desarrollados sobre esta Arquitectura y la UCI no quiere estar excluida de las prestaciones y beneficios que proporciona su utilización. Teniendo en cuenta los elevados índices de complejidad de los artefactos que se generan en los proyectos que trabajan sobre la Arquitectura MDA, se hace imprescindible asegurar su calidad de forma eficiente. Sin embargo, actualmente no existe un documento formal que cumpla con este requisito, lo que trae como consecuencia que no se logre asegurar la calidad de proyectos que utilicen dicha Arquitectura, y se corre el riesgo de que existan defectos en los programas, en los diseños e incluso en los requisitos, las especificaciones o en otra documentación de estos proyectos, reduciendo las capacidades de los programas para cumplir completa y efectivamente las necesidades de los usuarios.

Teniendo en cuenta la situación anteriormente planteada se traza el siguiente **problema de la investigación**:

¿Cómo contribuir al Aseguramiento de la Calidad de proyectos de software sobre la Arquitectura MDA?

Luego de ser analizado el problema existente de define como **objetivo general** de la investigación:
Elaborar un Plan de Aseguramiento de la Calidad adaptado a proyectos sobre la Arquitectura MDA.

Se plantean como **objetivos específicos**:

1. Caracterizar la Arquitectura MDA
2. Caracterizar actividades de Aseguramiento de la Calidad.
3. Identificar aspectos de la Arquitectura MDA sujetos al Aseguramiento de la Calidad.
4. Elaborar la propuesta del Plan de Aseguramiento de la Calidad para proyectos de software sobre la Arquitectura MDA.

Para dar cumplimiento a los objetivos establecidos se acometieron las siguientes **tareas**:

1. Estudiar distintas Herramientas para el Aseguramiento de Calidad.
2. Estudio de estándares, normas y modelos para el aseguramiento de la calidad de software.
3. Estudiar las características de la Arquitectura MDA.
4. Definir los objetivos de calidad para proyectos con Arquitectura MDA.
5. Definir la organización, las tareas y responsabilidades en áreas del Aseguramiento de Calidad.
6. Estudiar y proponer Métricas de Calidad.
7. Proponer Revisiones y Auditorías para un producto con Arquitectura MDA y estudiar el procedimiento de Revisiones y Auditorías que se realiza en la UCI.
8. Estudiar y proponer Pruebas para un producto con Arquitectura MDA.
9. Proponer Plan de Capacitación para los trabajadores de un producto con Arquitectura MDA.

Para llevar a cabo las tareas enunciadas se planteo el siguiente **objeto de estudio:**
Proceso de Aseguramiento de la Calidad de proyectos de software sobre la Arquitectura MDA.

Como parte del objeto de estudio que se va a investigar se definió el **campo de acción**:
Proyectos sobre la Arquitectura MDA en la UCI.

Se tiene la siguiente **idea a defender:** Con la elaboración de un Plan de Aseguramiento de la Calidad eficiente para los proyectos de software sobre la arquitectura MDA, se espera lograr el cumplimiento de las metas de trazadas, así como la entrega del producto en tiempo y con los niveles de calidad requeridos.

Debido a que los conocimientos que se tienen acerca del problema son suficientes para plantear una idea a defender y que se tiene una identificación clara de los aspectos externos del problema en cuestión y que los objetivos de esta investigación abarcan la caracterización de estructuras y funciones así como el establecimiento de correlaciones y clasificaciones entre los elementos del problema que necesitan ser resueltos, se ha decidido encaminar la investigación sobre una **estrategia descriptiva** que permita dar cumplimiento a los objetivos trazados y proponer una posible solución al problema.

Para el desarrollo de la investigación es necesaria la utilización de métodos científicos que proporcionen la obtención de información relacionada al problema. **Los métodos a utilizar son:**

- Los métodos teóricos lógicos y métodos teóricos históricos, dentro de los lógicos el método hipotético deductivo.
- Los métodos empíricos, específicamente la observación.

Estructuración de la tesis.

La tesis está estructurada en tres capítulos. El Capítulo I, referido al marco relacionado con el Aseguramiento de la Calidad y Arquitectura MDA, al estudio de diferentes definiciones y elementos necesario para la elaboración de un Plan de Aseguramiento de la Calidad para proyectos de software sobre la Arquitectura MDA. El Capítulo II, se realiza la propuesta del Plan de Aseguramiento de la Calidad para proyectos de software sobre la Arquitectura MDA. En el Capítulo III hace la validación de la propuesta del plan.

Capítulo 1: "Fundamentación Teórica".

1.1 Introducción.

En este capítulo se realiza un estudio sobre el estado de los elementos fundamentales que se deben de tener en cuenta para elaborar una propuesta de un Plan de Aseguramiento de la Calidad para proyectos sobre Arquitectura MDA (Model Driven Architecture). Además se argumentarán las definiciones necesarias para elaborar un Plan de Aseguramiento de la Calidad así como sus principales características y conceptos más importantes.

1.2 Calidad de Software.

Existen muchas definiciones de Calidad de Software según diversos autores reconocidos mundialmente. Una definición bien completa y abarcadora es brindada por Roger Pressman (PRESSMAN 1993) donde define Calidad de Software como: "…concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente".

Sin embargo, haciendo un resumen de las definiciones de autores como Humphrey (HUMPHREY), la norma ISO 8402 (ISO 1994) y el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, siglas de Institute of Electrical and Electronic Engineers) se puede llegar a un concepto más sencillo pero fácil de entender y que tiene implícito los principales aspectos para definir la Calidad de Software y concluir de que consiste en el grado en que un sistema, componente, proceso o entidad pueda satisfacer las necesidades de los usuarios y las implícitas haciendo el trabajo de una forma fiable y consistente. Esto requiere que el software desarrollado tenga pocos defectos.

Para construir y poder identificar un software de calidad en algunas bibliografías como *Informática. Temario para la preparación de oposiciones* (MARÍA LUISA GARZÓNVILLAR 2005) se establecen requisitos o parámetros que deben de cumplir estos productos para garantizar:

1. Concordancia del software con los requerimientos: El cliente desea que el software satisfaga una serie de requisitos o metas iniciales, y si ni siquiera se alcanzan estos objetivos, el software carecerá por completo de calidad.
2. Desarrollo coherente, aplicando correctamente los criterios de la ingeniería del software: Es uno de los objetivos de la ingeniería del software. Luego está claro que se debe seguir una metodología correcta y apropiada al proyecto, si se quiere aumentar la calidad del resultado final.
3. Desarrollo de requerimientos implícitos al proyecto: Siempre existen una serie de requerimientos que el cliente no especifica, pero que son deseables. Por ejemplo, que el software sea fácil de mantener, que sea fácil de usar, etc. Si no se alcanzan estos requerimientos, el software carecerá de calidad.

Entre las definiciones revisadas la más general y completa se encuentra en el Manual de Gestión de la Calidad Total a la Medida (MALEVSKI 1995) donde se define Calidad de Software como: "…el conjunto de propiedades y características de un producto o servicio, que le confieren la aptitud para satisfacer necesidades expresadas. Las necesidades pueden incluir aspectos relacionados con la aptitud para el uso, seguridad, disponibilidad, confiabilidad, mantenimiento, aspectos económicos y de medio ambiente. Este término no se emplea para expresar un grado de excelencia en un sentido comparativo, ni se usa con un sentido cuantitativo para evaluaciones".

Visto un conjunto de conceptos y definiciones de Calidad de Software se puede llegar a la conclusión que un producto de software que proporciona las prestaciones que son más importantes para los usuarios y que esté libre de defectos es un producto de calidad. Las necesidades de los usuarios se expresan en los requisitos, el desarrollo, clarificación y refinamiento de los requisitos es primordial para alcanzar la calidad del software.

1.2.1 Gestión de la Calidad.

Una definición de Gestión de la Calidad clara y sencilla de entender es ofrecida por la Norma ISO 9000 (ISO 2000): "…conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implanta por medios tales como la planificación de la calidad, el control de la calidad, el aseguramiento de la calidad y la mejora de la calidad, en el marco del sistema de calidad".

En la Norma ISO 9001 (ISO 2000) se expone una definición más general que la anterior: "…la gestión de la calidad es el conjunto de acciones, planificadas y sistemáticas, necesarias para dar la confianza adecuada de que un producto o servicio va a satisfacer los requisitos de calidad".

Para otros autores como Carlos López (LÓPEZ 2002): "…la gestión de la calidad incluye la planificación estratégica, la asignación de recursos y otras actividades sistemáticas, tales como la planificación, las operaciones y las evaluaciones relativas a la calidad".

En el Manual de Gestión de la Calidad Total a la Medida (MALEVSKI 1995): "…es aquel aspecto de función general de la gestión de una organización que define y aplica la política de calidad. La gestión de la calidad incluye la planificación, las asignaciones de recursos y otras actividades sistemáticas, tales como los planes de calidad".

Después de ver las definiciones anteriores, se puede concluir que la Gestión de la Calidad no son más que todas las actividades de la función gerencial realizadas de forma periódica que determinan la política de calidad, los objetivos y las responsabilidades y que los ponen en práctica por medios tales como la planificación, el aseguramiento, el control y el mejoramiento de la calidad, dentro del sistema de calidad.

1.2.2 Control de la Calidad.

Todos los conceptos de Control de la Calidad tienen como aristas fundamentales, controlar la evolución y desarrollo de los procesos y a su vez detectar y eliminar los errores de los mismos.

Muchos autores y bibliografías han definido Control de la Calidad, pero una definición bien abarcadora es brindada por Juan Manuel Cuevas Lovelle (LOVELLE 1999) coincidiendo con la Norma ISO 9001 (ISO 2000): "…son las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en dos objetivos fundamentales: mantener bajo control un proceso y eliminar las causas de los defectos en las diferentes fases del ciclo de vida. En general son las actividades para evaluar la calidad de los productos desarrollados".

Para Carlos López (LÓPEZ 2002) el Control de la Calidad lleva implícito la aplicación de técnicas operativas de actividades, que tienen dos objetivos fundamentales: mantener bajo control un proceso y eliminar las causas de defecto en las diferentes fases del bucle de la calidad, con el fin de conseguir los mejores resultados económicos.

1.2.3 Aseguramiento de la Calidad.

El Aseguramiento de la Calidad de los productos y servicios en los mercados internos e internacionales es hoy un factor decisivo en la subsistencia de las empresas.

A continuación se brindan definiciones de diversos autores que permiten apreciar el Aseguramiento de la Calidad y qué nivel de importancia se le otorga dentro del desarrollo de software.

Según Lovelle (LOVELLE 1999) el Aseguramiento de Calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto (software) satisfará los requisitos dados de calidad. Se diseña para cada aplicación antes de comenzar a desarrollarla y no después.

Algunos autores prefieren decir Garantía de Calidad en vez de Aseguramiento.

– Garantía, puede confundir con garantía de productos.

– Aseguramiento pretende dar confianza en que el producto tiene calidad.

Otra definición es consignada por Carlos López (LÓPEZ 2002): "…desde el punto de vista de la eficacia, el Aseguramiento de la Calidad implica generalmente, una evaluación permanente de aquellos factores que influyen en la adecuación del proyecto y de las especificaciones a las aplicaciones previstas y además, la verificación y la auditoría de las operaciones de producción, de instalación y de inspección. Para proporcionar la debida confianza, puede ser preciso que se aporten las pruebas oportunas".

Para Yoram Malevski (MALEVSKI 1995) el Aseguramiento de la Calidad: "…son todas aquellas acciones planificadas y sistemáticas necesarias para proporcionar la confianza adecuada de que un producto o servicio satisface los requisitos de calidad establecidos. Para que sea efectivo, el Aseguramiento de la Calidad requiere, generalmente, una evaluación permanente de aquellos factores que influyen en la adecuación del diseño y de las especificaciones según las aplicaciones previstas, así como también verificaciones y auditorías a las operaciones de producción, instalación e inspección. Dentro de una organización, el Aseguramiento de la Calidad sirve como una herramienta de la Gestión".

Pero una definición fácil de entender y muy abarcadora es brindada en el Estándar Internacional 730 de la IEEE (STD-730 1998): "…el Aseguramiento de la Calidad de software es un modelo planificado y sistemático de todas las acciones necesarias a fin de asegurar que el ítem o producto cumpla con los requerimientos técnicos establecidos".

Luego de analizar los conceptos anteriormente expuestos se puede afirmar que el Aseguramiento de la Calidad no está completo a menos que los requisitos de calidad reflejen completamente las necesidades del cliente. Para ser efectivo, requiere una evaluación continua de los factores que afectan a la calidad, así como el control de cambios o la gestión de la configuración y auditorías periódicas. Para estar convencidos si se logró asegurar la calidad de un software, existen métricas de calidad para realizarle mediciones al software y así evaluar los resultados alcanzados.

1.2.4 Métricas de Calidad.

Las métricas del software se refieren a un amplio elenco de mediciones. La medición se puede aplicar al proceso del software con el intento de mejorarlo sobre una base continua. Se puede utilizar en el proyecto del software para ayudar en la estimación, el control de calidad, la evaluación de productividad y el control de proyectos. Finalmente, el ingeniero de software puede utilizar la medición para ayudar a evaluar la calidad de los resultados de trabajos técnicos y para ayudar en la toma de decisiones tácticas a medida que el proyecto evoluciona.

Es difícil, y en algunos casos, imposible, desarrollar medidas directas de los factores de calidad del software.
Cada factor de calidad Fc se puede obtener como combinación de una o varias métricas:
Fc= c1* m1+ c2* m2+… + cn* mn
–ci: factor de ponderación de la métrica i, que dependerá de cada aplicación específica.
–mi: métrica i
–Habitualmente se puntúan de 0 a 10 en las métricas y en los factores de calidad.

Métricas para determinar los factores de calidad.
–Facilidad de auditoría.
–Exactitud.
–Normalización de las comunicaciones.
–Completitud.
–Concisión.
–Consistencia.

–Estandarización de los datos.
–Tolerancia de errores.
–Eficiencia de la ejecución.
–Facilidad de expansión.
–Generalidad.
–Independencia del hardware.
–Instrumentación.
–Modularidad.
–Facilidad de operación.
–Seguridad.
–Autodocumentación.
–Simplicidad.
–Independencia del sistema software.
–Facilidad de traza.
–Formación.

Hay cuatro razones para medir los procesos del software, los productos y los recursos (PRESSMAN 1993):

- Caracterizar: Se caracteriza para comprender mejor los procesos, los productos, los recursos y los entornos y para establecer las líneas base para las comparaciones con evaluaciones futuras.

- Evaluar: Se evalúa para determinar el estado con respecto al diseño. Las medidas utilizadas son los sensores que permiten conocer cuándo los proyectos y los procesos están perdiendo la pista, de modo que puedan poner bajo control. También se evalúa para valorar la consecución de los objetivos de calidad y para evaluar el impacto de la tecnología y las mejoras del proceso en los productos y procesos.

- Predecir: Se predice para poder planificar. Realizar mediciones para la predicción implica aumentar la comprensión de las relaciones entre los procesos y los productos y la construcción de modelos de estas relaciones, por lo que los valores que se observan para algunos atributos pueden ser utilizados para predecir otros. Esto se hace cuando se quiere establecer objetivos alcanzables para el coste, planificación, y calidad - de manera que se puedan aplicar los recursos apropiados-.Las medidas de predicción también son la base para la extrapolación de tendencias, con lo que la estimación para el coste, tiempo y calidad se puede actualizar basándose en la evidencia actual. Las proyecciones y las

estimaciones basadas en datos históricos también ayudan a analizar riesgos y a realizar intercambios diseño/coste.

- Mejorar: Se mide para mejorar cuando se recoge la información cuantitativa que ayuda a identificar obstáculos, problemas de raíz, ineficiencias y otras oportunidades para mejorar la calidad del producto y el rendimiento del proceso.

1.2.5 Plan de Calidad.

El Plan de Calidad se diseña orientado a poner en marcha un conjunto de actuaciones que faciliten la ruta hacia la Calidad Total. Establece el marco donde actuar y una guía de estrategias, actividades y recursos que aplicar durante un período concreto de tiempo.

Para argumentar más, en el Manual de Gestión de la Calidad Total a la Medida (MALEVSKI 1995) se define plan de calidad como:…"un documento que establece las prácticas específicas de calidad, recursos y secuencia de actividades relativas a un producto, servicio, contrato o proyecto, en particular".

El Plan de Aseguramiento de Calidad generado por un grupo de aseguramiento de un proyecto sirve como guía para el desarrollo e institución de las actividades de Aseguramiento de la Calidad. Un Plan de Aseguramiento debe contener por lo general (AGÜERO 2008):

- Objetivos de la calidad del proyecto y su enfoque para su consecución.
- Documentación referenciada en el Plan.
- Gestión del Aseguramiento de la Calidad.
- Documentación de desarrollo y de control o gestión.
- Estándares, normas o prácticas que hay que cumplir.
- Actividades de revisión y auditorías.
- Gestión de la Configuración del software.
- Informes de problemas.
- Pruebas.
- Herramientas, técnicas y métodos de apoyo.
- Recogida, mantenimiento y almacenamiento de datos sobre la documentación de actividades de Aseguramiento de la Calidad realizadas en el proyecto.

1.3 Gestión de la Configuración.

A lo largo del ciclo de vida del proceso de software, los productos de software evolucionan. Desde la concepción del producto y la captura de requisitos inicial hasta la puesta en producción del mismo, y posteriormente desde el inicio del mantenimiento hasta su retiro, se van realizando una serie de cambios, tanto en el código como en la documentación asociada. La Gestión de Configuración del Software es una disciplina encargada del control de la evolución de los productos de software.

El grupo Hista Internacional (INTERNACIONAL 2007) define la Gestión de Configuración como el proceso de identificar y definir los elementos en el sistema, controlando el cambio de estos elementos a lo largo de su ciclo de vida, registrando y reportando el estado de los elementos y las solicitudes de cambio, y verificando que los elementos estén completos y que sean los correctos.

Sin embargo, Rodolfo Villarroel Acevedo (ACEVEDO 2004) define de forma más detallada y explícita la Gestión de Configuración de Software (SCM) como una disciplina de la Ingeniería de Software que se preocupa en identificar y documentar las características funcionales y físicas de los elementos de configuración, de controlar los cambios a tales características y a reportar el proceso de tales cambios y su estado de implantación.

Pressman (PRESSMAN 1993) plantea:… “Si no controlamos el cambio el nos controlará a nosotros y esto nunca es bueno. Es muy fácil para un flujo de cambios incontrolados llevar al caos un proyecto de software”. Por esta razón la Gestión de Configuración es una actividad esencial del Aseguramiento de la Calidad y una práctica formal de la ingeniería del software.

1.4 Herramientas de Aseguramiento de la Calidad.

En cualquier proceso de mejora y de avance hacia la excelencia es necesario contar con diversos instrumentos que permitan ordenar, medir, comparar y estructurar información, de manera que permitan tanto generar nuevas ideas como resolver los diferentes problemas que se vayan presentando. Las herramientas para el Aseguramiento de la Calidad que a continuación se exponen son las más utilizadas habitualmente y se pueden aplicar en cada una de las fases y etapas de cualquier proceso de mejora para (FOMENTO 2008):

· Identificar y/o detectar problemas.
· Analizar los problemas y las causas.
· Toma de decisiones y selección de alternativas.
· Evaluación, control y seguimiento de acciones.

Existen un sin fin de herramientas para la calidad. De entre todas en esta investigación se hizo una selección de las más relevantes estructurándolas de la siguiente manera:
- Técnicas para la planificación.
- Técnicas para el control.
- Técnicas para la mejora y resolución de problemas.
- Técnicas de calidad en servicios.

Técnicas para la planificación

- Benchmarking: Proceso sistemático y continuo de medición y comparación de una organización con las mejores prácticas con el objetivo de obtener información que permita a la organización mejorar su desempeño.
- DOE: Design of Experiments, DEE, Diseño de Experimentos: Método empleado para la optimización de procesos. Con su implantación se reduce el número de pruebas, con lo que el desarrollo de productos puede ser organizado de forma más económica.
- FMEA: Failure Mode and Effects Analysis, AMFE**:** Análisis Modal de Fallos y Efectos: Método preventivo, cuyo uso sistemático permite la identificación e investigación de las causas y los efectos de los posibles fallos y debilidades en el producto o proceso y para la formulación de acciones correctivas tendentes a minimizar dichos efectos.
- QFD: Quality Function Deployment, Despliegue de la Función de Calidad: Técnica que identifica los requisitos del cliente y proporciona una disciplina para asegurar que estos requisitos estén presentes en el diseño del producto y en el proceso de planificación. Reduce los ciclos de desarrollo de productos, aumentando la calidad y disminuyendo los costes.

Técnicas para el control

- SPC: Statistical Process Control, CEP, Control Estadístico de Procesos: Herramienta para asegurar la calidad de los productos mediante el control de los procesos. Se sustituye la preocupación de

controlar el producto una vez fabricado por el interés de prevenir la aparición de defectos midiendo la aptitud de los procesos para producir productos conformes y combatiendo la variabilidad con el fin de obtener procesos estables en el tiempo. El CEP se basa en la utilización de gráficos de control que dependen del tipo de característica de estudio y de la naturaleza de cada proceso.

- Índices de capacidad:
  - Índice de capacidad de máquina: Herramienta que tiene como objetivo valorar la capacidad de calidad de una máquina comparando la dispersión generada por ésta con las tolerancias del parámetro a valorar.

    Cm: Compara la dispersión de la máquina con las tolerancias del parámetro.

    Cmk: Valora no sólo la dispersión sino también el centraje.
  - Índice de capacidad de proceso: Herramienta que tiene como objetivo valorar la capacidad de calidad de un proceso con respecto a un parámetro y periodo de tiempo determinados, estimando la dispersión generada por todos sus factores de variabilidad y comparándola con las tolerancias del parámetro.

    Cp: Compara la dispersión del proceso con las tolerancias del parámetro.

    Cpk: Valora no sólo la dispersión sino también el centraje.
- Auditoría de calidad: Examen metódico e independiente que se realiza para determinar si las actividades y los resultados relativos a la calidad satisfacen las disposiciones previamente establecidas y para comprobar que estas disposiciones se llevan a cabo eficazmente y que son adecuadas para alcanzar los objetivos previstos.

  Según su ámbito de aplicación se dividen en: auditorías de producto, auditorías de proceso y auditorías de sistema.

  Según su ámbito de actuación se dividen en: auditorías internas (realizadas por personal propio de la organización) y auditorías externas (llevadas a cabo por personal independiente de la organización).

<u>Técnicas para la mejora y resolución de problemas</u>

- Brainstorming o tormenta de ideas: Herramienta utilizada por un grupo de personas para aflorar el máximo número de ideas relacionadas con un concepto. Se basa en el respeto de todas las ideas de los participantes con la finalidad de estimular la participación y creatividad de todos los miembros del grupo.

- Reingeniería: Revisión fundamental y rediseño radical de procesos para alcanzar mejoras espectaculares en medidas críticas y contemporáneas de rendimiento, tales como costes, calidad, servicio y rapidez.
- Ciclo PDCA (Plan, Do, Check, Act): Ciclo de planificación, realización, control y actuación que actúa como guía para llevar a cabo la mejora continua y lograr de una forma sistemática y estructurada la resolución de problemas.
- Diagrama de flujo: Herramienta utilizada para representar, mediante la utilización de símbolos estándares, las secuencias e interrelaciones de actividades que conforman un proceso (Anexo 1).
- Histograma: Gráfico de barras que muestra de forma visual la distribución de frecuencias de datos cuantitativos de una misma variable (Anexo 2).
- Diagrama de correlación o de dispersión: Gráfico que muestra la existencia o no de una relación entre dos variables (Anexo 3).
- Diagrama de Pareto: Gráfico de barras organizado de mayor a menor frecuencia, que compara el nivel de importancia de todos los factores que intervienen en un problema o cuestión (Anexo 4).
- Diagrama de Ishikawa, diagrama causa-efecto o diagrama de espina de pez: Representación gráfica de las relaciones lógicas que existen entre las causas y subcausas que producen un efecto determinado (Anexo 5).
- Cartas de control: Representación gráfica de los distintos valores que toma una característica correspondiente a un proceso. Permite observar la evolución de este proceso en el tiempo y compararlo con unos límites de variación fijados de antemano que se usan como base para la toma de decisiones.
- Diagrama de árbol: Herramienta empleada para ordenar de forma gráfica las distintas acciones o gestiones que se deben llevar a cabo para solventar el problema o situación sometida a estudio (Anexo 6).
- Diagrama matricial: Herramienta que ordena gráficamente grupos de datos representando los puntos de conexión lógica existentes entre ellos. Las disposiciones más comunes son: diagrama matricial en “L”, diagrama en “A” o matriz triangular; diagrama matricial en “T”, diagrama matricial en “Y” y diagrama matricial en “X”.
- Diagrama matricial para el análisis de datos o matrices de priorización: Herramienta empleada para la toma de decisiones en base a la priorización de actividades, temas, características de productos, etc., según criterios de ponderación conocidos. Se utiliza una combinación de las técnicas de diagrama de árbol y diagrama matricial.

- Diagrama de decisión: Herramienta cuyo objetivo es identificar, representar y eliminar todos los problemas posibles que pueden suceder en el proceso de implantación de soluciones a un problema.
- Diagrama de flechas: Herramienta utilizada para planificar y controlar el desarrollo y progreso de cualquier actividad mediante una representación de red.

Técnicas de calidad en servicios

Las técnicas de calidad vistas hasta ahora han ido surgiendo en entornos industriales. Algunas de ellas se han adaptado para aplicarlas a servicios, pero además han ido apareciendo otra serie de técnicas de la calidad específicas para los servicios. A continuación se enumeran las principales técnicas de calidad aplicables a los servicios:

- Modelo GAP: Se basa en que el cliente percibe la calidad de un servicio como la diferencia entre lo que espera del mismo y lo que realmente percibe. Esta diferencia es la suma de una serie de diferencias parciales:

  Gap 1: Diferencia entre el servicio esperado por el cliente y lo que la dirección percibe que el cliente espera.

  Gap 2: Diferencia entre lo que la dirección percibe que el cliente espera y las especificaciones que se marcan para el servicio.

  Gap 3: Diferencia entre las especificaciones y el servicio realizado.

  Gap 4: Diferencia entre el servicio realizado y el servicio percibido por el cliente.
- Técnica de la viñeta: Técnica empleada en el desarrollo de nuevos servicios. El método consiste en elaborar varias variantes para el servicio (viñetas), a partir de las características más relevantes para los clientes. Dichas viñetas se emplean para realizar una encuesta y elegir la opción favorita de los clientes.
- Blueprinting: Método empleado para localizar las posibles fuentes de fallos de un servicio a partir de la representación gráfica del mismo. Ayuda a juzgar la calidad de un servicio.
- Método secuencial de incidentes: Método empleado para conocer la opinión del cliente en cada fase del proceso. Se determinan las fases del proceso y se recogen los comentarios de los clientes, tanto favorables como desfavorables, sobre cada una de estas fases. De esta forma se consigue un conocimiento más profundo del servicio.
- Encuestas a los clientes: Desarrollo de cuestionarios cuyo análisis ayuda a conocer mejor y a acercarse más a la identificación de las necesidades y expectativas de los clientes.

- Serv-Qual: Método empleado para medir la satisfacción del cliente con el servicio y priorizar las acciones de mejora. Al cliente se le pregunta la importancia que para él tiene cada uno de los atributos del servicio recibido y el grado de satisfacción con cada uno de estos atributos. Estos datos (importancia del atributo y prestación recibida) se representan en un diagrama, llamado diagrama IP (Importance, Performance), para determinar el orden de prioridades en la actuación para la mejora del servicio.
- Análisis de relevancia de frecuencias: Método empleado para establecer prioridades. Tras elaborar una lista con los posibles problemas, se elabora una encuesta en la que se pregunta a los clientes la frecuencia de aparición/detección y la importancia de los problemas planteados. Se procede al análisis de los datos recogidos y a su visualización para determinar prioridades y sacar una serie de conclusiones sobre la actuación.
- AMFE para servicios: Método empleado para la prevención y el perfeccionamiento del servicio. Se buscan los posibles errores del proceso y las posibles consecuencias de dichos errores. A continuación se trata de buscar las posibles causas de los errores y se elabora un plan de acción para eliminarlas.
- Gestión de quejas: Acciones sistemáticas y estructuradas que llevan a cabo las organizaciones para la recogida, evaluación y búsqueda de soluciones a las quejas de sus clientes.

1.5 Modelos de Calidad.

A nivel mundial, existen múltiples metodologías modernas de ingeniería de software que son prácticamente desconocidas. Las empresas que desarrollan o utilizan software, requieren de apoyo tecnológico importante para llevar a cabo procesos de adopción de metodologías modernas para desarrollo y Aseguramiento de la Calidad del software.

Los modelos de calidad pueden ser utilizados para construir mejores productos y asegurar su calidad.

Implantar modelos tiene como objetivo principal que las empresas desarrollen sistemáticamente, productos, bienes y servicios de mejor calidad y cumplan con las necesidades de deseos de los clientes. Para eso, se requiere de un modelo que permita unir la misión de la empresa y el esfuerzo de cada área en una sinergia de resultados hacia la competitividad y la calidad de clase mundial; y tener procesos y procedimientos ágiles; y comprensibles para todos los involucrados, pasando por las etapas de desarrollo, prueba, producción y satisfacción al cliente.

Por la importancia de la aplicación de modelos de calidad para construir mejores productos y asegurar su calidad se hace a continuación un estudio de distintos modelos en aras de elaborar un Plan de Aseguramiento de la Calidad.

1.5.1 ISO.

ISO es la denominación que recibe la Agencia Internacional de Normalización (International Organization for Standarization) que agrupa en su seno cerca de cien países.

- ISO 8402

El ISO 8402, en su glosario de términos, define el Aseguramiento de la Calidad de la siguiente manera: "…todas las acciones sistemáticamente planificadas en una empresa, necesarias para proveer una adecuada confianza de que los productos o servicios puedan satisfacer determinados requerimientos de Calidad".

- ISO 12119

Se definen los requisitos de calidad e instrucciones para la realización de pruebas. Además de explicar diferentes métodos de testeo, determina los detalles que debe respetar todo paquete de software. Algunos de ellos son:

- Documentación para el usuario de fácil comprensión, que incluya un índice y un manual de instalación.
- Que el software le informe si el programa fue correctamente instalado.
- Función de ayuda con recursos de hipertextos.
- Mensajes de error con información necesaria para su solución.
- Identificación de los archivos del sistema operativo usados por el programa.
- Capacidad de interrumpir un proceso demorado, sin que se cuelgue el equipo.
- Posibilidad de anular funciones de efectos irreversibles, como borrar.

- Serie ISO 9000

La importancia de la aplicación de las normas ISO 9000 para el desarrollo e implementación de sistemas de aseguramiento de la calidad radica en que son normas prácticas, no normas académicas. Por su sencillez han permitido su aplicación generalizada sobre todo en pequeñas y medianas empresas.
Siendo la calidad hoy uno de los factores esenciales de la competencia en cualquier actividad, se ha generado la necesidad de implementar sistemas normalizados de aseguramiento de la calidad. Las normas ISO 9000 brindan el marco que permite evaluar razonablemente por parte de terceros la efectividad del sistema.

¿Qué son las normas ISO Serie 9000?

La serie ISO 9000 es un conjunto de cinco normas relacionadas entre sí, son normas genéricas, no específicas que permiten ser usadas en cualquier actividad ya sea industrial o de servicios.
La importancia de la aplicación de las normas ISO 9000 para el desarrollo e implementación de sistemas de aseguramiento de la calidad radica en que son normas prácticas. Por su sencillez han permitido su aplicación generalizada sobre todo en pequeñas y medianas empresas.
Brindan el marco para documentar en forma efectiva los distintos elementos de un sistema de calidad y mantener la eficiencia del mismo dentro de la organización.

La serie de normas ISO destinadas al aseguramiento de la calidad esta formada por distintas normas armonizadas entre sí. Las mismas son:

- ISO 9000

Cumple el papel de eje distribuidor y distribuidor del sistema. Expone el alcance real de la serie. Define la filosofía general de las normas los distintos tipos, niveles y pautas para la aplicación de las distintas normas.

- ISO 9001

Se aplica cuando la empresa debe responsabilizarse por todas las etapas del ciclo, es decir: diseño, desarrollo y elaboración.

- ISO 9002

Se aplica cuando las características del bien o servicio son definidas por el cliente.

- ISO 9003

Cubre las obligaciones de aseguramiento de calidad en las áreas de control final y pruebas. Es de limitada aplicación por lo que existen planes para su eliminación.
En los casos de exigencia contractual las normas aplicables son las normas ISO 9001/2/3. La norma a aplicar depende del alcance de la actividad de la empresa, no de una elección a voluntad.

- ISO 9004-1/ ISO 9004-2

Establecen condiciones y pautas para guiar a las empresas en la implementación de su propio sistema de aseguramiento de calidad. Su desarrollo no es válido para certificación o registro.

Complementan la serie de normas ISO 9000 las siguientes:

- ISO 8402

Vocabulario. Clarifica y normaliza los términos relativos a la calidad.

- ISO10011-1

Auditoría. Establece los principios básicos, criterios y prácticas de una auditoría y provee lineamientos para establecer, planificar, realizar y documentar auditorías de sistemas de la calidad.

- ISO10011-2

Criterios para la calificación de auditores. A fin de que las auditorías de los sistemas de calidad sean conducidas en forma uniforme y efectiva se ha desarrollado esta norma que constituye una guía sobre los criterios de calificación de auditores.

- ISO10011-3

Gestión de programas de auditoría. Define los lineamientos básicos para administrar programas de auditorías de sistemas de la calidad.

- ISO10013

Guía para la elaboración de manuales de calidad.

Además de estas existe la norma ISO 9126 (9126 2001) con cada una de sus partes.

- ISO 9126-1

Modelo de Calidad. Define cada unos de los objetivos de calidad. Sintetiza una serie de características que deben reunir los programas para que sean considerados de calidad.

- ISO 9126-2

Métricas Externas. Define métricas para evaluar el comportamiento del sistema mediante el ensayo.

- ISO 9126-3

Métricas Internas. Define métricas para aplicarse a productos de software no ejecutables (como especificación o código fuente) durante el diseño y la programación. Lo que permite evaluar la calidad desde las etapas más tempranas.

- ISO 9126-4

Métricas de Calidad en el Uso. Define métricas para medir hasta que punto el producto satisface las necesidades de los usuarios para lograr las metas especificadas con la efectividad, productividad, seguridad y satisfacción en un contexto determinado de uso.

1.5.2 IEEE.

Dentro de la serie de normas del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) sobre software algunos estándares ANSI/IEEE están orientados al Aseguramiento de la Calidad a nivel del proyecto (GARCÍA 2005).

Estándar 730: Proporciona la estructura de la documentación del Plan de Aseguramiento de la Calidad.
Estándar 1061: Definición de métricas para productos y para procesos, así como procedimientos para la recogida de valores de métricas.

Existen también estándares para otras actividades relacionadas con la calidad como pruebas, verificación y validación, revisiones, etc. Los principales se recogen en la Tabla 1.

IEEE 730-1998	Planes de aseguramiento de la calidad del software.
IEEE 829	Documentación de pruebas del software.
IEEE 982.1, 982.2	Diccionario estándar de medidas para producir software fiable.
IEEE 1008-1987	Pruebas de unidad del software.
IEEE 1012-1998	Verificación y validación del software.
IEEE 1028-1997	Revisiones y Auditorías del software.
IEEE 1044-1993	Clasificación estándar para anomalías del software.
IEEE 1061-1992	Estándar para una metodología de métricas de calidad del software.
IEEE 1228-1994	Planes de Seguridad del Software.

Tabla 1: Estándares de la IEEE.

IEEE STD-828: Estándar para planes del manejo de las configuraciones de software. El Plan de Gestión de la Configuración software describe el proceso de Gestión de Configuración software.

IEEE STD-829: Estándar para documentación de pruebas de software.
Este estándar describe un conjunto de documentos de prueba que están asociadas con los aspectos dinámicos de software ensayo (es decir, la ejecución de los procedimientos y el código).
El propósito del Plan de Pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables y manejo de riesgos de un proceso de pruebas.

IEEE STD-1012: Estándar para la planificación de verificación y validación de software. Los planes de validación y verificación se utilizan para determinar si el producto software desarrollado se ajusta a sus requisitos, y si cumple con las expectativas del usuario.

IEEE STD-1028: Estándar para Revisiones y Auditorías del Software.

- Define procedimientos para definir y llevar a cabo procesos de revisión y auditoría del software.
- Describe cinco tipos de revisiones y auditorías que se pueden utilizar.
- Incluye tanto al producto como al proceso de software.
- No prescribe el uso de revisiones ni auditorías particulares.

1.5.3 CMMI.

¿Qué es el CMM - CMMI?

El CMM – CMMI (Capability Maturity Model Integration) es un modelo de calidad del software que clasifica las empresas en niveles de madurez. Estos niveles sirven para conocer la madurez de los procesos que se realizan para producir software.

Niveles CMM – CMMI

Los niveles CMM - CMMI son 5:

- Inicial o Nivel 1 CMM - CMMI.

Este es el nivel en donde están todas las empresas que no tienen procesos. Los presupuestos se disparan, no es posible entregar el proyecto en fechas, hay que trabajar durante noches y fines de

semana para terminar un proyecto. No hay control sobre el estado del proyecto, el desarrollo del proyecto es completamente opaco, no se sabe lo que pasa en él.
Si no se sabe el tamaño del proyecto y no se sabe cuanto se lleva hecho, nunca se sabrá cuando se va a terminar.

- Repetible o Nivel 2 CMM - CMMI.

Quiere decir que el éxito de los resultados obtenidos se puede repetir. La principal diferencia entre este nivel y el anterior es que el proyecto es gestionado y controlado durante el desarrollo del mismo. El desarrollo no es opaco y se puede saber el estado del proyecto en todo momento.

Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión de Requisitos.
- Planificación de Proyectos.
- Monitorización y Control de Proyectos.
- Medición y Análisis.
- Aseguramiento de la Calidad.
- Gestión de la Configuración.

- Definido o Nivel 3 CMM - CMMI.

Resumiéndolo mucho, alcanzar este nivel significa que la forma de desarrollar proyectos (gestión e ingeniería) esta definida, definida quiere decir que está establecida, documentada y que existen métricas (obtención de datos objetivos) para la consecución de objetivos concretos.

Los procesos que hay que implantar para alcanzar este nivel son:

- Desarrollo de requisitos.
- Solución Técnica.
- Integración del producto.
- Verificación.
- Validación.
- Desarrollo y mejora de los procesos de la organización.
- Definición de los procesos de la organización.
- Planificación de la formación.
- Gestión de Riesgos.

- Análisis y resolución de toma de decisiones.

La mayoría de las empresas que llegan al nivel 3 paran aquí, ya que es un nivel que proporciona muchos beneficios y no ven la necesidad de ir más allá porque tienen cubiertas la mayoría de sus necesidades.

- Cuantitativamente Gestionado o Nivel 4 CMM - CMMI.

Los proyectos usan objetivos medibles para alcanzar las necesidades de los clientes y la organización. Se usan métricas para gestionar la organización.

Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión cuantitativa de proyectos.
- Mejora de los procesos de la organización.

- Optimizado o Nivel 5 CMM - CMMI.

Los procesos de los proyectos y de la organización están orientados a la mejora de las actividades. Mejoras incrementales e innovadoras de los procesos que mediante métricas son identificadas, evaluadas y puestas en práctica.

Los procesos que hay que implantar para alcanzar este nivel son:

- Innovación organizacional.
- Análisis y resolución de las causas.

Normalmente las empresas que intentan alcanzar los niveles 4 y 5 lo realizan simultáneamente ya que están muy relacionados.

CMM-CMMI: Aseguramiento de la calidad

El objetivo del Aseguramiento de la Calidad es proporcionar personas y gestión con el objetivo de que los procesos y los elementos de trabajo cumplan los procesos.
Esto se consigue mediante:

- Evaluar objetivamente la ejecución de los procesos, los elementos de trabajo y servicios contra las descripciones de procesos, estándares y procedimientos.

- Identificar y documentar los elementos no conformes.
- Proporcionar información a las personas que están usando los procesos y a los gestores, de los resultados de las actividades del aseguramiento de la calidad.
- Asegurar de que los elementos no conformes son arreglados.

Esta es un área de proceso clave, que a veces no se le da la suficiente importancia, pero que sin ella no será posible implanta un modelo de calidad. (GRACIA 2005).

1.6 Arquitectura de Software.

La arquitectura de un sistema es la especificación de las partes del mismo, las conexiones entre ellos, y las normas de interacción entre las partes del sistema haciendo uso de las conexiones especificadas.

Una definición fácil de entender y bien detallada es brindada por Luis E. Corredera (COLSA): "…la arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad".

Toda arquitectura de software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de ellos constituye una descripción parcial de una misma arquitectura y es deseable que exista cierto solapamiento entre ellos. Esto es así porque todas las vistas deben ser coherentes entre sí, evidente dado que describen la misma cosa.

1.6.1 Arquitectura MDA.

Producto de las mejores prácticas en Ingeniería de Software surge la Arquitectura Dirigida por Modelos (MDA – Model Driven Architecture).
(CORREDERA 2006)

¿Qué es MDA?

MDA es el acrónimo de Model Driven Architecture (Arquitectura Dirigida por Modelos), un concepto promovido (pero no creado) por la OMG, que propone basar el desarrollo de software en modelos especificados utilizando UML, para que, a partir de esos modelos, se realicen trasformaciones que generen código u otro modelo, con características de una tecnología particular (o con menor nivel de abstracción). MDA define un framework para procesar y relacionar modelos. Suele escucharse que MDA es la evolución natural de UML, ya que tiende a incrementar la cantidad de código generado, a partir de especificaciones detalladas en UML.

¿Qué no es MDA?

Hoy MDA es uno de los tantos acrónimos de moda y, como ocurre algunas veces con las modas, el concepto puede tender a malinterpretarse. Por lo tanto, se debe conocer que:

- MDA no es un proceso de desarrollo.
- MDA no es una especificación.
- MDA no es una implementación.
- MDA no es una implementación de referencia de ningún estándar particular.
- MDA no es un concepto maduro aún.
- MDA no es simplemente generar código.
- MDA no tiene, aún, una visión unificada en la industria.

Modelos MDA

Los modelos juegan un rol trascendental en MDA. Como un framework para construir sistemas, MDA abstrae el sistema a construir en distintas capas de abstracción. Tradicionalmente, el OOAD (Object Oriented Analysis and Design) contiene, entre otros, una vista de análisis, una vista de diseño detallado y código -representando la vista de negocios de un sistema-, la vista de arquitectura y la vista de implementación. MDA agrega una capa de abstracción más, que representa el contexto de negocio del sistema. En la Figura 1 se muestran las diferentes capas de abstracción (layers). El gráfico se hace más abstracto hacia la izquierda y se vuelve más concreto hacia la derecha.

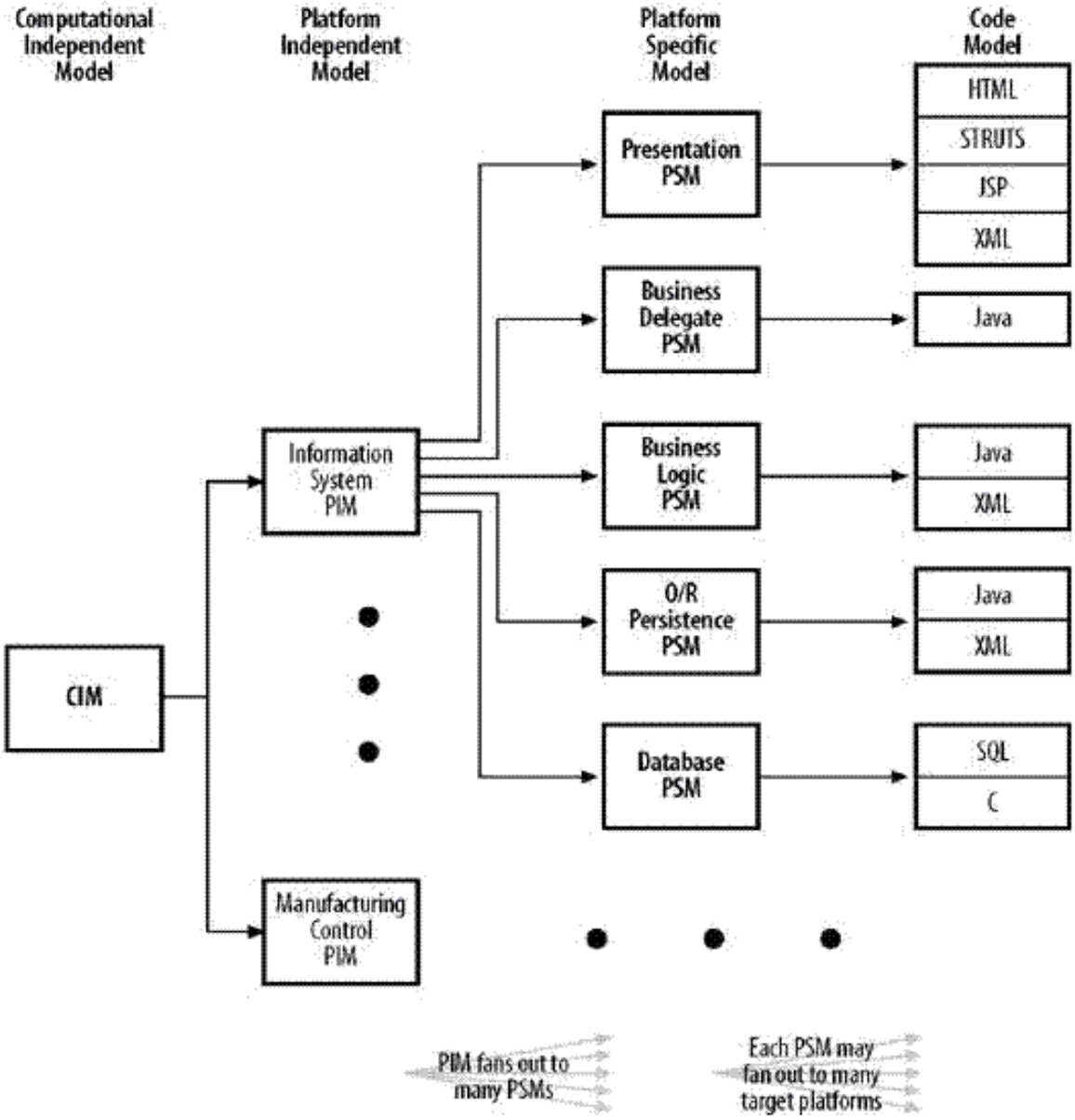


Figura 1: Un ejemplo de modelo MDA y sus relaciones.

Los modelos concretos exceden en número a los modelos abstractos. A medida que avanzamos en las transformaciones, los modelos se vuelven más concretos, transformando al modelo abstracto en uno compatible con una tecnología o plataforma. La situación inversa de llevar el código hacia un modelo concreto -también conocido como ingeniería reversa- rara vez ocurre, excepto cuando el punto de partida es el código mismo. Esto se produce debido a que MDA promueve la fuerte separación entre las responsabilidades de requerimientos del negocio y las responsabilidades tecnológicas. La ventaja de esta "separación de responsabilidades" es que ambos aspectos pueden evolucionar

individualmente sin generar dependencias entre sí. De está manera, la lógica de negocio responderá a las necesidades del negocio y no dependerá de vicisitudes técnicas.

- CIM (Computational-independent model)

CIM debe su nombre a este foco en el negocio por sobre la tecnología, que en español se traduce como: “Modelo Independiente de la Computación”.
El CIM se centra en los requerimientos y representa el nivel más alto del modelo de negocios. Usa un lenguaje para modelar procesos de negocios que no es UML, aunque este lenguaje puede ser derivado perfectamente utilizando MOF (meta-object facility). El CIM transciende a los sistemas; cada proceso de negocio interactúa con trabajadores humanos y/o componente de máquina. El CIM describe solamente aquellas interacciones que tienen lugar entre los procesos y las responsabilidades de cada trabajador, sea o no humano. Un objetivo fundamental del CIM, es que cualquiera que pueda entender el negocio y los procesos del mismo puede comprenderlo, ya que éste evita todo tipo de conocimiento especializado o de sistemas.

- PIM (Platform-independent model)

El PIM, que se traduce al castellano como “Modelo Independiente de la Plataforma”, representa el modelo de procesos de negocio a ser implementado. Comúnmente se usa UML o un derivado de UML para describir el PIM. El PIM modela los procesos y estructuras del sistema, sin hacer ninguna referencia a la plataforma en la (o las) que será desplegada la aplicación. A su vez, ignora los sistemas operativos, los lenguajes de programación, el hardware y la topología de red. Suele ser el punto de entrada de todas las herramientas para MDA e incluso de muchos artículos que hablan de MDA, dejando de lado el CIM.

- PSM (Platform-specific model)

El PSM, que se traduce al castellano como “Modelo Específico de la Plataforma”, representa la proyección de los PIMs en una plataforma específica. Un PIM puede generar múltiples PSMs, cada uno para una tecnología distinta. Generalmente, los PSMs deben colaborar entre sí para una solución completa y consistente. Normalmente, esto se realiza en UML, creando distintos profiles que definen

un PSM para cada tecnología requerida. Los PSMs tienen que lidiar explícitamente con los sistemas operativos, los lenguajes de programación, las plataformas (CORBA, .Net, J2EE, ETC), etc.

- Code model

El modelo de código representa el código desplegable (deployable), normalmente en un lenguaje de programación de alto nivel, como Java, C#, C++, VB, JSP, etc. Idealmente, el modelo de código está listo para compilar y no debería requerir la intervención humana; el despliegue de la aplicación podría ser automatizado. Según los puristas y algunos fanáticos de MDA, en un ambiente MDA maduro no se debería pensar en el código más que como simples archivos, o como un mero objeto intermedio para generar el ejecutable final. Pero debido a que MDA no está maduro, y difícilmente se llegue alguna vez a la utopía de no tener que tocar ningún código, los desarrolladores seguirán necesitando conocer la tecnología para complementar la generación de código, debuguear la aplicación y, sobre todo, lidiar con muchos y variados errores inesperados, extraños y divertidos.

Decisiones de Diseño

MDA promueve la transformación de modelos que representan lógicas de negocios complejas (CIM), hasta llegar al código ejecutable y desplegable (Code Model). Pero, ¿qué pasa con las decisiones de diseño, aquellas decisiones que se toman cuando, por ejemplo, se tiene que desarrollar un sistema donde la mayoría de las transacciones sólo leen datos, pero diez de ellas hacen uso intensivo del procesador? ¿Qué pasaría si todos los mapeos se hicieran de igual manera? Se pudiera pensar que esto degradaría la calidad final percibida de la aplicación. Para corregir este problema, MDA promueve el uso de marcas (Marks), las cuales indican aspectos específicos para tener en cuenta en cada transformación. Un PIM generado a partir de un PIM y Marcas se denomina PIM Marcado o Marked PIM. La utilización de las marcas establece que las decisiones respecto a aspectos tecnológicos queden fuera de los modelos principales.

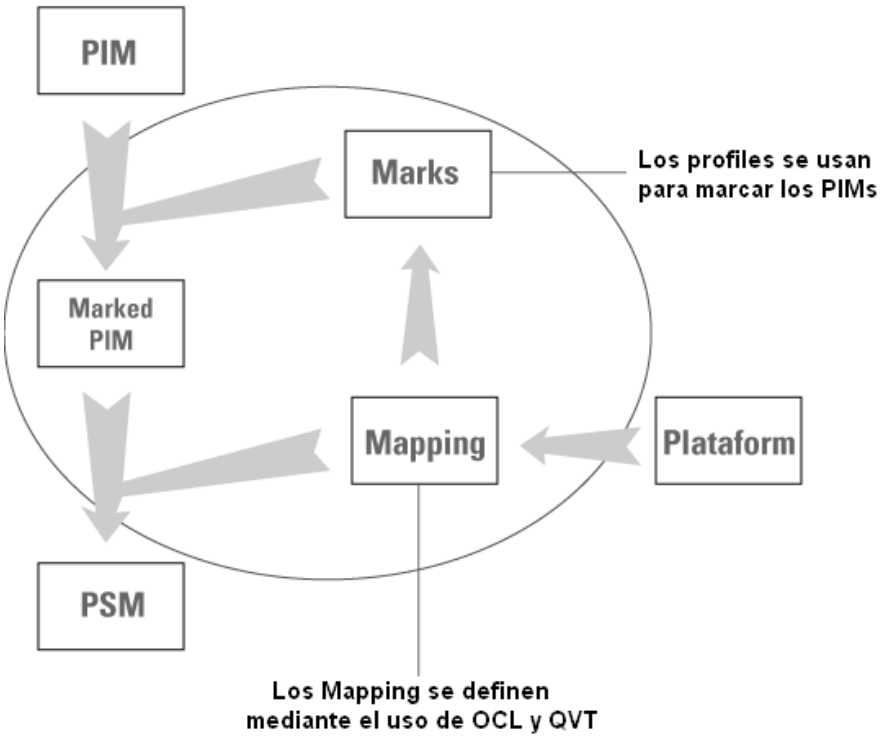


Figura 2: Utilización de marcas y mapeos.

Ahora bien, para realizar el mapeo entre un PIM marcado y, por ejemplo, un PSM, es necesario detallar cómo se mapean esas marcas; para eso se definen los Mappers (mapeadores). Puede verse la relación entre los mapeadores, las marcas y los PSM en la figura anterior. Los mapeos se hacen utilizando un QVT: Query, Views, and Transformations.

Ventajas de MDA

La ventaja principal de MDA radica en la clara y estricta separación de responsabilidades. Por un lado, modelar los PIMs, que representan los modelos del negocio, y por otro lado, los PSMs con las preocupaciones tecnológicas. Esto permitirá que ambos modelos puedan evolucionar por separado. De esta manera, si se quisiera, por ejemplo, modificar un aspecto técnico, bastará con modificar el PSM sin que estos tengan impacto en la lógica de negocios. Esta idea viene de un concepto que, en ingeniería de software, se llama “Guías de Diseño”. Particularmente, una de esas guías dice que el modelado de la solución debe ser dirigido por el negocio. Esta guía se basa en la afirmación de que un cambio en el negocio seguramente produzca un cambio en el código, pero no lo inverso: los cambios

en el código no deberían impactar en el negocio. MDA también permite: lidiar con la complejidad del negocio, modelando a éste por separado y permitiendo su análisis y mejora; disminuir costos, si se cuenta con una herramienta MDA adecuada a nuestras necesidades; mejorar la calidad de nuestros modelos y procesos, mediante su análisis y la separación de responsabilidades.

Estándares involucrados en MDA

Las tecnologías más importantes involucradas, para poder llevar a la práctica los conceptos subyacentes en MDA son: Meta Object Facility (MOF), Unified Modeling Language (UML), XML Model Interchange (XMI), Common Warehouse Meta-model (CWM), Software Process Engineering Meta-model (SPEM), Action Semantics Language (ASL), Query-View-Transformation (QVT), UML profiles.

Herramientas MDA

Las herramientas MDA deberían proveer la capacidad de transformar modelos de negocios puro (CIMs) en aplicaciones completas, desplegables y capaces de ejecutar con un mínimo de decisiones técnicas. Lamentablemente, nos encontramos muy lejos de que MDA o alguna herramienta MDA provea todas estas facilidades para cualquier negocio y problemática a desarrollar. Tampoco existe un líder en herramientas MDA, debido a que son muy sofisticadas. Algunas de las herramientas más conocidas son:

1. ATL ATLAS Transformation Language
2. OptimalJ is a MDA tool for J2EE.
3. ArcStyler is a MDA tool for J2EE and .NET.
4. UMT UML Model Transformation
5. ArgoUML
6. Codagen
7. Rational Architect
8. MDA Transf
9. Enterprise Architect
10. GReAT
11. AndroMDA

La importancia de MDA para el desarrollador

Actualmente, MDA es importante para el desarrollador promedio. Quienes son bendecidos con la capacidad de utilizar la creatividad para transformar ceros y unos en aplicaciones útiles para nuestros clientes. En los últimos dos años, muchas organizaciones han comenzado a prestar atención a MDA, ya que promueve el uso eficiente de modelos de sistemas en el proceso de desarrollo de software.

MDA representa para los desarrolladores, una nueva manera de organizar y administrar arquitecturas empresariales, basada en la utilización de herramientas de automatización de etapas en el ciclo de desarrollo y servicios. De esta forma, permite definir los modelos y facilitar trasformaciones paulatinas entre diferentes modelos. Es decir que, a partir de un modelo, se puede generar otro de menor abstracción.

Un ejemplo común de generación de modelos, es la generación de código a partir del modelo de diseño, mediante el uso de una herramienta de modelado UML. En este caso se usa una herramienta para transformar el modelo de diseño en el "modelo" de código.
(ANACLETO 2006)

1.7 Conclusiones.

Como se ha visto en el desarrollo del presente capítulo, lograr asegurar la calidad de un producto de software no es tarea fácil pero si debe de ser prioridad para sus desarrolladores puesto a su papel primordial dentro del proceso de desarrollo de software. El Aseguramiento de la Calidad es una disciplina estudiada por diversos autores como Pressman, Humphrey, y otros especialistas en el tema, ha sido abordada en modelos y estándares utilizados a nivel mundial como ISO, IEEE, CMMI, entre otros. Entre los estándares estudiados se utilizarán para la elaboración de la propuesta aquellos que abordan aspectos significativos en su estructuración como:
Los estándares ISO 9126-1, ISO 9126-2, ISO 9126-3 e ISO 9126-4, donde se definen los objetivos y las métricas que pueden ser utilizadas.
Los estándares IEEE 1028, ISO 10011-1 e ISO 10011-2 donde se hace referencia a las revisiones y auditorías a realizar en un producto de software, y auditores responsables de su ejecución.

El estándar IEEE 730 donde se proporciona toda la documentación para la elaboración de un Plan de Aseguramiento de la Calidad.

El estándar IEEE 828 será utilizado en el proceso de gestión de configuración dentro de la propuesta.

El estándar IEEE 829 se utilizara para definir las pruebas a las que será sometido el producto una vez que se le aplique la propuesta del plan.

Capítulo 2: "Solución Propuesta del Plan de Aseguramiento de la Calidad".

2.1 Introducción.

En este capítulo se realiza una propuesta del Plan de Aseguramiento de Calidad para proyectos de software sobre la Arquitectura MDA, con cada uno de los elementos necesarios a tener en cuenta para su elaboración y adaptados a las características específicas que posee la Arquitectura MDA.

2.2 Objetivos de Calidad.

En esta sección se incluyen los requerimientos del producto de software que están alineados con los requerimientos de calidad.

La calidad del producto del software se debe evaluar usando un modelo definido. El modelo de calidad se usará al fijar los objetivos de calidad para los productos del software y productos de software intermedios.

En la NC-ISO/IEC 9126-1 (9126-1 2005) se describe un modelo en dos partes para calidad de los productos de software: a) calidad interna y externa y b) calidad durante el uso. La primera parte del modelo especifica seis características para la calidad interna y externa, que son además divididas en sub-características que se manifiestan externamente cuando el software se usa como una parte del sistema computarizado, y son un resultado de los atributos internos del software. La segunda parte del modelo especifica cuatro características de calidad durante el uso. La calidad durante el uso es el efecto combinado para el usuario de las seis características de calidad del producto de software.

Calidad interna: Total de características del producto de software desde una perspectiva interna. La calidad interna se mide y se evalúa con respecto a los requisitos de la calidad interna. Durante la aplicación del código, la revisión y el ensayo se pueden mejorar algunos detalles de la calidad del producto de software, pero la naturaleza básica de la calidad del producto de software representada

por la calidad interna, permanece invariable a menos que sean objeto de otro diseño. Las partes del modelo se definen como:

Calidad externa: Total de características del producto de software desde una perspectiva externa. Es la calidad que se obtiene cuando se ejecuta el software, y por lo general se mide y evalúa mientras se somete a ensayo en un ambiente simulado, con datos simulados, y utilizando métricas externas. Durante el ensayo deberá ser posible detectar y eliminar la mayor parte de los defectos. Sin embargo, es posible que queden algunos después del ensayo. Debido a que resulta difícil corregir la arquitectura del software u otros aspectos fundamentales del diseño del software, el diseño básico permanece por lo general invariable durante todo el ensayo.

Calidad estimada (o pronosticada) durante el uso: Calidad que se estima o pronostica para el producto final de software en cada etapa de desarrollo para cada característica de la calidad durante el uso, basada en el conocimiento de la calidad interna y externa.

La calidad en el uso es la visión del usuario de la calidad. El logro de la calidad durante el uso depende del logro de la calidad externa necesaria, que a su vez depende del logro de la necesaria calidad interna (Figura 3).

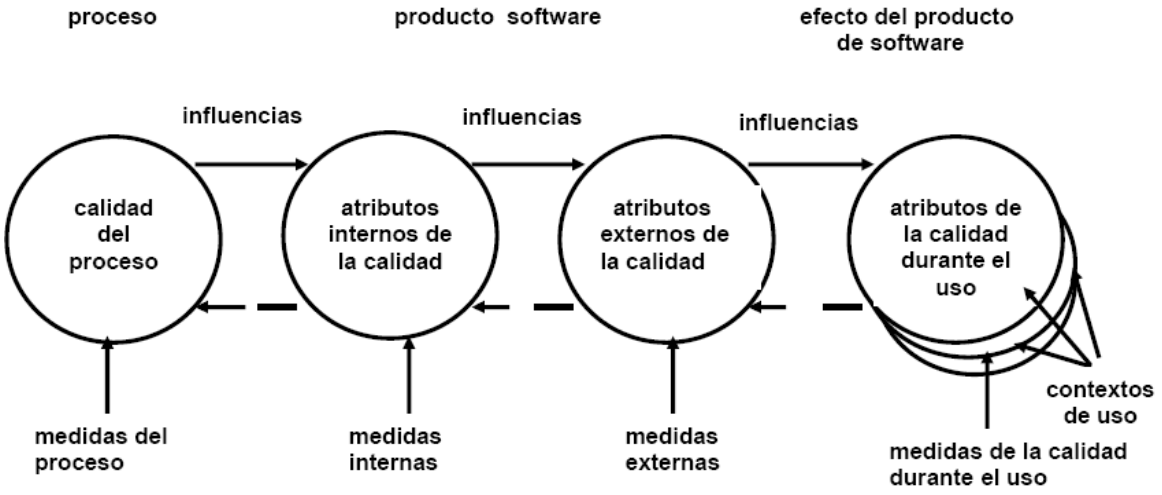


Figura 3: Calidad en el uso.

Los atributos de la calidad durante el uso se clasifican en cuatro características: eficacia, productividad, seguridad y satisfacción.

Eficacia: Capacidad del producto de software de permitir que los usuarios logren objetivos especificados con precisión e integridad en un contexto especificado.

Productividad: Capacidad del producto de software de permitir que los usuarios dediquen una cantidad de recursos apropiada en relación con la eficacia alcanzada en un contexto de uso especificado.
Entre los recursos se pueden incluir el tiempo para completar la tarea, los esfuerzos del usuario, los materiales o el costo de utilización en términos financieros.

Seguridad: Capacidad del producto de software de alcanzar niveles aceptables de riesgo de daños a las personas, el negocio, el software, la propiedad o el ambiente en un contexto de uso especificado.
Por lo general los riesgos son ocasionados por deficiencias en la funcionalidad (incluyendo la seguridad informática), confiabilidad, usabilidad o mantenibilidad.

Satisfacción: Capacidad del producto de software de satisfacer a los usuarios en un contexto de uso especificado.
La satisfacción es la respuesta del usuario a la interacción con el producto, e incluye la actitud hacia el uso del producto.

En el modelo para la calidad interna y externa se categorizan los atributos de calidad del software en seis características (la funcionalidad, la confiabilidad, la usabilidad, la eficiencia, la mantenibilidad y la portabilidad), que a su vez son divididas en sub-características (Figura 4).

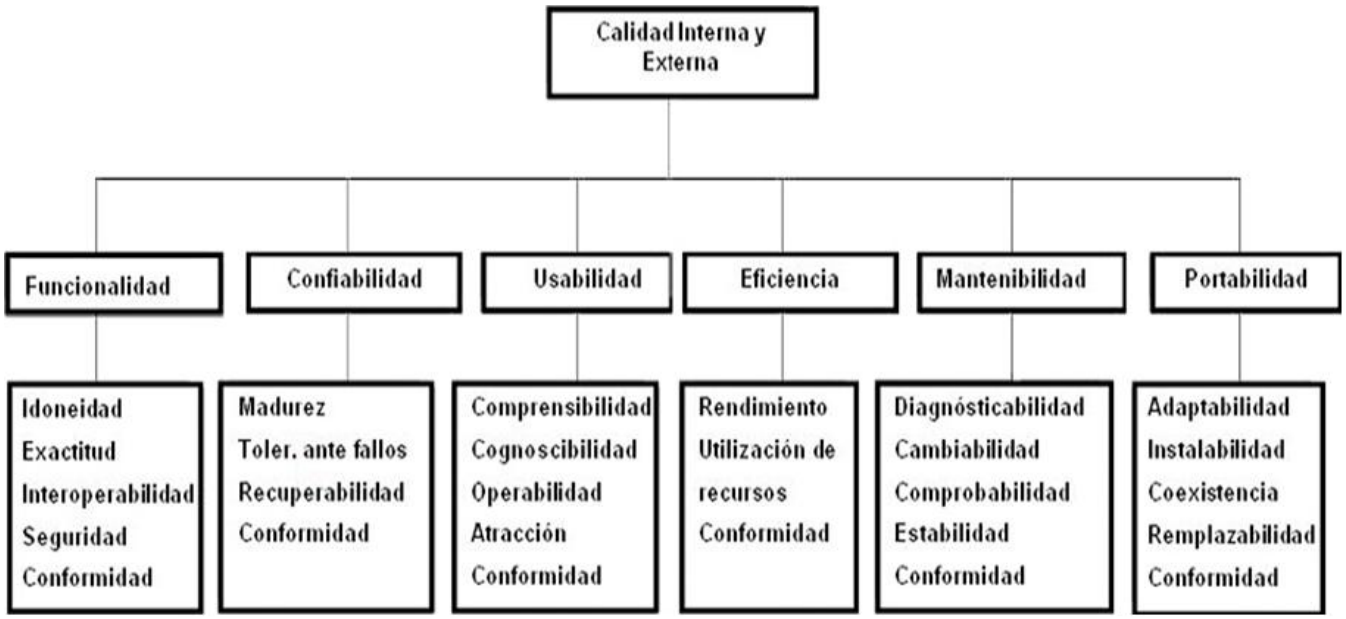


Figura 4: Modelo de calidad para la calidad interna y externa.

Cada unas de las características y sub-características se definen en el Anexo 7.

Debido a las prestaciones y características de la Arquitectura MDA, en esta investigación se consideran algunos requisitos significativos de calidad e importantes de mantener dentro del Plan de Aseguramiento de Calidad para cualquier proyecto de software desarrollado sobre esta arquitectura, a continuación se exponen los requisitos y el porqué de su denominación:

Mantenibilidad: MDA separa de forma radical las responsabilidades, en el PIM se representa el modelo de negocio y los PSMs las tecnológicas requeridas, cuando se quiere modificar un aspecto técnico solo basta con hacer cambios en los PSMs, sin tener impacto en la lógica del negocio, permitiendo hacerle modificaciones específicas a la aplicación (sub-característica: cambiabilidad). Debido a que MDA se basa en el desarrollo de software mediante modelos, para que, a partir de transformaciones a esos modelos, se genere código u otro modelo, es importante la estabilidad de sus modelos, para minimizar el número de efectos inesperados o cambios en el código generado (sub-característica: estabilidad).

Portabilidad: El objetivo de MDA es mejorar la portabilidad a través de la separación de arquitecturas de sistemas de arquitecturas de las plataformas. En MDA, los modelos PSMs representan la proyección de los PIMs en una plataforma específica, existiendo así para cada PIM

uno o varios PSMs cada uno para una tecnología distinta y colaborando entre sí para una solución completa, facilitando esto que el producto de software desarrollado sobre esta arquitectura tenga la capacidad de ser llevado de un entorno a otro (sub-característica: instalabilidad). Y adicionándole al software la posibilidad de ser adaptado a ambientes específicos sin aplicar otros medios que los suministrados con el propósito de cumplir con los fines esperados (sub-característica: adaptabilidad).

Los restantes atributos de calidad como son la funcionalidad, la confiabilidad, la usabilidad y la eficiencia, también son importantes para el logro de la calidad total, pero para un producto de software con arquitectura MDA dependerán de las características propias de cada proyecto en particular.

2.3 Gestión.

2.3.1 Organización.

Se describe la estructura de la organización en áreas del Aseguramiento de Calidad, la estructura general para proyectos de software se especifica en el Plan de Desarrollo de Software. Una propuesta de estructura básica para cualquier proyecto sería la Figura 5.

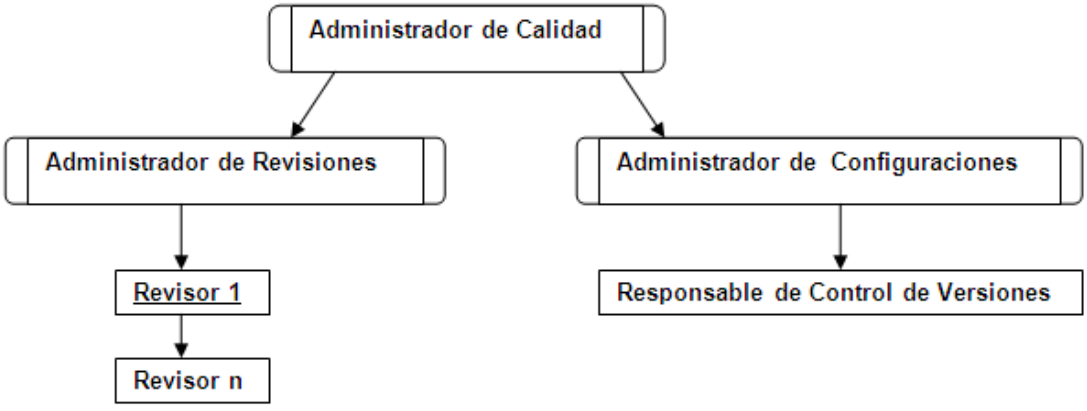


Figura 5: Estructura de organización.

2.3.2 Tareas y responsabilidades.

En la Tabla 2 se muestran cada una de las tareas de Aseguramiento de la Calidad y el responsable de su ejecución.

Tarea de Aseguramiento de Calidad	Responsable	Comentarios
Definición de estándares y procedimientos.	Administrador de Calidad.	Se definen los estándares y procedimientos a utilizar para realizar las actividades.
Revisiones internas.	Administrador de Revisiones.	Se realizan las revisiones internas a los artefactos y a los procesos definidos en el proyecto por parte de los revisores.
Definición del Plan de Configuración.	Administrador de Configuraciones.	Se estructura el Plan de Configuración especificando los elementos y mecanismos de configuración, y control de versiones, entre otras.
Administrar la Configuración.	Administrador de Configuraciones.	Se ejecutan las actividades definidas en el Plan de Configuración.
Toma e implantación de acciones correctivas.	Jefe de Proyecto.	Basadas en las no conformidades de las revisiones se toma un grupo de acciones correctivas para corregirlas.

Tabla 2: Tareas y responsabilidades.

2.4 Métricas.

Los requisitos para la calidad del producto de software incluyen el criterio para la evaluación de la calidad para satisfacer las necesidades de diseñadores, serviciadores o proveedores de soporte,

clientes y usuarios finales. En esta sección se presentan métricas de calidad para cada uno de los requisitos de calidad determinados en la investigación como significativos para la Arquitectura MDA, métricas para el Aseguramiento de la Calidad y métricas específicas para evaluar el diseño de un software con Arquitectura MDA.

Las métricas para la medición de la característica Mantenibilidad

Las métricas de mantenibilidad (Tabla 3) deben ser capaces de medir un atributo como es el comportamiento del serviciador, el usuario o el propio sistema, incluyendo el software, cuando es objeto de mantenimiento o modificación (9126-3).

Métrica	Medición (Fórmula)	Descripción
Conformidad de la mantenibilidad	[X = 1 - A / B] A = Número de elementos especificados que requiriendo estar en conformidad no han sido implementados durante las pruebas. B = Número total de elementos especificados que requieren estar en conformidad.	Esta métrica responde a cuán conforme es la mantenibilidad del producto de software con las regulaciones, las normas y convenciones que les son aplicables. Contar el número de elementos que se encontraron en conformidad y compararlo con el número de elementos que requieren estar en conformidad según la especificación. 0 <= X <= 1 A mayor cercanía al 1 mejor.

Tabla 3: Métricas de Mantenibilidad.

Las métricas para la medición de la característica Portabilidad

Las métricas de portabilidad (Tabla 4) deben ser capaces de medir un atributo como es el comportamiento del operador y el sistema durante las actividades de implementación y distribución del producto de software (9126-3).

Métrica	Medición (Fórmula)	Descripción
Facilidad de instalación	[X = A/B] A = Número de casos en los cuales el usuario tiene éxito en las operaciones para una instalación adecuada a su conveniencia. B = Número total de casos en los cuales el usuario intenta adecuar la instalación a su conveniencia.	Esta métrica responde si el usuario o serviciador puede instalar fácilmente el producto de software en su ambiente de operación. Observar el comportamiento del usuario o serviciador cuando tratan de instalar el producto de software en su ambiente de operación. 0 <= X <= 1 Más cercano al 1 resultará mejor.
Facilidad de reintento de ejecución del instalador	[X = A/B] A = Número de casos en los cuales el usuario no tiene éxito al reintentar ejecutar el instalador. B = Número total de casos en los cuales el usuario reintenta ejecutar la instalación durante la operación de instalación.	Esta métrica responde si el usuario o serviciador puede fácilmente reintentar ejecutar el programa instalador del software. 0 <= X <= 1 Más cercano al 1 resultará mejor.
Conformidad de la portabilidad	[X = 1 - A / B] A = Número de elementos especificados que requiriendo estar en conformidad no han sido implementados durante las pruebas. B = Número total de elementos especificados que requieren estar en conformidad.	Esta métrica responde a cuán conforme es la portabilidad del producto de software con las regulaciones, las normas y convenciones que les son aplicables. Contar el número de elementos que se encontraron en conformidad y compararlo con el número de elementos que requieren estar en conformidad según la especificación. 0 <= X <= 1 A mayor cercanía al 1 mejor

Tabla 4: Métricas de Portabilidad.

Las métricas para el Aseguramiento de la Calidad.

Las métricas para el Aseguramiento de la Calidad (Tabla 5) miden el desarrollo de los proyectos y ayudan a los líderes y directivos de los mismos en la toma decisiones y acciones correctivas, así como el mejoramiento continuo de los procesos, obteniéndose mejores resultados (UCI).

1. Esfuerzo dedicado a las actividades de SQA = horas/ hombre
2. Tiempo dedicado al SQA= Esfuerzo dedicado a las actividades de SQA/Esfuerzo total de proyecto.
3. Cumplimiento de las actividades de SQA =Número de revisiones, pruebas y auditorías realizadas/ Número de revisiones, pruebas y auditorías planeadas.
4. Eficiencia y efectividad
    - En la respuesta a las NC= NCSR / (NCR + NCSR).

    NCSR: Número de no conformidades sin resolver elevadas al líder de proyecto.

    NCR: Número de no conformidades reportadas.
    - En las pruebas internas= Cantidad. NC PI/ (Cant. NC PL+ Cant. NC PI).
        - PI: pruebas de internas.
        - PL: pruebas de liberación.

Métrica	Medición (Fórmula)	Descripción
Tamaño Estimado		Tamaño estimado Se expresa en LOC(líneas de código)
Tamaño		Tamaño real Se expresa en LOC(líneas de código)
Error Estimando Tamaño	([Tamaño] - [Tamaño Estimado]) / [Tamaño Estimado]	Permite apreciar el margen de error en la estimación de tamaño. Esta métrica constituye un indicador de nuestra calidad en la estimación de tamaño. Entre más cercano a 0 sea el valor mejor será nuestra estimación de tamaño. Valores sustancialmente mayores que 0 indican que el tamaño real esta siendo sustancialmente mayor que el tamaño estimado y nuestra

		estimación está siendo demasiado irreal. Valores sustancialmente menores que 0 indican que nuestra estimación está siendo muy conservadora.
Defectos Removidos Estimados		Estimación de los defectos a eliminar.
Defectos Removidos		Defectos eliminados reales.
Defectos Inyectados Estimados		Estimación de defectos a inyectar.
Densidad Defectos Estimada	[Defectos Removidos Estimados] / [Tamaño Estimado]	Revela la cantidad estimada de defectos que se eliminan por líneas de código. La tendencia en esta métrica es ir disminuyendo paulatinamente.
Densidad Defectos	[Defectos Removidos] / [Tamaño]	Revela la cantidad real de defectos que se eliminan por líneas de código. La tendencia en esta métrica es ir disminuyendo paulatinamente
Estimación de defectos inyectados por horas	60*[Defectos Inyectados Estimados] / [Tamaño Estimado]	Representa la cantidad estimada de defectos a inyectar por hora en el desarrollo de un proyecto. Esta métrica también puede ser evaluada a nivel de fases. Donde se tendrían en cuenta la estimación de defectos a inyectar para esa fase y su tiempo estimado. El objetivo o valor ideal para esta métrica es 0, La tendencia debe ser a ir disminuyendo gradualmente.
Defectos inyectados por horas	60*[Defectos Inyectados] / [Tamaño]	Representa la cantidad de defectos inyectados por hora en el desarrollo de un proyecto. Esta métrica también puede ser evaluada a nivel de fases. Donde se tendrían en cuenta los defectos a inyectar para esa fase y su tiempo. El objetivo o valor ideal para esta métrica es 0, La tendencia debe ser a ir disminuyendo gradualmente.

% de Apreciación Estimado del costo de la calidad	∑(Tiempo Estimado en las fases de apreciación)/ [Tiempo Estimado]	Calcula qué % del Tiempo Estimado total representa el Tiempo Estimado para las fases de apreciación. El tiempo estimado que se invierte en revisar o inspeccionar, es un costo en el que se incurre en favor de la calidad. La meta a lograr es que este porcentaje vaya aumentando en la medida que se aumente en madurez. Al aumentar la Apreciación Estimada del costo de la calidad, debe ir disminuyendo la cantidad de defectos detectados en prueba, lo cual evidencia la relación entre el costo de la calidad de valoración y la mejora de la calidad de producto. Esta métrica sólo se refleja en los resúmenes si existe alguna fase de apreciación.
% de Apreciación del costo de la calidad	∑(Tiempo en las fases de apreciación)/ [Tiempo]	Calcula qué % del Tiempo total representa el Tiempo para las fases de apreciación. El tiempo estimado que se invierte en revisar o inspeccionar, es un costo en el que se incurre en favor de la calidad. La meta a lograr es que este porcentaje vaya aumentando en la medida que se aumente en madurez. Al aumentar la Apreciación del costo de la calidad, debe ir disminuyendo la cantidad de defectos detectados en prueba, lo cual evidencia la relación entre el costo de la calidad de valoración y la mejora de la calidad de producto. Esta métrica sólo se refleja en los resúmenes si existe alguna fase de apreciación.
% de Fallo Estimado del costo de la calidad	∑(Tiempo Estimado en las fases fallo)/ [Tiempo Estimado]	Calcula qué % del Tiempo Estimado total representa el Tiempo Estimado en las fases de fallo. La meta a alcanzar es esta métrica vaya disminuyendo en la medida que se aumente en madurez, debido a que se debe invertir más tiempo en fases de apreciaciones que en fases de fallo.

		Esta métrica sólo se refleja en los resúmenes si existe alguna fase de fallo.
% de Fallo del costo de la calidad	∑(Tiempo en las fases fallo)/ [Tiempo]	Calcula qué % del Tiempo total representa el Tiempo en las fases de fallo. La meta a alcanzar es esta métrica vaya disminuyendo en la medida que se aumente en madurez, debido a que se debe invertir más tiempo en fases de apreciaciones que en fases de fallo. Esta métrica sólo se refleja en los resúmenes si existe alguna fase de fallo.
Estimación de la razón apreciación-fallo.	[% de Apreciación Estimada del costo de la calidad]/ [% de Fallo Estimado del costo de la calidad]	Calcula la razón que existe entre el % Apreciación Estimado del costo de la calidad y el % Fallo Estimado del costo de la calidad. La meta a alcanzar es que la Estimación de la razón apreciación- fallo vaya aumentando en la medida que se aumente en madurez, debido a que debe ir aumentando el % Apreciación Estimada del costo de la calidad y disminuyendo el % Fallo Estimado del costo de la calidad.
Razón apreciación-fallo.	[% de Apreciación del costo de la calidad] / [% de Fallo del costo de la calidad]	Calcula la razón que existe entre el % Apreciación del costo de la calidad y el % Fallo del costo de la calidad. La meta a alcanzar es que la razón apreciación-fallo vaya aumentando en la medida que se aumente en madurez, debido a que debe ir aumentando el % de Apreciación del costo de la calidad y disminuyendo el % de Fallo del costo de la calidad.
% Estimado del costo de la calidad	[% de Apreciación Estimada del costo de la calidad]	Esta métrica no aparece directamente en los resúmenes ya que su significado es el mismo que Apreciación Estimada del costo de la calidad.
% del costo de la calidad	[% de Apreciación del costo de la calidad]	Esta métrica no aparece directamente en los resúmenes ya que su significado es el mismo que Apreciación del costo de la calidad.

Tabla 5: Métricas de Aseguramiento de la Calidad.

Las Métricas que a continuación se exponen ayudarán a la evaluación del diseño de software con Arquitectura MDA.
(PRESSMAN 1993)

Métricas del Modelo de Diseño

Proporcionan al diseñador una mejor visión interna y ayudan a que el diseño evolucione a un nivel superior de calidad.
Se dividen en:

- Métricas de diseño arquitectónico.

Pretende medir la complejidad del diseño de alto nivel. Se concentra en la complejidad de la arquitectura, con de caja negra, aunque rompan el diseño, no entran en ningún caso a evaluar el código.

- Métricas de diseño a nivel de componente.

Tienen como objetivo llevar un control de la calidad del producto que se está desarrollando y estimar el impacto que ese producto tendrá en las fases posteriores del proceso de desarrollo.

Métricas de Diseño Arquitectónico

Henry y Kafura proponen en 1981 la métrica de complejidad de expansión – concentración, que considera las estructuras de datos que recoge (concentran) o actualizan (expansión).
MHK = longitud (i) x [fin (i) + fout (i)] 2
Donde longitud (i) = número de sentencias en lenguaje de programación

En 1991, Fenton propone medidas de morfología simples basadas en la estructura de módulos jerárquicos del sistema:
Tamaño = n + a (número de nodos + número de aristas)
Profundidad
Anchura
Relación arco-nodo, r = a/n

Métrica de Cohesión

Para medir la cohesión se puede usar el trabajo de Bieman y Ott, el cual define varios conceptos:

- número de elementos (tokens).
- rebanada de datos (data slice).
- adhesivo (glue).
- superadhesivo (superglue).

Con ellas se calcula la cohesión funcional fuerte
CFF = número de superadhesivos / número de elementos

Cohesión Funcional Débil, se calcula:
CFD = número de adhesivos / número de elementos

Mientras más cercano sean CFF ó CFD a 1, mayor será la cohesión del módulo.

Métrica de Acoplamiento

Proporciona una indicación de la conectividad de un módulo con otros.
Medidas para el acoplamiento de flujo de datos de control.

- di = número de parámetros de datos de entrada
- ci = número de parámetros de control de entrada
- do = número de parámetros de datos de salida
- co = número de parámetros de control de salida

Indicador de acoplamiento de módulo.
mc = k / M, donde k = 1, constante de proporcionalidad.
M = di + a x ci + do + b x co+ gd + c x gc + w + r, con a = b = c = 2
Cuando mayor es mc, menor es el acoplamiento del módulo.

2.5 Estándares y Guías.

En la Tabla 6 se listan de los estándares y guías a utilizar para el desarrollo de un proyecto con Arquitectura MDA y la ejecución de su Plan de Aseguramiento de la Calidad. Estos estándares y guías se pondrán a disposición de todos los trabajadores del proyecto de software con Arquitectura MDA en el Expediente de Proyecto para ser estudiados y utilizados dependiendo de las necesidades y rol de cada trabajador.

Estándar	Ubicación	Comentarios
ISO std 9126-1	Expediente de Proyecto	Se definen los objetivos de calidad a evaluar en el proyecto.
ISO std 9126-2	Expediente de Proyecto	Define métricas para evaluar el comportamiento del sistema mediante el ensayo.
ISO std 9126-3	Expediente de Proyecto	Define métricas para aplicarse a productos de software no ejecutables (como especificación o código fuente) durante el diseño y la programación. Lo que permite evaluar la calidad desde las etapas más tempranas.
ISO std 9126-4	Expediente de Proyecto	Define métricas para medir hasta que punto el producto satisface las necesidades de los usuarios para lograr las metas especificadas con la efectividad, productividad, seguridad y satisfacción en un contexto determinado de uso.
ISO std 10011-1	Expediente de Proyecto	Establece los principios básicos, criterios y prácticas de una auditoría y provee lineamientos para establecer, planificar, realizar y documentar auditorías de sistemas de la calidad.
ISO std 10011-2	Expediente de Proyecto	Criterios para la calificación de auditores. A fin de que las auditorías de los sistemas de calidad sean conducidas en forma uniforme y efectiva se ha desarrollado esta norma que constituye una guía sobre los criterios de calificación de auditores.

IEEE std 828	Expediente de Proyecto	Elaboración de planes de manejo de configuración donde se describe todo el proceso de Gestión de Configuración del software.
IEEE std 829	Expediente de Proyecto	Describe un conjunto de documentos de prueba que están asociadas con los aspectos dinámicos como la ejecución de los procedimientos y el código del software.
IEEE std 1028	Expediente de Proyecto	Define procedimientos para definir y llevar a cabo procesos de revisión y auditoría del software.
IEEE std 730	Expediente de Proyecto	Proporciona la estructura de la documentación del plan de aseguramiento de la calidad.
UML	Expediente de Proyecto Descargar de www.uml.org	El Lenguaje Unificado de Modelado (UML) es un estándar de modelado de lenguaje para visualizar, especificar y documentar sistemas de software. Los modelos utilizados con MDA se pueden expresar mediante el lenguaje UML.
MFO	Expediente de Proyecto Descargar de www.mfo.org	Meta-Object Facility (MOF) es una tecnología que proporciona un modelo de repositorio que puede ser utilizado para especificar y manipular modelos, favoreciendo así la coherencia en la manipulación de modelos en todas las fases de la utilización de MDA.

Tabla 6: Estándares y Guías.

2.6 Revisiones y Auditorías.

Una parte importante en el Plan de Aseguramiento de la Calidad lo constituyen las Revisiones y Auditorías, que se realizan en los diferentes momentos del ciclo de desarrollo del proyecto. Con ellas se detectan defectos que pueden ser corregidos oportunamente, de forma tal que el producto final tenga la calidad necesaria.

Las Revisiones son actividades de control de calidad, que permiten detectar defectos en los proyectos de software. Las Revisiones pueden ser de dos tipos, tal y como se muestra en la Figura 6, dinámicas y estáticas. Las dinámicas son las que detectan los defectos ejecutando el software,

fundamentalmente son las ejecutadas en la fase de prueba del proyecto. Las estáticas son visuales y se realizan sin necesidad de que el software esté ejecutándose. Ambas son de suma importancia y una combinación adecuada puede detectar gran cantidad de defectos y por tanto mejorar la calidad del producto final (MARTHA DUNIA DELGADO DAPENA 2005).

Figura 6: Tipos de Revisiones.

El autor Mario Delgado (DELGADO 2003) recomienda realizar al menos las siguientes revisiones y auditorías durante el desarrollo del proyecto:

• Inspecciones del Colectivo: Ejecutadas por el propio equipo de desarrollo, en este caso se recomiendan las revisiones de los requisitos, la arquitectura, el Diseño y los ciclos de desarrollo. Por otro lado, las revisiones de la arquitectura y el diseño son las que revisan el diseño preliminar y crítico respectivamente, que persiguen el objetivo de detectar defectos antes de pasar a la fase de codificación, y por último las revisiones al finalizar cada ciclo de desarrollo detectan defectos presentes en los componentes y artefactos del proyecto desarrollados en cada uno de los ciclos.

• Auditoría de la Configuración: Ejecutadas por personal ajeno al desarrollo del proyecto, se deben ejecutar antes de incluir la línea base del proyecto en el repositorio de elementos de configuración, para garantizar la estabilidad de los elementos de configuración. Además debe realizarse al terminar cada versión del software. El objetivo fundamental de estas revisiones es detectar defectos que los propios desarrolladores no sean capaces de detectar porque estén relacionados con la lógica de pensamiento seguida por ellos en el desarrollo del proyecto, en ella se verifica que todos los requisitos han sido cumplidos y que el software y su documentación están completos y listos para entregar.

• Revisiones Dinámicas: Se deben ejecutar las actividades de evaluación y prueba del software, considerando en cada momento el tipo de prueba que se requiere.

Las Revisiones y Auditorías a cada uno de los proyectos desarrollados en la Universidad de las Ciencias Informáticas se realizan a nivel central por un grupo encargado de la Dirección de Calidad, pero no deja de ser recomendable para un producto de software con Arquitectura MDA la ejecución de Inspecciones del Colectivo, y dentro de ellas las revisiones de la arquitectura y el diseño para encontrar los defectos antes la codificación generada por la arquitectura.
Las Auditorías realizadas a nivel central en la Universidad son basada mediante el procedimiento:

Nombre:

Auditorías a la actividad productiva.

Objetivo:

Establecer el procedimiento para realizar auditorías a la actividad productiva en la UCI y proporcionar la forma específica de cómo llevar a cabo las actividades de auditoría, orientar sobre las normas en base a las cuales se harán.

La auditoría examina y evalúa el grado de correspondencia entre los procedimientos, lineamientos y disposiciones establecidos y su aplicación a fin de determinar el grado de planificación, organización, dirección, control y si se han alcanzado las metas propuestas.

Alcance:

Este procedimiento es aplicable a las auditorías a realizar, a la actividad productiva de la UCI coordinadas desde la Dirección de Calidad.

Responsables:

Ejecuta: Especialistas de la Dirección de Calidad y personal designado para ejecutar la auditoría.

Responsable de su ejecución: Director de Calidad y Director General.

Revisa y actualiza este procedimiento: Dirección de Calidad.

Fiscaliza su cumplimiento: Dirección de Calidad, Grupo de Control Interno, Director de Calidad y Director General.

El desarrollo del procedimiento se explica en el Anexo 8.

2.6.1 Tareas generales de Revisiones y Auditorías.

En la Tabla 7 se describen las actividades a desarrollar mediante el procedimiento Auditorías a la actividad productiva que se realiza a nivel central de en la Universidad.

Día	Actividad	Contenido
1	Reunión de inicio.	Se da el curso a los revisores y se planifica la revisión
2	Revisión de la documentación.	Trabajo individual en función de la asignación de tareas.
3	Revisión de la documentación.	Trabajo individual en función de la asignación de tareas.
4	Revisión de la documentación.	Trabajo individual en función de la asignación de tareas.
5	Reunión de preparación de la reunión de apertura.	Desarrollar el borrador del acta de Reunión de Apertura.
	Reunión de apertura.	Informar los objetivos, alcance y criterios de evaluación, acordar actividades y aseguramientos.
6	Reunión de preparación de la reunión de apertura.	Entrevistas y revisiones de evidencias con el equipo de proyecto.
7	Reunión de preparación de la reunión de apertura.	Entrevistas y revisiones de evidencias con el equipo de proyecto.
8	Reunión de preparación de la reunión de apertura.	Entrevistas y revisiones de evidencias con el equipo de proyecto.
9	Reunión de preparación de las conclusiones.	Preparación de las recomendaciones y de reunión de cierre de las revisiones.
10	Reunión de Pre cierre.	Acordar las recomendaciones.
	Preparar documentación de cierre.	Preparar documentación del expediente y comenzar Informe de Cierre.
11	Reunión de cierre.	Presentar conclusiones y evaluación.

Tabla 7: Actividades del Procedimiento “Auditorías a la actividad productiva”.

Criterios de Evaluación:

- Gestión de proyecto
  - Definición del proyecto.
  - Establecimiento del alcance del proyecto.
  - Gestión de los riesgos del proyecto.
  - Establecimiento de la estructura organizacional, los roles y las responsabilidades.
  - Asignación de los roles y las tareas.
  - Vinculación con los procesos docentes e investigativos.
  - Empleo de la metodología seleccionada y de la notación UML.
  - Registro del capital humano, los recursos materiales y
  - los resultados del proyecto.
- Planificación
  - Registro de las actividades conjuntas y visitas de los diferentes factores al proyecto.
  - Distribución de las PC y asignación de los horarios de producción.
  - Establecimiento y cumplimiento del cronograma del proyecto.
  - Registro del tiempo de trabajo de los desarrolladores y de los defectos detectados.
  - Registro de las estimaciones realizadas (esfuerzo y tiempo de desarrollo).
- Soporte
  - Establecimiento de la gestión de configuración.
  - Establecimiento del plan de aseguramiento de la calidad.
  - Definición y empleo de las herramientas a utilizar en el desarrollo del proyecto.
- Ingeniería
  - Establecimiento del glosario de términos del proyecto.
  - Gestión de los requisitos.
  - Definición de la arquitectura.
  - La conformidad del producto con los requerimientos y del proyecto.
- Expediente de proyecto
  - Conformidad con el esquema del expediente de proyecto.
  - Completamiento del expediente de proyecto.

2.7 Pruebas y Evaluación.

El objetivo de la prueba de software es descubrir errores. Para conseguir este objetivo se planifica y se ejecuta una serie de pasos que van revisando todos los elementos del software. La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas:

• Prueba de Desarrollador.

Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para la prueba de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración. Se recomienda que estas pruebas cubran más que las pruebas de unidad.

• Prueba Independiente.

Es la prueba que es diseñada e implementada por alguien independiente del grupo de desarrolladores. El objetivo de estas pruebas es proporcionar una perspectiva deferente y en un ambiente más rico que los desarrolladores. Una vista de la prueba independiente es la prueba independiente de los stakeholder, que son pruebas basadas en las necesidades y preocupaciones de los stakeholders.

• Prueba de Unidad.

Es la prueba enfocada a los elementos más pequeño del software que pueden ser probados. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca.

• Prueba de Integración.

Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Estas pruebas descubren errores o incompletitud en las especificaciones de las interfaces de los paquetes. Esta prueba debe ser responsabilidad de desarrolladores y de independientes, sin solaparse las pruebas.

• Prueba de Sistema.

Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes

de software y hardware han sido integrados. En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados.

• Prueba de Aceptación.

Prueba de aceptación del usuario es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

En todas las fases del desarrollo del proyecto hay que probar el software que se va construyendo, aunque como el grueso de la programación se realiza en la construcción, es en esa fase en la que se centran los mayores esfuerzos de este flujo.

Como en la Arquitectura MDA se genera el código fuente, el mayor peso de las pruebas caerá sobre la fase de transición, donde se efectuarán pruebas del sistema.

La etapa de prueba es tan o más importante que todas las realizadas hasta el momento puesto que en ella se refleja la calidad con que ha sido llevada a cabo la proyección del sistema.

A continuación se definen un conjunto de modelos que dan soporte al proceso de generación de pruebas del sistema desde una perspectiva MDA.

Proceso de Generación de Pruebas del Sistema.

(JAVIER J. GUTIÉRREZ 2006)

Toda prueba consta tradicionalmente de tres elementos:

1. Interacciones entre el sistema y la prueba.
2. Valores de prueba.
3. Resultados esperados.

Los dos primeros elementos permiten realizar la prueba y el tercer elemento permite evaluar si la prueba se superó con éxito o no.

Un proceso de pruebas consta generalmente de cuatro fases: la fase de diseño de pruebas, la fase de codificación, la fase de ejecución y la fase de análisis de los resultados.

El objetivo de un proceso de generación de pruebas del sistema es desarrollar las dos primeras fases y obtener esos tres elementos a partir del modelo de requisitos del propio sistema bajo prueba. Dicho proceso toma como punto de partida los requisitos y, a partir de ellos genera los resultados y construye las pruebas. La Figura 7 ilustra un proceso genérico que recoge las ideas principales extraídas después de realizar un estudio comparativo sobre 12 propuestas. A partir de este estudio comparativo y de varios casos prácticos, se han identificado un conjunto de actividades pertenecientes al proceso de generación de pruebas que son independientes de la plataforma de la implementación, es decir, dichas actividades no se ven afectadas si, por ejemplo, el sistema a probar es un sistema web o un sistema de escritorio monousuario. De esta manera, es posible generar un conjunto de pruebas independientes de la plataforma. Sólo es necesario conocer los detalles de la plataforma a la hora de implementar las pruebas generadas.

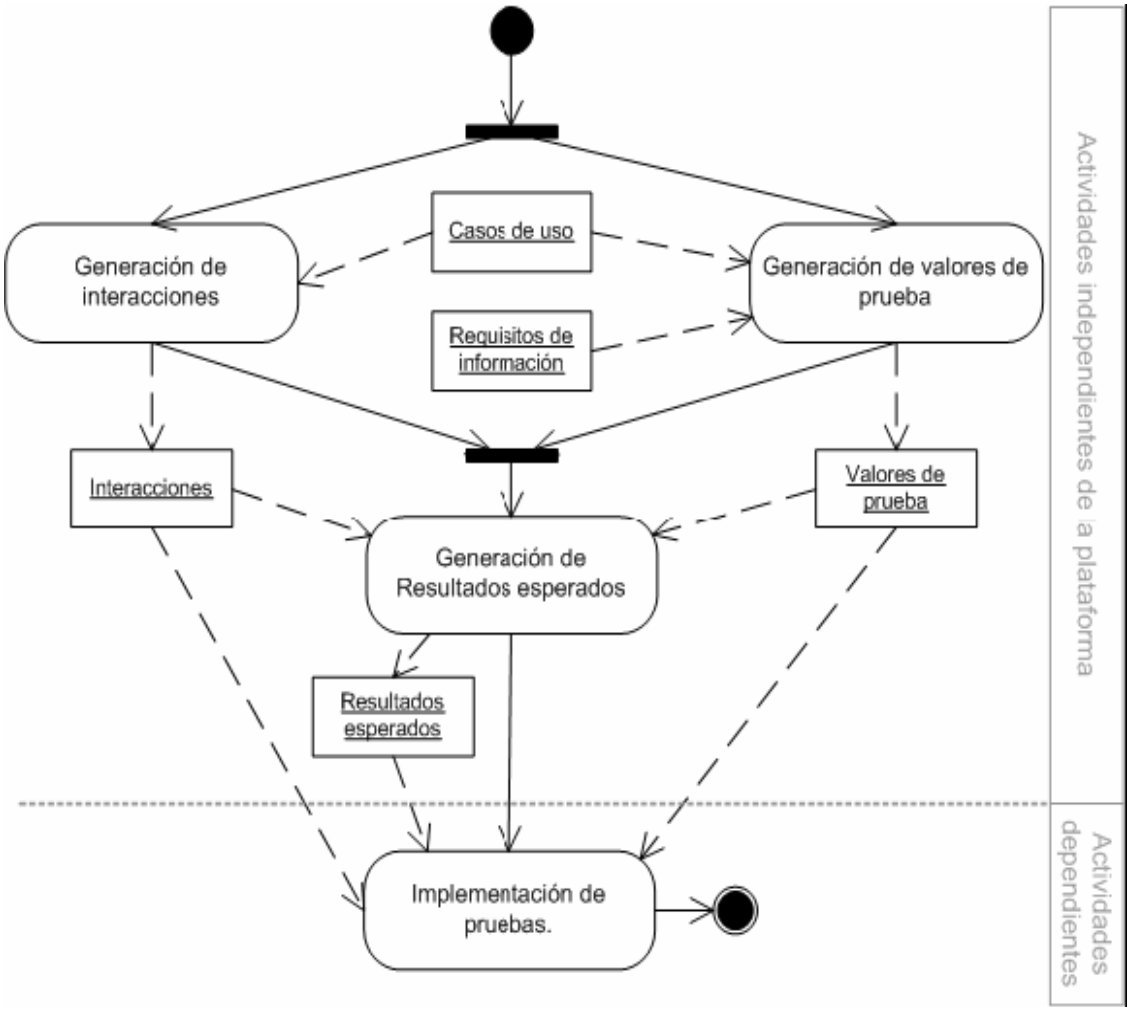


Figura 7: Actividades pertenecientes al proceso de generación de pruebas.

Modelo General para la Generación de Pruebas.

En este punto se describen un conjunto de modelos de prueba independientes y dependientes de la plataforma (PITs y PDTs). En la Figura 8 se muestran los modelos implicados. Las líneas entre modelos implican las dependencias y las futuras transformaciones.

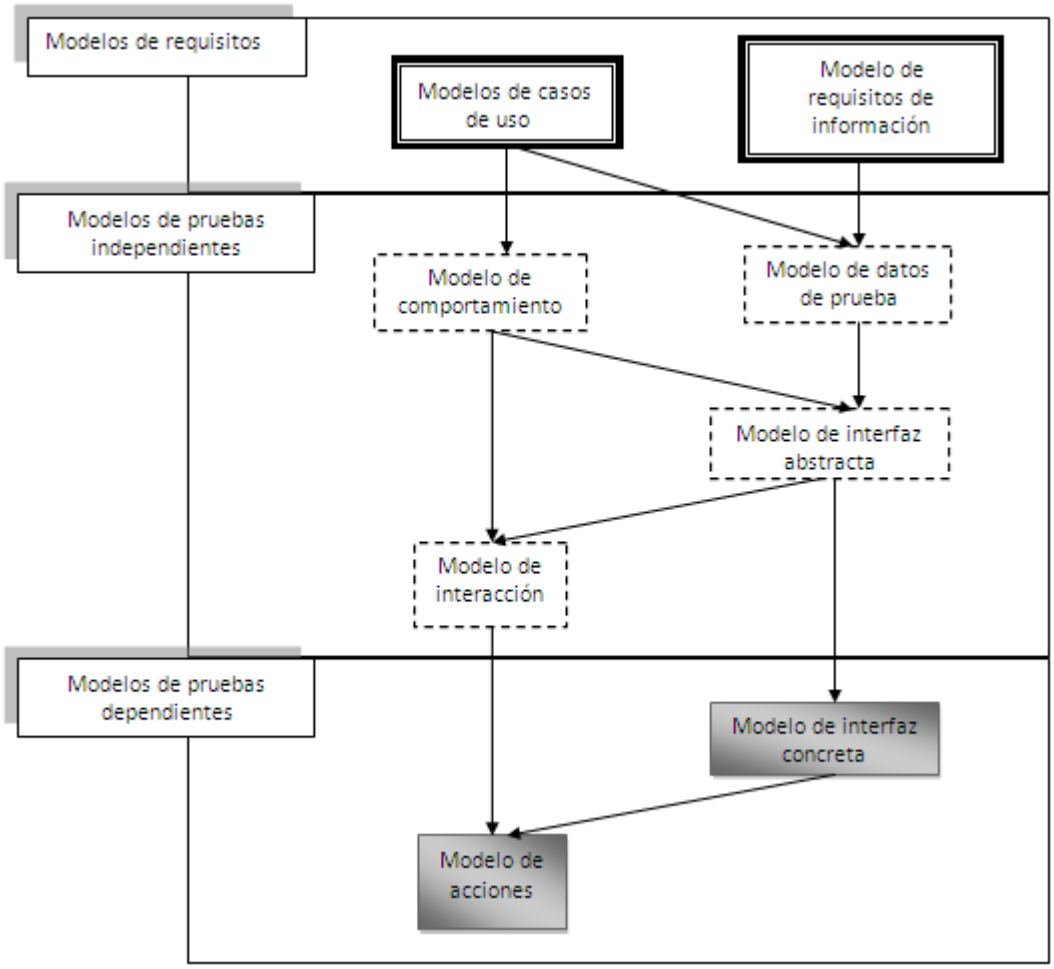


Figura 8: Modelos para la generación de pruebas.

1. Modelos de requisitos.

Los únicos modelos de requisitos necesarios son los casos de uso y los requisitos de almacenamiento, aunque otros modelos, como por ejemplo modelos de interfaces o modelos de navegación pueden enriquecer el proceso de prueba.

2. Modelo de comportamiento.

Un gran número de técnicas de requisitos están basadas en casos de uso definidos en prosa. Pero no es sencillo manipular casos de uso escritos en prosa. Por este motivo, el primer paso del proceso sistemático de generación de pruebas consiste en expresar dicha prosa mediante un modelo formal manipulable de manera automática.

El objetivo del modelo de comportamiento es expresar la misma información contenida en una plantilla de caso de uso de una forma fácilmente manipulable. Las propuestas estudiadas utilizan como modelos de comportamiento diagramas UML de estados, diagramas UML de secuencia o diagramas UML de actividades. En este caso se seleccionó el diagrama de actividades ya que, a diferencia de los diagramas de secuencia, permiten expresar caminos alternativos fácilmente y, a diferencia de los diagramas de estados, permiten expresar la interacción entre el sistema y los actores externos identificando claramente a cada uno de los participantes. En la Tabla 8 se muestra la definición en XML de un caso de uso como ejemplo. En la Figura 9 se muestra el diagrama de actividades correspondiente.

```
<useCase id="01-AddLink">
<description> This use case allow to introduce a new link. </description>
<mainSequence>
<step id="1"> The user selects the option for introduce a new link.
</step>
<step id="2"> The System selects top category and shows the form to
introduce the information of a link (SR-02). </step>
<step id="3"> The User introduces information of the new link and press
insert button. </step>
<step id="4"> The System stores the new link and shows the main screen of the system.
</step>
</mainSequence>
<alternative>
<alstep id="3.1.i"> If the user press cancel button then the use case ends.
</alstep>
<alstep id="3.2.i"> If the user selects a different category (SR-01) then
system changes the category and the result is to show the form again and execute
step 2.
</alstep>
<alstep id="4.1.p"> If the link name or link url is empty, then the system
shows an error message with the result of execute step 2.
</alstep>
</alternative>
</useCase>
```

Tabla 8: Caso de uso en formato XML.

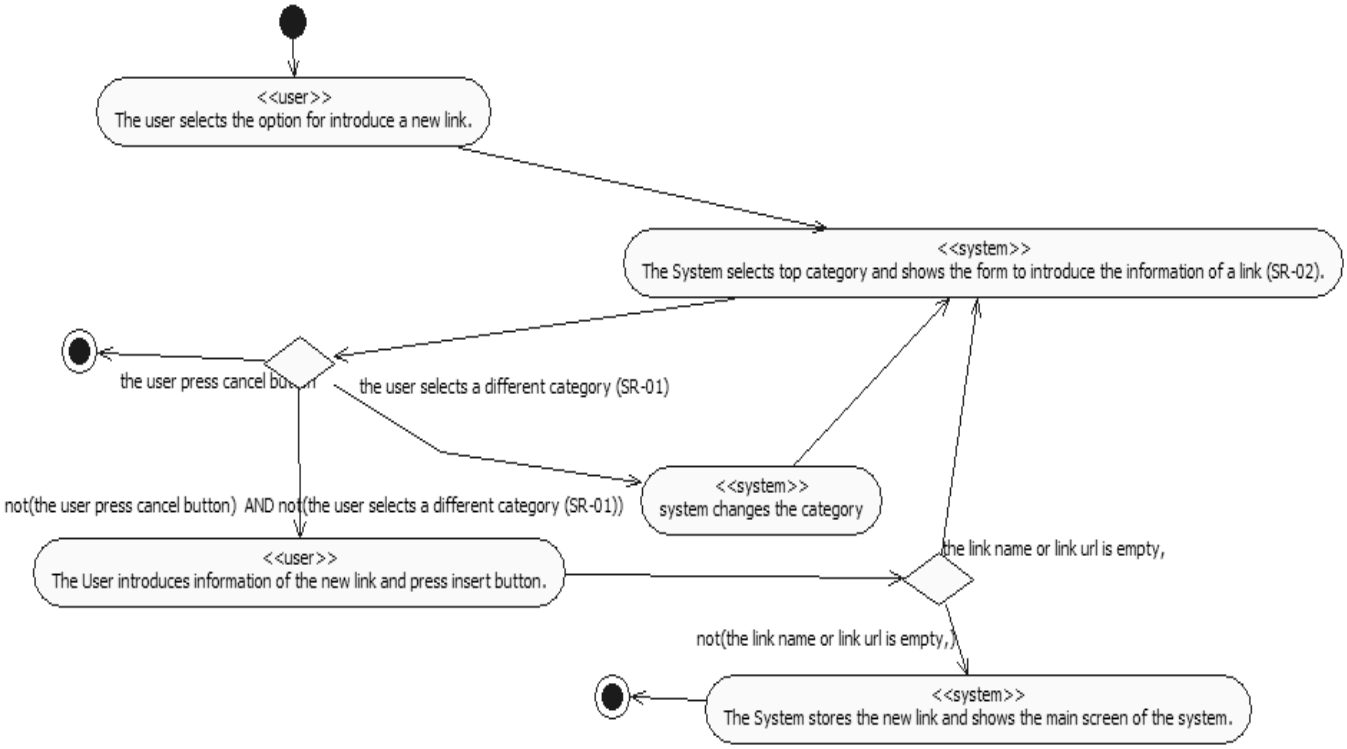


Figura 9: Modelo de comportamiento.

3. Modelo de datos de prueba.

Los casos de uso contienen elementos variables cuyos valores o comportamiento difiere de una ejecución de un caso de uso a otra. Algunos ejemplos son la información suministrada por un actor, una opción seleccionada por un actor, o la información mostrada por el sistema como resultado del caso de uso.

Los objetivos del modelo de datos de prueba son dos. En primer lugar, el modelo de datos de prueba (Figura 10) expresa todas las variables del caso de uso, su estructura si son tipos complejos (como clientes o compras), las restricciones que puedan existir entre ellos y las particiones de sus respectivos dominios. Esto se realiza mediante un diagrama de clases según la notación propuesta en el Testing Profile de UML. Dicho diagrama de clases puede extraerse automáticamente. Las clases se obtienen a partir de los requisitos funcionales y las distintas particiones se obtienen a partir de las condiciones evaluadas en las alternativas del diagrama de comportamiento. Este diagrama de clases puede refinarse posteriormente añadiendo particiones adicionales si fuera necesario.

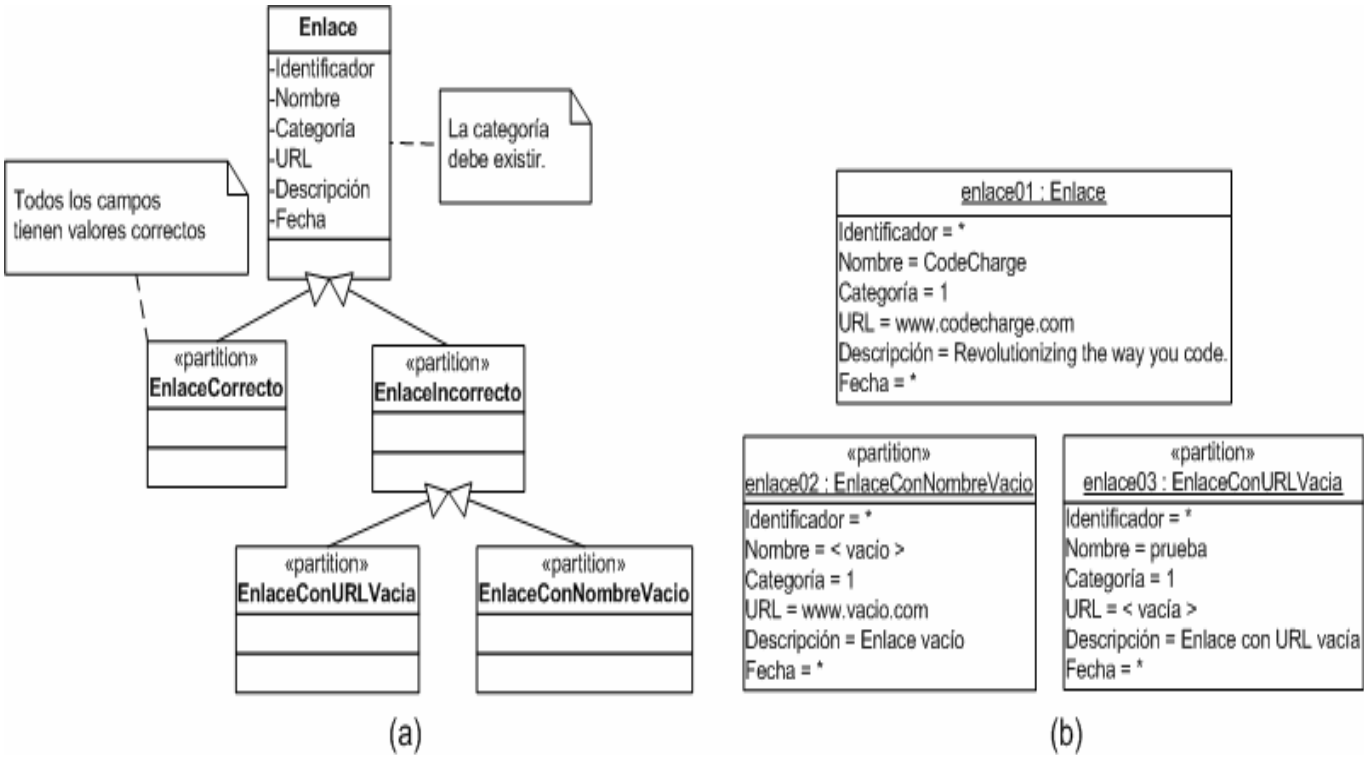


Figura 10: Ejemplo de modelo de datos de prueba.

4. Modelo de interfaz abstracta.

Los modelos anteriores nos indican lo que una prueba debe hacer (ejecutar un escenario posible de un caso de uso), qué información hay que suministrarle y qué información nos va a devolver. Sin embargo estos modelos aún son demasiado abstractos y no se pueden convertir en modelos dependientes de la plataforma ni en pruebas ejecutables de manera directa. Por este motivo, a partir de los modelos anteriores, se obtienen los modelos de interfaz abstracta y de interacción.

Lo primero que es necesario saber para poder implementar las pruebas es cómo interactuar con el sistema, es decir, cuál va a ser la interfaz que el sistema presenta a la prueba para que esta pueda interactuar con él. El objetivo del modelo de interfaz abstracta es definir las interfaces que el sistema ofrecerá para poder realizar la funcionalidad expresada en el modelo de casos de uso y en el modelo de comportamiento. No es necesario, sin embargo, entrar en detalles de la implementación de dichas interfaces.

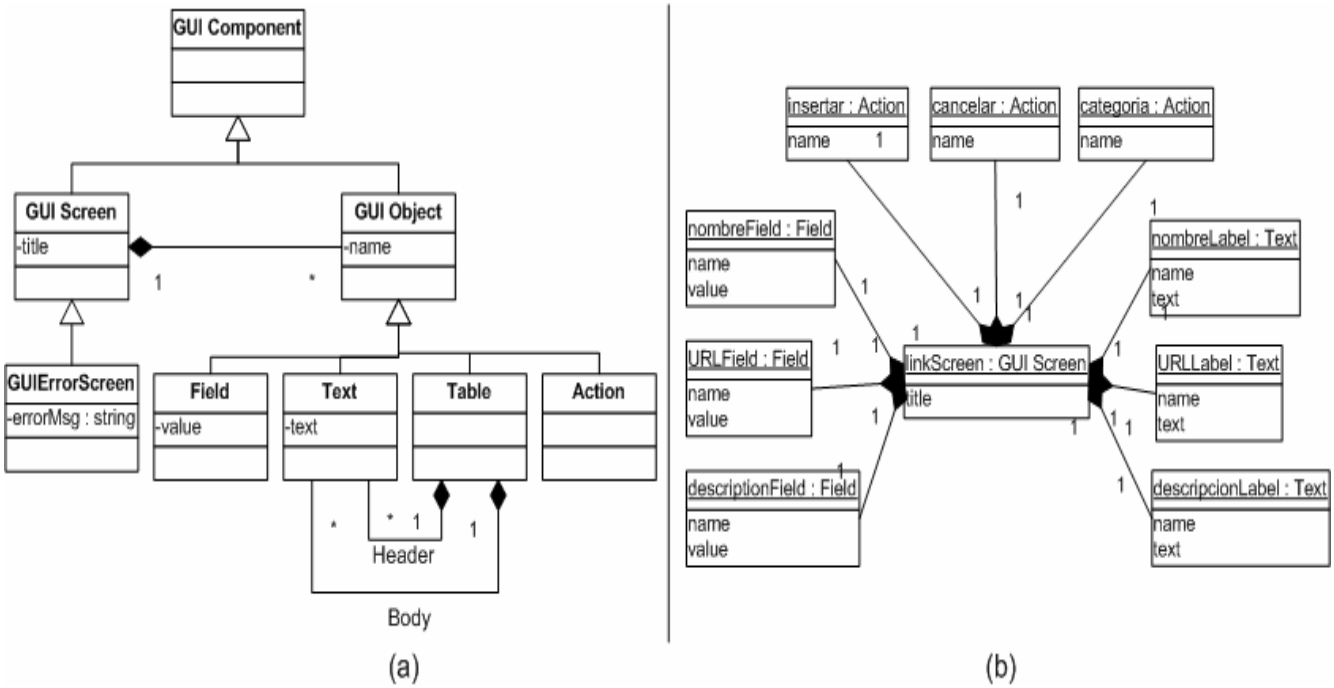


Figura 11: Modelo de componentes para la construcción de interfaces abstractas.

Para lograr la independencia de la plataforma este modelo se construye a partir de un metamodelo de componentes que abstraen las características de los componentes específicos. Este metamodelo puede adaptarse y ampliarse fácilmente. En la Figura 11a se muestra un ejemplo de metamodelo para interfaces gráficas de usuario y, en la Figura 11b, un modelo de interfaz abstracta basada en dicho metamodelo.

5. Modelo de interacción.

Una vez que se conocen las interfaces con las que las pruebas interactuarán, expresadas mediante el modelo de interfaz abstracta, se refina el modelo de comportamiento para indicar cómo realizar cada uno de los pasos del caso de uso sobre dicha interfaz. El objetivo del modelo de interacción es definir cómo realiza las pruebas sus acciones y definir los árbitros. En el contexto de las pruebas, un árbitro es elemento encargado de comprobar si la prueba fue superada o no.

Para lograr la independencia tanto de la plataforma como de cualquier herramienta concreta de prueba, se ha definido un conjunto de instrucciones para expresar las acciones entre la prueba y el sistema y las distintas comprobaciones que los árbitros pueden hacer.

Siguiendo el Testing Profile de UML, los modelos de interacción se expresan mediante diagramas de secuencia. Los árbitros, según su complejidad y reusabilidad, pueden incluirse en dicho

diagrama de secuencia o definirse aparte mediante nuevos diagramas de secuencia o diagramas de actividades.

6. Modelo de interfaz concreta y modelo de acciones.

Estos modelos permiten traducir las pruebas abstractas a pruebas ejecutables sobre el sistema. Para ello es necesario conocer las interfaces definitivas, incluir los detalles de dichos interfaces y completar las pruebas abstractas. El objetivo del modelo de interfaz concreta es expresar los elementos de la interfaz abstracta en función de los componentes concretos del sistema a probar. A partir de este modelo, ya se pueden expresar las pruebas a nivel de implementación. El objetivo del modelo de acción es expresar los elementos del modelo de interacción mediante un lenguaje de una herramienta de prueba concreta. Estos modelos dependen de la arquitectura y herramienta de prueba que se utilice para ejecutar las pruebas generadas, dado que las pruebas abstractas deben ser traducidas a pruebas comprensibles por dicha herramienta.

Herramienta para la Generación de Pruebas basadas en Modelo MDA.

Actualmente se está desarrollando un conjunto de herramientas para la generación de pruebas basada en los modelos. La primera herramienta desarrollada se llama ObjectGen y permite convertir un elemento del modelo de casos de uso en un elemento del modelo de comportamiento. Esta herramienta es de código abierto y puede descargarse libremente.

Siguiendo la metodología propuesta por MDA como se pudo apreciar se puede automatizar el proceso de pruebas del sistema trayendo esto consigo que estas se realicen en menos tiempo y que se genere la documentación de las mismas desde los artefactos previamente depurados y aceptados en una línea base de la arquitectura, indudablemente se introducirían menos errores en el momento de ejecutarse las pruebas y la cantidad de escenarios de prueba o posibles variantes a probar sería mayor que la que un ingeniero de pruebas pudiera ejecutar para encontrar posibles errores o fallas en el sistema. Si bien es irrefutable reconocer las ventajas de esta metodología para realizar las pruebas es necesario aclarar que aunque el desarrollo de sistemas utilizando metodologías ágiles, con MDA o con metodologías robustas no se puede renunciar a realizar dentro del proceso de pruebas un conjunto de prácticas o rutinas dentro de las pruebas que aún no han sido incorporadas a este procedimiento para realizarlas, se refiere a las pruebas de Caja Blanca, pruebas de estrés, pruebas de seguridad, pruebas de carga, pruebas a los instaladores de las

aplicaciones entre otras, que contribuyen a que el producto final tenga la calidad esperada. Incorporar esta metodología al proceso de la realización de las pruebas resulta beneficioso pero se recomienda la ejecución de las prácticas antes mencionadas, con o sin herramientas que la automaticen para alcanzar un mayor grado de efectividad de las pruebas.

2.8 Herramientas y Técnicas.

Selección de algunas herramientas y técnicas estudiadas en el epígrafe 1.4 a utilizar en las actividades de la propuesta del Plan de Aseguramiento de la Calidad para productos de software con Arquitectura MDA:

- Técnica para la planificación.

QFD: Quality Function Deployment, Despliegue de la Función de Calidad.
Como los objetivos de esta técnica es identificar los requisitos del cliente y proporcionar una disciplina para asegurar que estos requisitos estén presentes en el diseño del producto y en el proceso de planificación. Su utilización en cualquier producto de software con Arquitectura MDA, ayudaría a identificar los requisitos y asegurar que estén presentes en el diseño del modelo CIM que es donde se representan los requisitos de los clientes dentro de esta arquitectura.
Además de reducir los ciclos de desarrollo de productos, aumentando la calidad y disminuyendo los costes.

- Técnica para el control.

Auditoría de calidad: Esta técnica se basa en la realización de exámenes metódicos e independientes que se realizan para determinar si las actividades y los resultados relativos a la calidad satisfacen las disposiciones previamente establecidas y para comprobar que estas disposiciones se llevan a cabo eficazmente y que son adecuadas para alcanzar los objetivos previstos del producto de software.

- Técnica para la mejora y resolución de problemas.

Reingeniería: Las funciones principales de esta técnica son las revisiones fundamentalmente y el rediseño radical de procesos para alcanzar mejoras espectaculares en medidas críticas y contemporáneas de rendimiento, tales como costes, calidad, servicio y rapidez. La implementación de esta técnica en los proyectos de software con Arquitectura MDA ayudará a la resolución de cualquier problema surgido en el diseño de los modelos.

2.9 Gestión de Configuración.

En el desarrollo de software, los cambios, debidos principalmente a modificaciones de requisitos y fallos, son inevitables. Normalmente se trabaja en equipo por lo que es preciso llevar un control y registro de los cambios con el fin de reducir errores, aumentar la calidad y la productividad y evitar los problemas que puede acarrear una incorrecta sincronización en dichos cambios, al afectar a otros elementos del sistema o a las tareas realizadas por otros miembros del equipo de proyecto.

En la sección 1.3 se definió la Gestión de la Configuración como la encargada de controlar los cambios que se le van efectuando a cada elemento del sistema durante todo el ciclo de vida del proceso de software.

El objetivo de la Gestión de la Configuración es mantener la integridad de los productos que se obtienen a lo largo del desarrollo de los sistemas de información, garantizando que no se realizan cambios incontrolados y que todos los participantes en el desarrollo del sistema disponen de la versión adecuada de los productos que manejan. Así, entre los elementos de configuración de software, se encuentran no únicamente ejecutables y código fuente, sino también los modelos de datos, modelos de procesos, especificaciones de requisitos, pruebas, etc.

La Gestión de Configuración se realiza durante todas las actividades asociadas al desarrollo del sistema, y continúa registrando los cambios hasta que éste deja de utilizarse.

La Gestión de Configuración facilita el mantenimiento del sistema, aportando información precisa para valorar el impacto de los cambios solicitados y reduciendo el tiempo de implementación de un

cambio, tanto evolutivo como correctivo. Así mismo, permite controlar el sistema como producto global a lo largo de su desarrollo, obtener informes sobre el estado de desarrollo en que se encuentra y reducir el número de errores de adaptación del sistema, lo que se traduce en un aumento de calidad del producto, de la satisfacción del cliente y, en consecuencia, de mejora de la organización.
(PÚBLICAS 2006)

Debido al gran trabajo y esfuerzo que requiere la Gestión de la Configuración por todas las actividades que en ella están implícitas como:

- Identificación de la Configuración.
- Control de la Configuración.
- Estado de la Configuración.
- Auditorías a la Configuración.

En esta investigación no se define el proceso de Gestión de la Configuración para proyectos de software sobre la Arquitectura MDA dejando abierto este punto a una nueva investigación.

2.10 Entrenamiento.

Es necesario que el personal involucrado en el desarrollo del proyecto alcance el nivel de conocimientos requeridos para el cumplimiento de sus actividades según los roles que desempeñe, para ello se implementa un Plan de Capacitación con los objetivos de desarrollar actividades de entrenamiento para que el equipo de proyecto ejecute las tareas del Plan de Aseguramiento de la Calidad para proyectos sobre Arquitectura MDA. Este Plan de Capacitación tiene como objetivos:

- Fortalecer la calidad operacional de todo el personal involucrado con el desarrollo del producto de software con Arquitectura MDA.
- Lograr que el personal trabaje para alcanzar un software de calidad.

Para el cumplimiento de estos objetivos es necesario que todos los trabajadores del proyecto cursen las asignaturas del perfil de calidad que sigue la Universidad, estas asignaturas son:

- Introducción a la Calidad.

Caracterizar los elementos conceptuales asociados a la Calidad y en especial a la calidad del software y los procesos asociados al Aseguramiento de la Calidad.

- Proceso de Desarrollo de Software.

Caracterizar el Proceso de Desarrollo de Software, adquiriendo una visión de la producción de software, así como las principales metodologías que se utilizan para modelar el ciclo de vida de un software, haciendo hincapiés en el RUP como proceso, y en sus principales artefactos.

- Pruebas de Software.

Definir los conceptos de pruebas de software, caso de uso, caso de pruebas, estrategias de pruebas, niveles, métodos y tipos de pruebas. Caracterizar el proceso de pruebas de software en la UCI. Aplicar Casos de pruebas y listas de chequeos. Elaborar el reporte de no conformidad.

- Administración de la Calidad de Software.

Desempeñar los diversos roles dentro del equipo SQA aplicando técnicas modernas y efectivas en la realización de sus funciones: Administrar elementos y funciones del Aseguramiento de la Calidad del Software; valorar el estado de un proceso a través de evaluaciones. Reflexionar acerca de los retos y prácticas alrededor de la Calidad del Proceso Software tomando como referencia los principios y procesos definidos en los modelos internacionales de calidad de software.

- Administración de Configuración de Software.

Caracterizar el Proceso de Gestión de Configuración de Software, describiendo los artefactos que se generan como resultado de dicho proceso. Identificar cada uno de los roles que intervienen en la Gestión de Configuración, y las responsabilidades que tienen asignadas. Estudiar algunas herramientas para automatizar el proceso.

Además de impartir talleres, cursos y conferencias para hacerle llegar los conocimientos necesarios a los trabajadores de forma didáctica para el desarrollo de un proyecto sobre la Arquitectura MDA. Para ello se desarrollarán:

1. Curso de Aseguramiento de la Calidad.

Definir cada una de las actividades de Aseguramiento de la Calidad.

2. Taller Introductorio a la Arquitectura MDA.

Definir la Arquitectura MDA, en qué esta basada, explicar cada uno de sus modelos, su importancia y características fundamentales.

3. Curso de Métricas.

Estudiar diferentes métricas para evaluar los requisitos de calidad y el diseño del software, así como parámetros para el Aseguramiento de la Calidad.

4. Conferencia de Estándares y Guías.

Estudiar distintos estándares y guías que servirán de apoyo para el desarrollo de las actividades de cada trabajador según el rol que desempeñe.

5. Taller de Pruebas.

Abordar los tipos y estrategias de pruebas específicas a realizar a un software con la Arquitectura MDA.

6. Curso de Herramientas y Técnicas.

Realizar un estudio de las diferentes técnicas y herramientas a utilizar de acuerdo a su funcionabilidad para el desarrollo del software.

2.11 Conclusiones.

En la elaboración de la propuesta del Plan de Aseguramiento de la Calidad para proyectos sobre Arquitectura MDA se identificaron los requisitos de calidad que son significativos para MDA, se describió la estructura de organización de forma general, las tareas de Aseguramiento de la Calidad y el responsable de su ejecución, se proponen métricas para evaluar los requisitos de calidad, métricas para el Aseguramiento de la Calidad y métricas para evaluar el diseño, dentro de ellas: para el diseño de la arquitectura y el diseño a nivel de componentes, se listaron de los estándares y guías a utilizar para en desarrollo de un proyecto y la elaboración de su Plan de Aseguramiento de

la Calidad, se describió el procedimiento “Auditorías a la Actividad Productiva” que se realiza a nivel central de la Universidad para las Revisiones y Auditorías de cada proyecto de software, además de recomendar las Inspecciones del Colectivo y dentro de ellas las revisiones de la arquitectura y el diseño para encontrar los defectos antes la codificación generada por la Arquitectura MDA, se definieron los tipos de pruebas a realizar, se abordó la Gestión de la Configuración dejando abierto a la investigación este punto dentro de la propuesta, para su desarrollo en proyectos con Arquitectura MDA, y se implementó un Plan de Capacitación para capacitar a todos los trabajadores involucrados de los conocimientos necesarios para el desarrollo de cualquier proyecto con Arquitectura MDA.

Capítulo 3: "Validación de la Propuesta de Solución".

3.1 Introducción.

Para validar la propuesta de solución, se utilizó el Método de Experto el cuál se fundamenta en la evaluación cuantitativa de criterios definidos en la investigación, y que permiten a los expertos realizar un estudio para determinar si se acepta o no la propuesta analizada.

3.2 Guía para la validación.

Para realizar la validación se efectuaron un conjunto de pasos, los cuales se especifican a continuación:

1. Se elaboraron los criterios de evaluación de acuerdo a las características de la propuesta y se organizaron en grupos acordes a sus categorías.

Grupo No 1: Criterios de mérito científico.

1. Valor científico.
2. Calidad de la investigación.
3. Contribución científica.
4. Responsabilidad científica y profesionalidad del investigador.

Grupo No 2: Criterios de implantación.

5. Necesidad de creación.
6. Satisfacción de las necesidades de los ingenieros de software.

Grupo No 3: Criterios de flexibilidad.

7. Adaptabilidad a proyectos productivos que utilicen Arquitectura MDA.
8. Fácil entendimiento.
9. Facilidad de aplicación.
10. Factible interpretación de los resultados después de la aplicación.

Grupo No 4. Criterios de impacto.

11. Aceptación de la propuesta.
12. Posibilidades de aplicación.
13. Organización en el proceso de aplicación.
14. Impacto en el área a la cuál está destinada.

2. Se le asignó un peso relativo de cada grupo de criterios de acuerdo al porciento que representa cada grupo del total y los intereses a evaluar.

Grupo No.1…… 30
Grupo No.2…… 15
Grupo No.3…… 30
Grupo No.4…… 25

3. Se realizó la selección del comité con una cantidad mínima de 7 expertos teniendo en cuenta su especialidad, grado científico y currículo.

4. Se le entregó a cada experto la propuesta de solución para el estudio del tema a evaluar y dos modelos, uno para la valoración del peso relativo de cada criterio (Anexo 9) y otro para la evaluación cuantitativa de cada criterio con una escala de 1-5 y la apreciación cualitativa con una clasificación final del proyecto en excelente, bueno, aceptable, cuestionable y malo (Anexo 10).

5. Después de recibir los valores del peso relativo de cada criterio se construyó la Tabla 9.

G	C/E	E1	E2	E3	E4	E5	E6	E7	Ep
1	C1	5	7	7	6	7	8	7	6.71
	C2	10	9	9	8	8	7	6	8.14
	C3	5	7	7	8	7	7	8	7
	C4	10	7	7	8	8	8	9	8.14
2	C5	8	8	8	8	10	6	8	8
	C6	7	7	7	7	5	9	7	7
3	C7	8	8	8	8	9	6	7	7.71
	C8	7	8	8	8	7	8	8	7.71
	C9	8	7	7	8	8	8	7	7.57
	C10	7	7	7	6	6	8	8	7
4	C11	7	6	6	6	7	5	6	6.14
	C12	5	7	6	7	8	6	6	6.42
	C13	7	6	6	6	5	7	6	6.14
	C14	6	6	7	6	5	7	7	6.28
T		100	100	100	100	100	100	100	100

Tabla 9: Resultado del trabajo de expertos.

6. Se verificó la consistencia en el trabajo de los expertos, para lo que se utilizó el coeficiente de concordancia de Kendall y el estadígrafo Chi cuadrado (X^2).

Para ello se siguió el procedimiento siguiente:

- Sea C el número de criterios que se evaluó y E el número de expertos que realizaron la evaluación.

- Para cada criterio se determinó:

  ∑E: Sumatoria del peso dado por cada experto.

  Ep: Puntuación promedio del peso dado por cada experto.

  M∑E: media de los ∑E.

  ΔC: Diferencia entre ∑E y M∑E.

- Se determinó la desviación de la media, que posteriormente se eleva al cuadrado para obtener la dispersión (S) por la expresión (Tabla 10).

$$S = \sum (\sum E - \sum \sum E / C)2$$

	∑E	∑E/C	∑E - ∑ ∑E / C	$(\sum E - \sum \sum E / C)^2$
C1	47	3.3571	-2.9996	8.9976
C2	57	4.0714	7.0004	49.0056
C3	49	3.5	0.9996	0.9992
C4	57	4.0714	7.0004	49.0056
C5	56	4	6.0004	36.0048
C6	49	3.5	0.9996	0.9992
C7	54	3.8571	4.0004	16.0032
C8	54	3.8571	4.0004	16.0032
C9	53	3.7857	3.0004	9.0024
C10	49	3.5	0.9996	0.9992
C11	43	3.0714	-6.9996	48.9944
C12	45	3.2142	-4.9996	24.9960
C13	43	3.0714	-6.9996	48.9944
C14	44	3.1428	-5.9996	35.9952
∑ ∑E / C		49.9996		
$S = \sum (\sum E - \sum \sum E / C)^2$				346

Tabla 10: Cálculo de la Dispersión (S) para hallar la concordancia entre los expertos.

- Conociendo la dispersión se calculó el coeficiente de concordancia de Kendall (W)

$$W = S / E^2 (C^3 – C)/ 12$$

- Con el coeficiente de concordancia de Kendall se calculó el Chi cuadrado real

$$X^2 = E (C-1) W$$

- Los valores obtenidos se muestran en la Tabla 11.

Expertos/Criterios	E_1	E_2	E_3	E_4	E_5	E_6	E_7	∑E	Ep	ΔC	ΔC^2
C_1	5	7	7	6	7	8	7	47	6.71	2.7515	7.5707
C_2	10	9	9	8	8	7	6	57	8.14	8.7515	76.588
C_3	5	7	7	8	7	7	8	49	7	0.7515	0.5647
C_4	10	7	7	8	8	8	9	57	8.14	8.7515	76.5887
C_5	8	8	8	8	10	6	8	56	8	7.7515	60.0857
C_6	7	7	7	7	5	9	7	49	7	0.7515	0.5647
C_7	8	8	8	8	9	6	7	54	7.71	5.7515	33.0797
C_8	7	8	8	8	7	8	8	54	7.71	5.7515	33.0797
C_9	8	7	7	8	8	8	7	53	7.57	4.7515	22.5767
C_{10}	7	7	7	6	6	8	8	49	7	0.7515	0.56475
C_{11}	7	6	6	6	7	5	6	43	6.14	6.7515	45.5827
C_{12}	5	7	6	7	8	6	6	45	6.42	4.7515	22.5767
C_{13}	7	6	6	6	5	7	6	43	6.14	6.7515	45.5827
C_{14}	6	6	7	6	5	7	7	44	6.28	5.7515	33.0797
DC	100	100	100	100	100	100	100	700	100	70.521	458.0865
M ∑E	49.7515										
W	0.0310										
X^2	2.821										
$X^2_{(\alpha, c-1)}$	27,6882										

Tabla 11: Cálculo de concordancia de Kendall.

- Con el Chi cuadrado calculado se comparó el obtenido de las tablas estadísticas.

- Como se cumple: $X^2_{real} < X^2_{(\alpha, c-1)}$
  Existe concordancia en el trabajo de expertos.

7. Después de comprobar la consistencia del trabajo de expertos se pudo definir el peso relativo de cada criterio (P).

8. Conociendo el peso de cada criterio y la calificación dada por los evaluadores en una escala de 1-5 se pudo construir la Tabla 12, para obtener el valor de P × c., donde (c), es el criterio promedio concebido por los expertos y se calculó el Índice de Aceptación (IA).
Donde IA = ∑ (P × c) / 5.

Criterios	Calificación (c)					P	P × c
	1	2	3	4	5		
C_1				x		0.0671	0.2684
C_2					x	0.0814	0.407
C_3				x		0.07	0.28
C_4				x		0.0814	0.3256
C_5					x	0.08	0.4
C_6				x		0.07	0.28
C_7					x	0.0771	0.3855
C_8					x	0.0771	0.3855
C_9				x		0.0757	0.3028
C_{10}				x		0.07	0.28
C_{11}				x		0.0614	0.2456
C_{12}					x	0.0642	0.321
C_{13}				x		0.0614	0.2456
C_{14}					x	0.0628	0.314
∑ (P × c)	4.441						
IA	0.8882						

Tabla 12: Calificación de cada criterio.

9. Por último se determinó la probabilidad de éxito de la propuesta según los rangos predefinidos de Índice de Aceptación:

Rangos predefinidos de Índice de Aceptación.
IA > 0,7 Existe alta probabilidad de éxito
0,7 > IA > 0,5 Existe probabilidad media de éxito
0,5 > IA > 0,3 Probabilidad de éxito baja

0,3 > IA Fracaso seguro

Como IA = 0.882, se ubica en el primer rango y la probabilidad de éxito de la propuesta es alta.

3.3 Conclusiones.

En este capítulo se evaluó la propuesta del Plan de Aseguramiento de la Calidad para proyectos de software sobre la Arquitectura MDA mediante el Método de Experto, el cual posibilitó realizar un análisis de los criterios de cada uno de los expertos consultados y determinar el Índice de Aceptación que tiene la propuesta de la investigación, obteniéndose concordancia en el trabajo realizado y una alta probabilidad de éxito de ser implementado en los proyectos de software sobre la Arquitectura MDA.

Conclusiones Generales

Durante el desarrollo de la investigación se cumplieron en su totalidad los objetivos trazados. Se logró caracterizar la Arquitectura MDA y las actividades de Aseguramiento de la Calidad, se identificaron aspectos de la Arquitectura MDA sujetos al Aseguramiento de la Calidad y se elaboró la propuesta del Plan de Aseguramiento de la Calidad para proyectos de software sobre la Arquitectura MDA. Concluyendo así que con la aplicación del Plan propuesto:

- Se aportará confianza en que el producto de software satisface los requisitos establecidos de acuerdo a las necesidades del cliente.
- Se logrará en cumplimiento de los objetivos de calidad.

Recomendaciones

- Definir las actividades de Gestión de Configuración específicas para proyectos sobre la Arquitectura MDA.
- Realizar estudios sobre los estándares existentes a nivel internacional para el Aseguramiento de la Calidad de proyectos de software.
- Realizar estudios sobre la herramienta ObjectGen para la generación de pruebas basada en los modelos.

Referencias Bibliográficas

9126-1, N. C. I. I. *CALIDAD DEL PRODUCTO. PARTE 1: MODELO DE LA CALIDAD*, 2005.

9126-3, I. Métricas Internas. .

9126, I., 2001.

ACEVEDO, R. V. *Mejoramiento del Proceso de Gestión de Configuración de Software*, 2004. 21.

AGÜERO, I. D. N. Áreas del aseguramiento de la calidad., 2008.

ANACLETO, V. A. *MDA: Reusabilidad Orientada al Negocio* 2006.

COLSA, L. E. C. D. Arquitectura dirigida por modelos para J2ME.

CORREDERA, L. E. *Arquitectura dirigida por modelos*, 2006.

DELGADO, M. *Inspecciones a Proyectos de Software, Garantía de Calidad.*, 2003.

FOMENTO, I. A. D. *Herramientas para la Gestión de la Calidad* 2008.

GARCÍA, J. *CONTROL Y GESTIÓN DEL ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE*, 2005.

GRACIA, J. *CMM - CMMI*, 2005.

HUMPHREY, W. S. *Introducción al Proceso Software Personal.*, p.

INTERNACIONAL, H. *Gestión de Configuración del Software*, 2007. [Disponible en: http://www.histaintl.com/soluciones/configuracion/configuracion.php

ISO. 1994.

JAVIER J. GUTIÉRREZ, M. J. E., MANUEL MEJÍAS Y ANTONIA M. REINA. *MODELOS DE PRUEBAS PARA PRUEBAS DEL SISTEMA*, 2006.

LÓPEZ, C., 2002. [Disponible en: http://www.gestiopolis.com/recursos/experto/catsexp/pagans/ger/42/calidad.htm

LOVELLE, J. M. C. *Calidad del Software*, 1999. 12.

MALEVSKI, Y. *Manual de Gestión de la Calidad Total a la Medida* 1995. p.

MARÍA LUISA GARZÓNVILLAR, E. L. C., IGNACIO PRIETO TINOCO, MARÍA DE LOS ANGELES SAMPALO DE LA TORRE *Informática. Temario para la preparación de oposiciones.*, 2005. 342 p.

MARTHA DUNIA DELGADO DAPENA, S. Á. C. Y. A. R. S. Propuesta de introducción de las revisiones en el proceso de desarrollo de software. , 2005.

PRESSMAN, R. S. *Ingeniería del Software. Un enfoque práctico.* 3ra. 1993. p.

PÚBLICAS, M. D. A. Gestión de la Configuración, 2006.

STD-730, I. *Plan de aseguramiento de la calidad del software.*, 1998.

UCI, D. D. C. *Métricas para el SQA*.

Bibliografía

- Arter, D. “Auditorías De Calidad Para Mejorar La Productividad". 2003.
- BEDINI, A. Extracto del libro en formato digital "Calidad tradicional y de Software".
- CUEVAS, A. B. M. Y. J. M. "Estándares y Guías", 2001.
- DIEZ, E. “Calidad del Software. CMM: Capability Maturity Model”.
- Fernández, L; Calvo-Manzano, J; “Ingeniería del software: Pruebas del Software”. 1996.
- Folgar, O. “Aseguramiento de la Calidad: ISO 9000”. 1996.
- Humphrey, W. “Introducción al Proceso de Software Personal”, 2001.
- Humphrey, W. “A self improvement Process for Software Engineers”, 2005.
- ISO/IEC 9126: Tecnología de la información. Evaluación del producto software. Características y directrices de calidad para su uso, 1991.
- Jacobson, I.; Booch, G. y Rumbaugh, J. “El Proceso Unificado de Desarrollo de software”. 2000.
- Minguet, J; Hernández, J; “La calidad del software y su medida”. 2003.
- Nava, V M. ¿Qué es la Calidad?. 2005.
- Pressman, R. “Ingeniería de Software: un enfoque práctico”, 3ra. Edición.
- Pressman, R. “Ingeniería de Software: un enfoque práctico”, 5ta. Edición.
- Sanguesa, M; Mateo, R; Ilzarbe, L. “Teoría y Práctica de la Calidad”. 2006.
- Stebbing, L. “Aseguramiento de la Calidad: El camino a la eficiencia y la competitividad”. 1991.

Glosario de Términos

Cliente: Una persona u organización, interna o externa a la organización productora que toma responsabilidad financiera por el sistema. El cliente es el último destinatario del producto desarrollado y sus artefactos.

Estándar: Que tiene el tamaño, la forma o cualquier otra característica que sigue al modelo. Se aplica a lo que se produce en serie. Que sigue una tendencia muy extendida. Aquello que se considera modelo.

Framework: Esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación.

Herramienta: Software que se utiliza para automatizar las actividades definidas en el proceso.

Interfaz: Colección de operaciones que son usadas para especificar un servicio de una clase o un componente.

Modelo: Cosa que ha de servir de objeto de imitación. Objeto, construcción u otra cosa con un diseño del que se reproduce más iguales. Esquema teórico de un sistema o de una realidad compleja que se elabora para facilitar su comprensión y estudio.

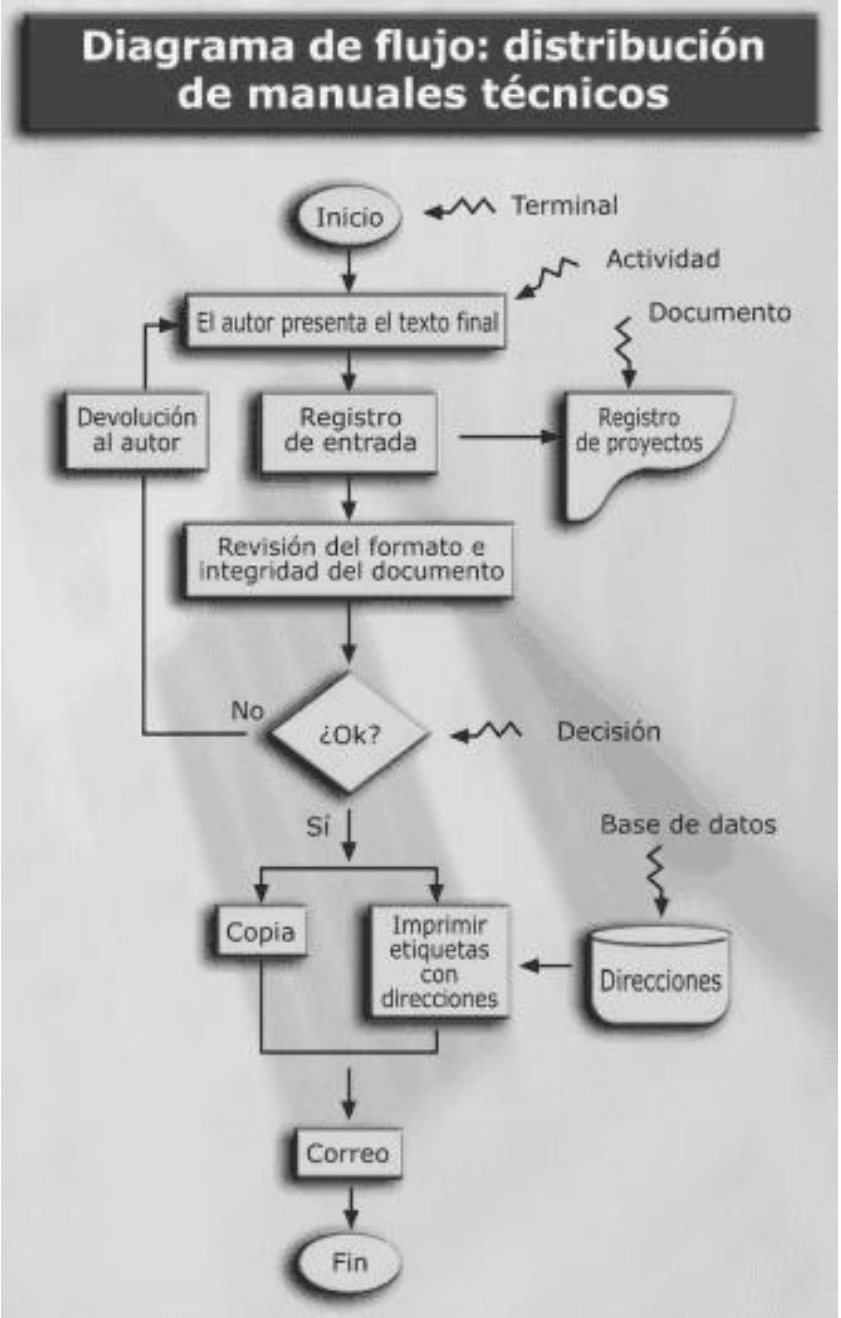
Rol: Papel, cometido o función que tiene o desempeña que interpreta un actor.

Sistema: Entidad material formada por partes organizadas (o sus "componentes") que interactúan entre sí de manera que las propiedades del conjunto, sin contradecirlas, no pueden deducirse por completo de las propiedades de las partes.

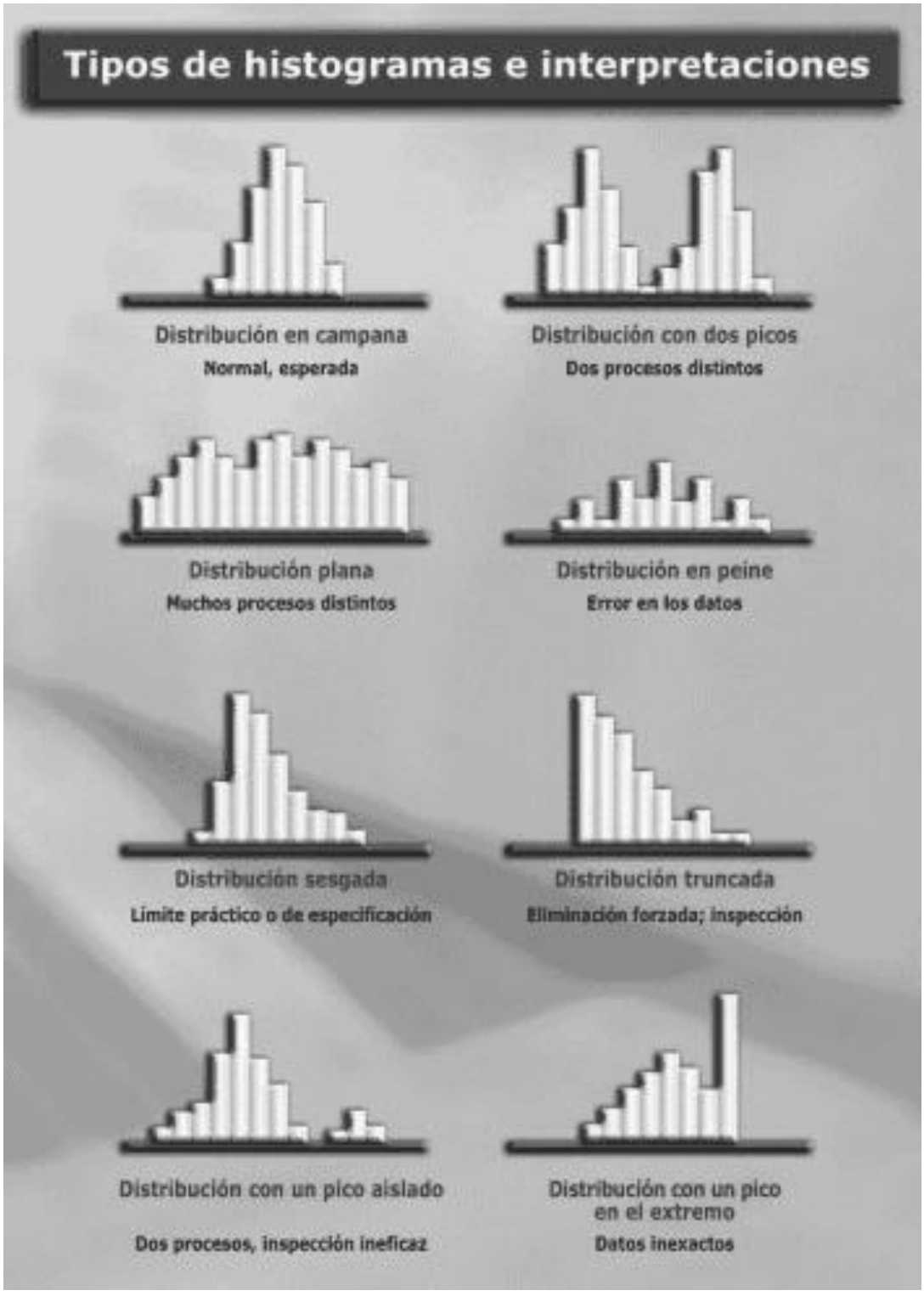
SQA: Software Quality Assurance, en español Aseguramiento de la Calidad del Software.

Stakeholders: Personas u organizaciones que están activamente implicadas en el negocio ya sea porque participan en él o porque sus intereses se ven afectados con los resultados del proyecto. Pueden ser los propietarios, la dirección, quienes financian, los clientes, los trabajadores, los proveedores, la competencia, la comunidad local, etc.

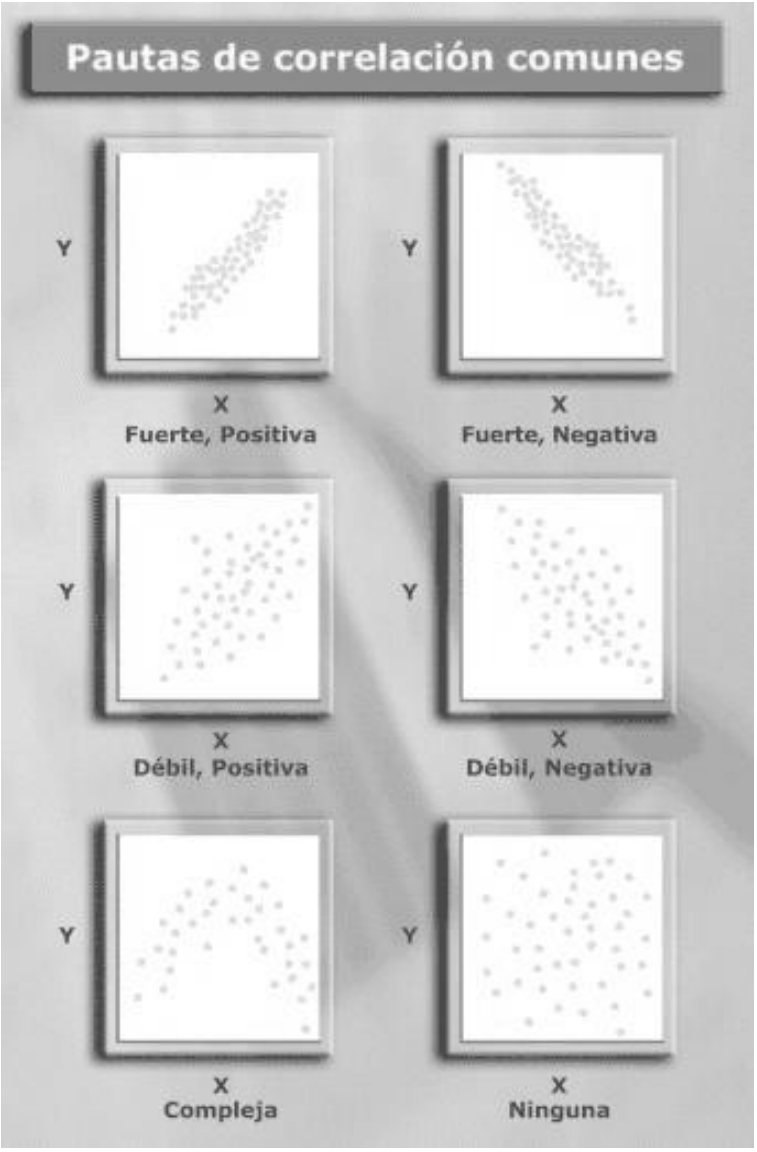
Anexo 1
Ejemplo de cómo se utiliza la Herramienta Diagrama de Flujo.



Anexo 2
Ejemplo de cómo se utiliza la Herramienta Histogramas.

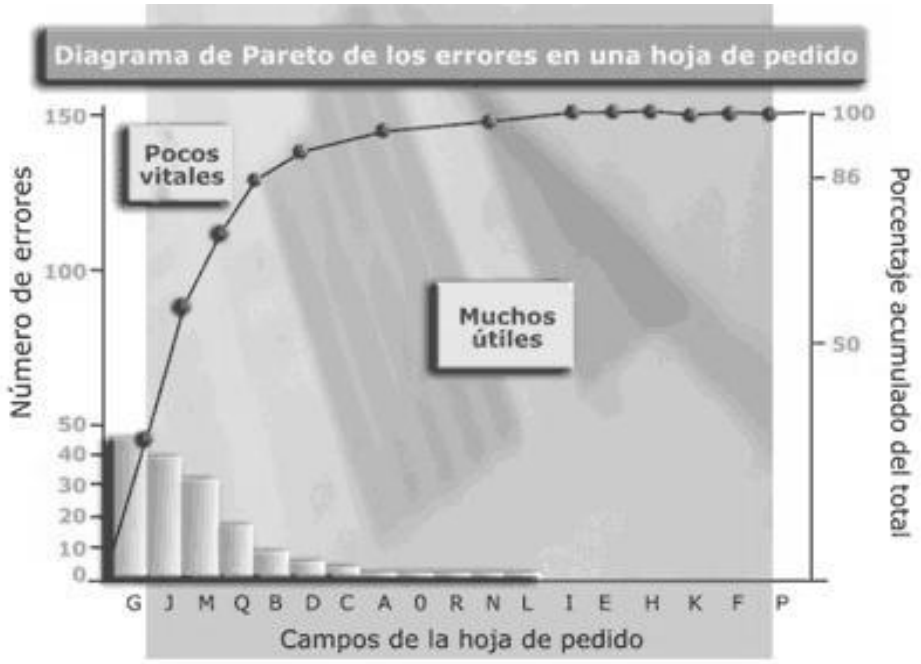


Anexo 3
Ejemplo de cómo se utiliza la Herramienta Diagrama de Dispersión.



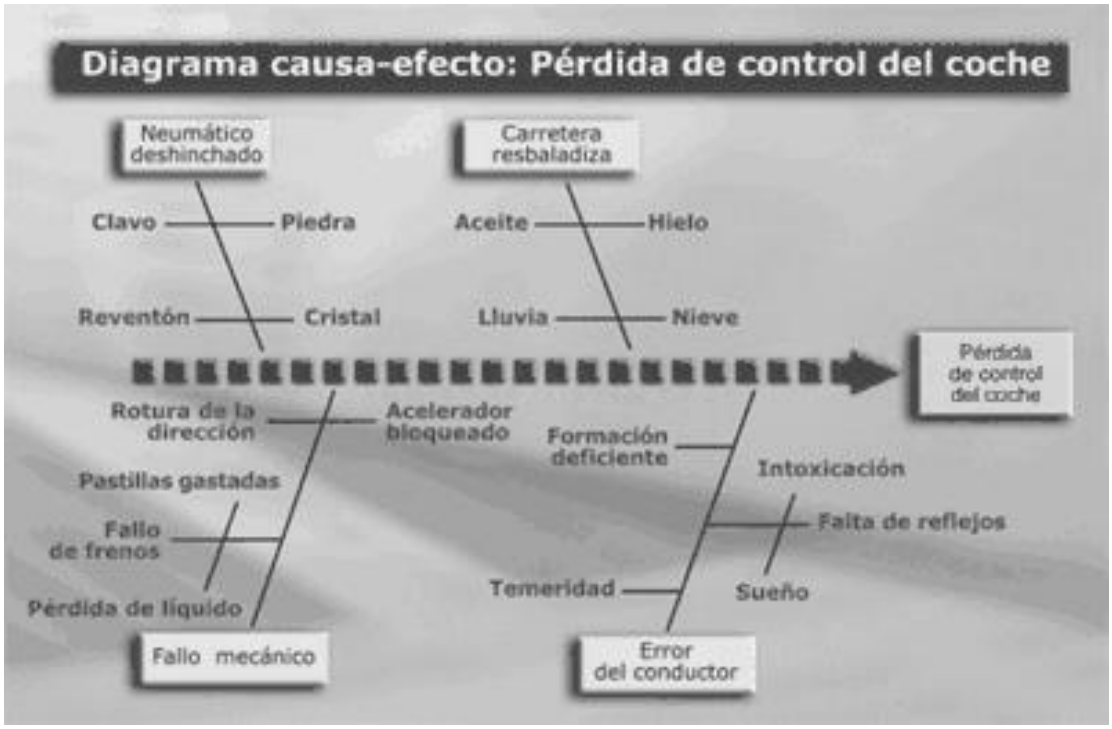
Anexo 4

Ejemplo de cómo se utiliza la Herramienta Análisis de Pareto.

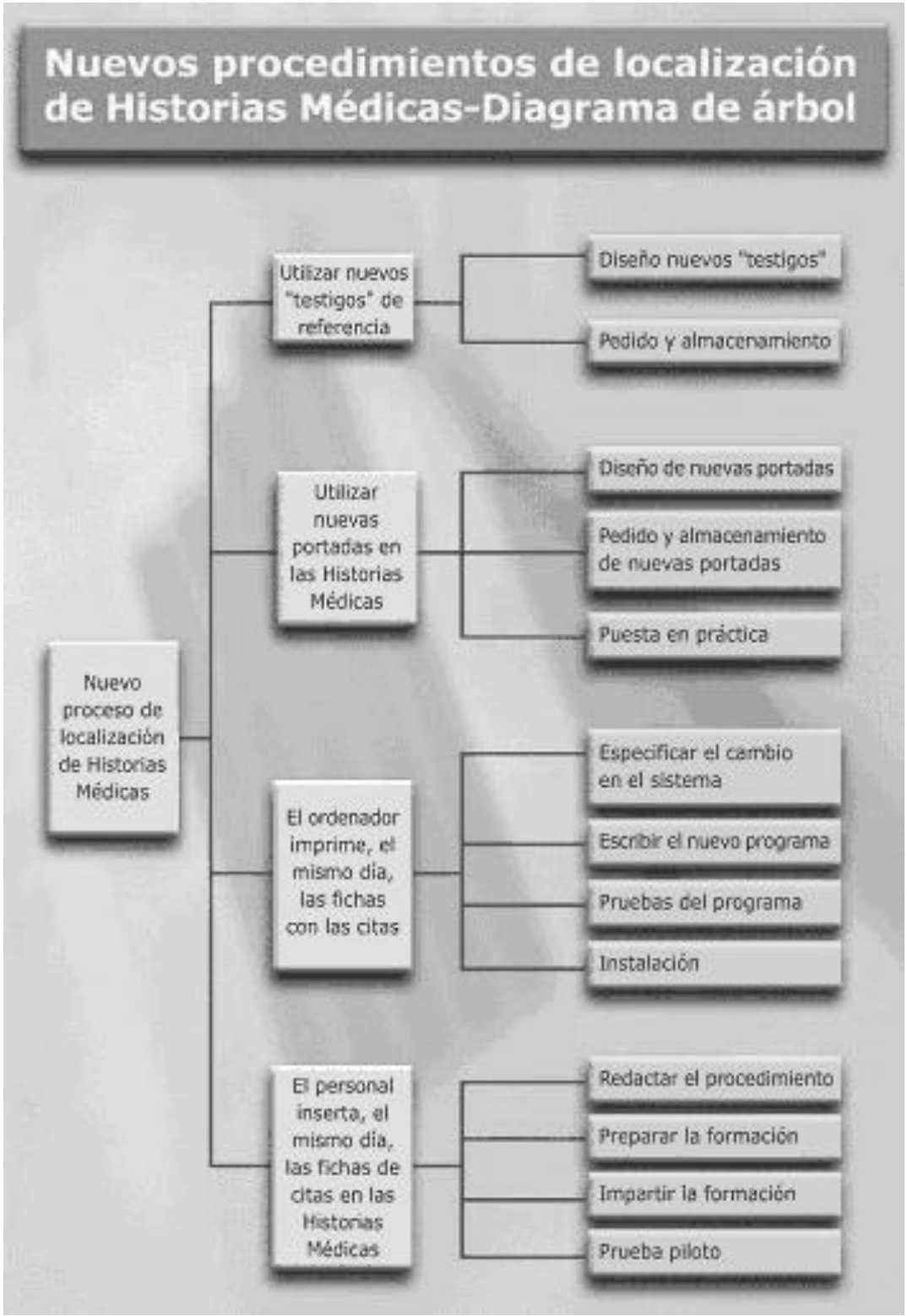


Anexo 5

Ejemplo de cómo se utiliza la Herramienta Diagrama Cauca-Efecto.



Anexo 6
Ejemplo de cómo se utiliza la Herramienta Diagrama de Árbol.



Anexo 7

Definiciones de las Características y Sub-características de calidad propuestas por el estándar ISO/IEC 9126.

La siguiente tabla muestra las definiciones de las características y sub-características de calidad del estándar ISO/IEC 9126 [NC-ISO/IEC 9126-1, 2005]:

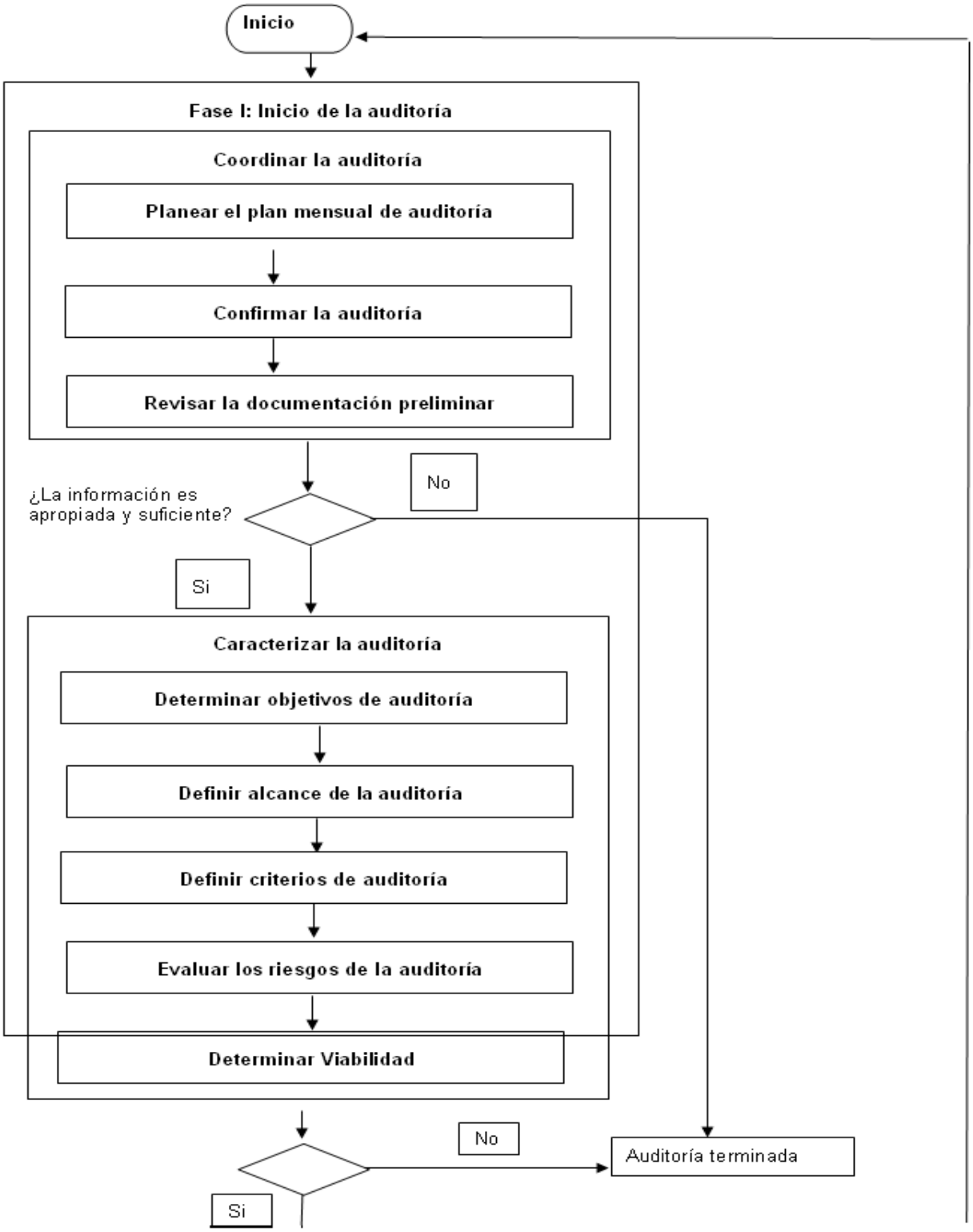
1. **Funcionalidad**	
	Es la capacidad del software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas cuándo el software se usa bajo las condiciones especificadas. Es el grado en que el software satisface las necesidades indicadas por las siguientes sub-características:
a. Idoneidad	Capacidad del software para mantener un conjunto apropiado de funciones para las tareas y los objetivos del usuario especificados.
b. Exactitud	Capacidad del software para proporcionar efectos o resultados correctos o convenidos con el grado de exactitud necesario.
c. Interoperabilidad	Capacidad del software para interactuar recíprocamente con uno o más sistemas especificados.
d. Seguridad (informática)	Capacidad del producto software para proteger información y los datos, para que personas o sistemas desautorizados no puedan leer o pueden modificar los mismos, y las personas o sistemas autorizados tenga el acceso a ellos.
e. Conformidad con la funcionalidad	Capacidad del software para adherirse a las normas que se le apliquen, convenciones, regulaciones, leyes y las prescripciones similares relativas a la funcionalidad.
2. **Confiabilidad o Fiabilidad**	
	Es la capacidad del producto software para mantener un nivel de ejecución especificado cuando se usa bajo las condiciones especificadas. Cantidad de tiempo que el software está disponible para su uso. Se refiere a las sub-características:
a. Madurez	Capacidad del producto software de evitar un fallo total como resultado de haberse producido un fallo del software.
b. Tolerancia ante fallos	Capacidad del producto software de mantener un nivel de ejecución o desempeño especificado en caso de fallos del software o de infracción de su interfaz especificada.

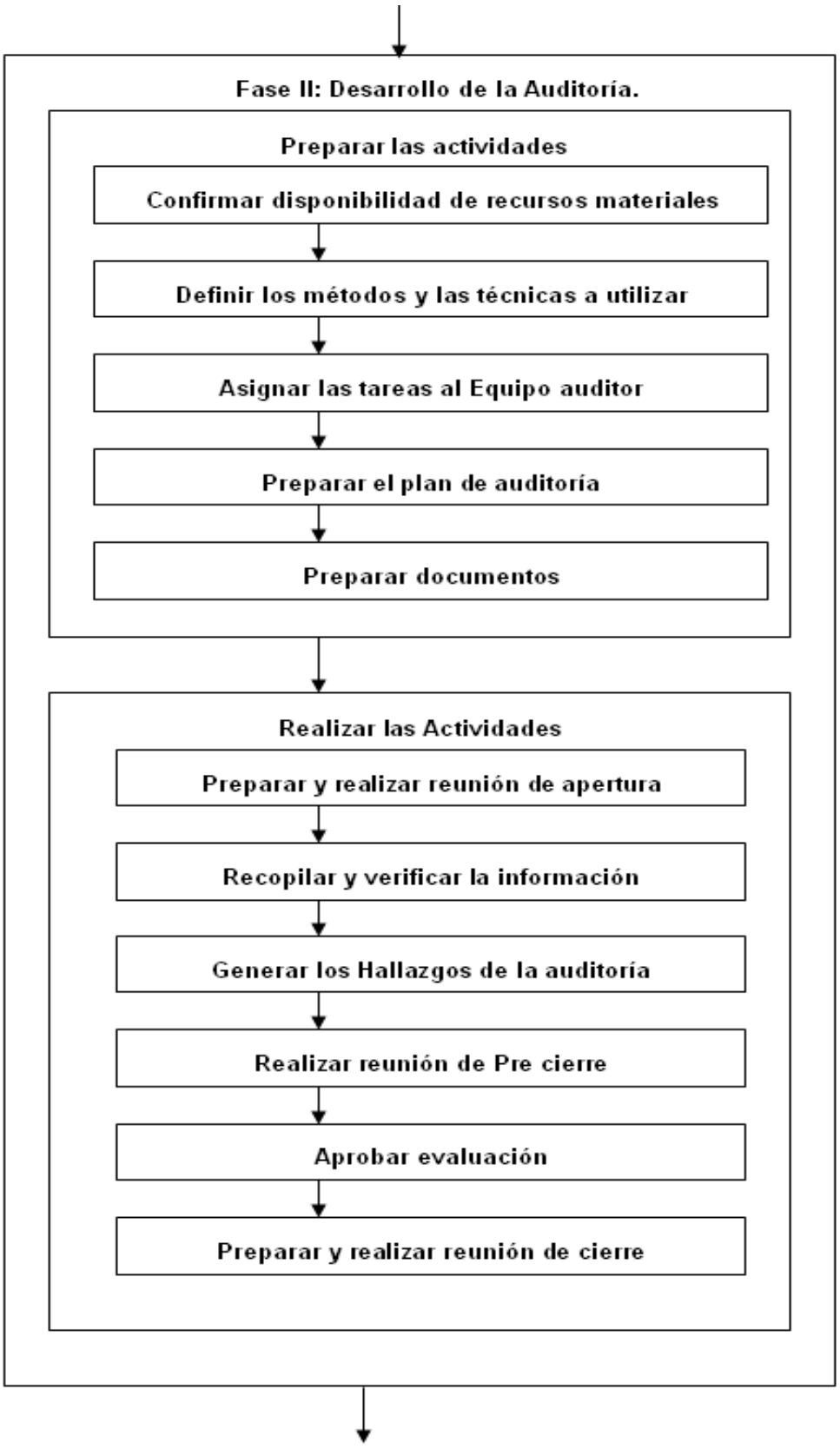
c. Recuperabilidad	Capacidad del producto software de restablecer un nivel de ejecución especificado y recuperar los datos directamente afectados en caso de fallo total.
d. Conformidad con la confiabilidad	Capacidad del producto software para adherirse a las normas que se le apliquen, convenciones, regulaciones, leyes y las prescripciones similares relativas a la confiabilidad.
3. **Usabilidad**	
	Grado con que el software es fácil de usar. Viene reflejado por las siguientes sub-características:
a. Comprensibilidad	Capacidad del producto software para permitirle al usuario entender si el software es idóneo, y cómo puede usarse para las tareas y condiciones de uso particulares.
b. Cognoscibilidad	Capacidad del producto software para permitirle al usuario aprender su aplicación.
c. Operabilidad	Capacidad del producto software para permitirle al usuario operarlo y controlarlo.
d. Atracción	Capacidad del producto software de ser atractivo o amigable para el usuario.
e. Conformidad con la usabilidad	Capacidad del producto software para adherirse a las normas, convenciones, guías de estilo o regulaciones relativas a la usabilidad.
4. **Eficiencia**	
	Capacidad del producto software para proporcionar una ejecución o desempeño apropiado. Grado con que el software hace óptimo el uso de los recursos del sistema. Esta característica está indicada por las siguientes sub-características:
a. Rendimiento	Capacidad del producto software para proporcionar apropiados tiempos de respuesta y procesamiento, así como tasas de producción de resultados, al realizar su función bajo condiciones establecidas.
b. Utilización de recursos	Capacidad del producto software para utilizar la cantidad y el tipo apropiado de recursos cuando el software realiza su función bajo las condiciones establecidas.
c. Conformidad con la eficiencia	Capacidad del producto de software de adherirse a las normas o convenciones que se relacionan con la eficiencia.

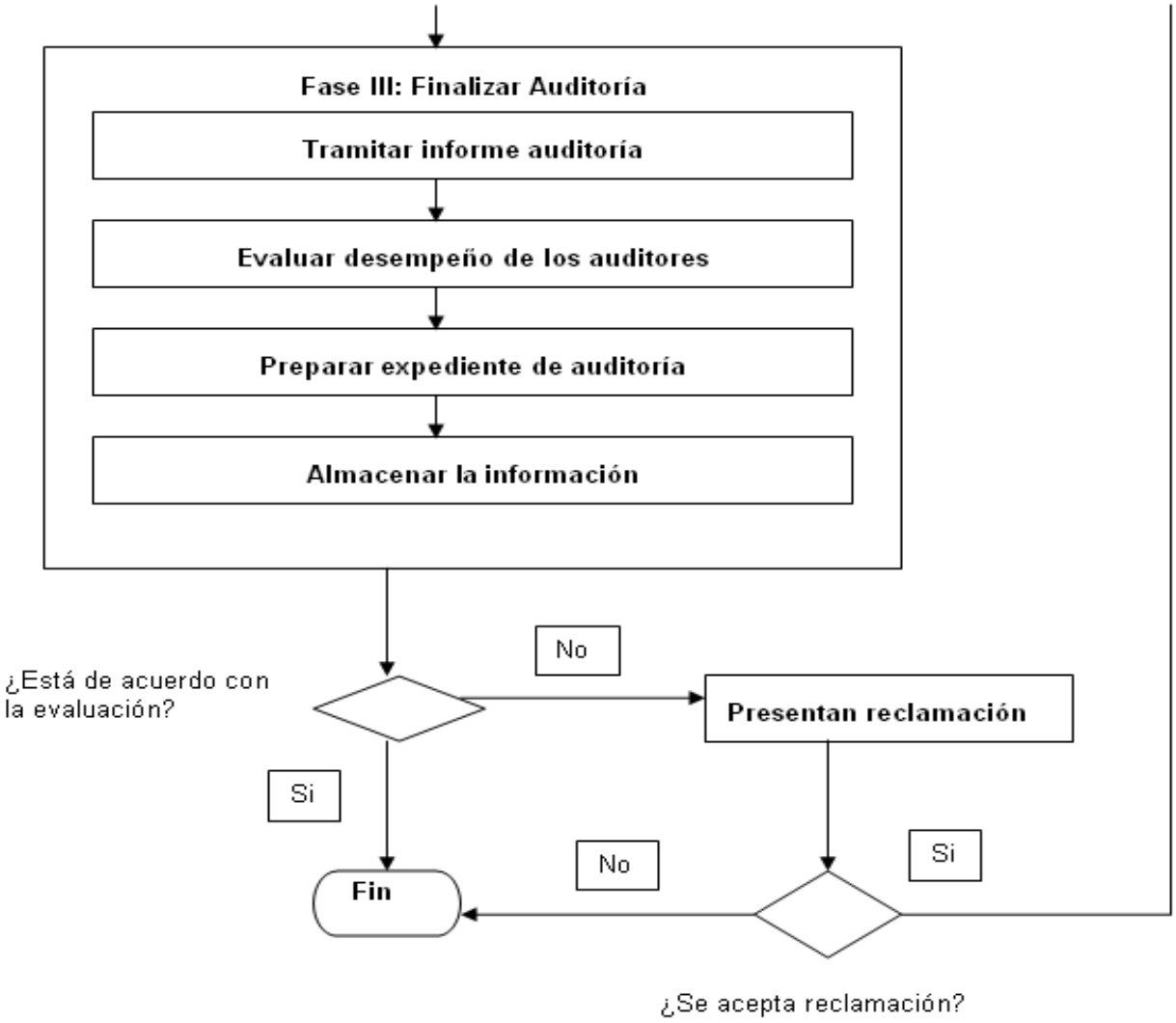
5. **Mantenibilidad**	
	Capacidad del producto software de ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software a cambios en el ambiente, así como en los requisitos y las especificaciones funcionales. Está reflejada por las siguientes sub-características:
a. Diagnósticabilidad	Capacidad del producto software de ser objeto de un diagnóstico para detectar deficiencias o causas de los fallos totales en el software, o para identificar las partes que van a ser modificadas.
b. Cambiabilidad	Capacidad del producto software para permitir la aplicación de una modificación especificada.
c. Estabilidad	Capacidad del producto software para minimizar los efectos inesperados de las modificaciones realizadas al software.
d. Comprobabilidad	Capacidad del producto software para permitir la validación de un software modificado.
e. Conformidad con la mantenibilidad	Capacidad del producto software para adherirse a las normas o convenciones que se relacionan con la mantenibilidad.
6. **Portabilidad**	
	La facilidad con que el software puede ser llevado de un entorno a otro. Se refiere a las sub-características:
a. Adaptabilidad	Capacidad del producto software de ser adaptado a los ambientes especificados sin aplicar acciones o medios de otra manera que aquellos suministrados con el propósito de que el software cumpla sus fines.
b. Instalabilidad	Capacidad del producto software de ser instalado en un ambiente especificado.
c. Coexistencia	Capacidad del producto software de coexistir con otro software independiente en un ambiente común y compartir los recursos comunes.
d. Remplazabilidad	Capacidad del producto software de ser usado en lugar de otro producto software especificado para los mismos fines y en el mismo ambiente.
e. Conformidad con la portabilidad	Capacidad del producto software de adherirse a las normas o convenciones relativas a la portabilidad.

Anexo 8

Desarrollo del Procedimiento Auditorías a la actividad productiva.




Fase II: Desarrollo de la Auditoría.
Preparar las actividades
Confirmar disponibilidad de recursos materiales
Definir los métodos y las técnicas a utilizar
Asignar las tareas al Equipo auditor
Preparar el plan de auditoría
Preparar documentos
Realizar las Actividades
Preparar y realizar reunión de apertura
Recopilar y verificar la información
Generar los Hallazgos de la auditoría
Realizar reunión de Pre cierre
Aprobar evaluación
Preparar y realizar reunión de cierre


Fase III: Finalizar Auditoría
Tramitar informe auditoría
Evaluar desempeño de los auditores
Preparar expediente de auditoría
Almacenar la información
¿Está de acuerdo con la evaluación?
No
Presentan reclamación
Si
Si
No
Fin
¿Se acepta reclamación?

Anexo 9
Método de Experto. Modelo 1.

Modelo No. 1
Guía para informar el peso de los criterios.

Fecha de recepción……

Fecha de entrega.....

Nombre y Apellidos del evaluador…….

• Le otorgará un peso a cada criterio de acuerdo a su opinión y el peso total de cada grupo debe sumar:

Grupo No.1…… 30

Grupo No.2……..15

Grupo no.3…… 30

Grupo No.4……..25

Para que el peso total asignado sea 100.

Grupo No. 1: Criterios de mérito científico

1. Valor científico.

Peso.....

2. Calidad de la investigación.

Peso.....

3. Contribución científica.

Peso.....

4. Responsabilidad científica y profesionalidad del investigador.

Peso.....

Grupo No. 2: Criterios implantación

5. Necesidad de creación.

Peso.....

6. Satisfacción de las necesidades de los ingenieros de software.

Peso.....

Grupo No.3: Criterios de generalización

7. Adaptabilidad a proyectos productivos que utilicen Arquitectura MDA.

Peso.....

8. Fácil entendimiento.

Peso.....

9. Facilidad de aplicación.

Peso.....

10. Factible interpretación de los resultados después de la aplicación.

Peso.....

Grupo No.4: Criterios de impacto

11. Aceptación.

Peso.....

12. Posibilidades de aplicación.

Peso.....

13. Organización en el proceso de aplicación.

Peso.....

14. Impacto en el área a la cuál está destinada.

Peso.....

Anexo 10
Método de Experto. Modelo 2.

Modelo No. 2

Guía para la evaluación.

Fecha de recepción……..

Fecha de entrega.....

Nombre y Apellidos del Evaluador……

- Criterios de medida que se evalúan en una escala de 1 - 5

Grupo No. 1: Criterios de mérito científico

1. Valor científico.
   Evaluación.....
2. Calidad de la investigación.
   Evaluación.....
3. Contribución científica.
   Evaluación.....
4. Responsabilidad científica y profesionalidad del investigador.
   Evaluación.....

Grupo No. 2: Criterios implantación

5. Necesidad de creación.
   Evaluación.....
6. Satisfacción de las necesidades de los ingenieros de software.
   Evaluación.....

Grupo No.3: Criterios de generalización

7. Adaptabilidad a proyectos productivos que utilicen Arquitectura MDA.
   Evaluación.....
8. Fácil entendimiento.
   Evaluación.....
9. Facilidad de aplicación.
   Evaluación.....
10. Factible interpretación de los resultados después de la aplicación.
    Evaluación.....

Grupo No.4: Criterios de impacto

11. Aceptación.
    Evaluación.....
12. Posibilidades.
    Evaluación.....
13. Organización en el proceso de aplicación.
    Evaluación.....

14. Impacto en el área a la cuál está destinada.

    Evaluación.....

- Categoría final de la propuesta

___ Excelente: Alta novedad científica, con aplicabilidad y resultados relevantes.

___ Bueno: Novedad científica, resultados destacados.

___ Aceptable: Suficientemente bueno con reservas.

___ Cuestionable: No tiene relevancia científica y los resultados son malos.

___ Malo: No aplicable.

- Valoración final

− Sugerencias del evaluador para mejorar la calidad de la propuesta.

− Elementos críticos que deben mejorarse.