Universidad de la Ciencias Informáticas

"Facultad 4"

"Propuesta de Listas de Chequeo para los Principales Artefactos Generados en el Expediente de Proyecto de Calidad"

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor(es): Osiris Pérez Lastra

Yuniel Hernández García

Tutor(es): Ing. Violena Hernández Aguilar

Co-tutor: Ing. Tayché Capote García

"Ciudad de Habana, 19 de junio del 2008
Año 50 del triunfo de la Revolución."

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

<nombre autor>

_____

Firma del Autor

<nombre autor>

_____

Firma del Autor

<nombre tutor>

_____

Firma del Tutor

Agradecimientos

A mi Tutora y Cotutora por ayudarme hasta el último momento.
A todos a aquellos que de una forma u otra persuadieron a Chucho para que su labor en la realización de esta investigación fuera más fructífera y asidua.
En especial a mi mamá y mis hermanos por brindarme apoyo incondicional; esperanzas, cuando no tenía y fuerzas, cuando me sentía debilitada.

Osiris

Hoy quiero agradecerles a todos aquellos que de uno forma u otra han hecho posible que hoy yo pueda graduarme esos que me han aguantado por más de 6 años .Quisiera mencionarlos a todos pero tendría que hacer otra tesis solo de nombre por eso solo les digo gracias a todos, mencionar aquellos que más cercanos estuvieron a mi de una forma u otra.
A mi mamá, por creer en mí y estar ahí siempre que la he necesitado.
A mi abuela, por educarme, criarme y quererme tanto.
A mi novia, por quererme tanto y apoyarme siempre dándome ánimos.
A mi compañera de tesis Osisris, que si ella no lo hubiera logrado, gracias por haberme resistido.
A Ivelis y Sama, por su ayuda en la Tesis.
A Hardys, por su ayuda y su amistad incondicional.
A Misladys, por escucharme y matarme el hambre!!!!!!.....

Yuniel.

Dedicatoria

… A mis padres y mis hermanos por tener siempre confianza en mí.

Osiris

A mi mamá y abuela por creer en mí y quererme mucho.

Yuniel.

Resumen

Hoy en día garantizar la calidad del software se ha convertido en tarea primordial para todas aquellas entidades que desarrollan sistemas informáticos. Normalmente el equipo de desarrolladores padece de una prisa patológica pues se encuentran en un estado de presión debido a la necesidad de cumplir con las fechas establecidas en el cronograma, repercutiendo esto de gran manera que el software presente errores tales como: mala elaboración de los artefactos que se originan en cada fase del ciclo de vida del software, incumplimiento de las normas metodológicas por las cuales se rige dicho sistema, el proceso de pruebas no se cumple o se ejecuta de una manera desorganizada y entre otras deficiencias.

Como una de las tantas actividades para asegurar la calidad del software el objetivo del presente trabajo se centra en brindar una herramienta sencilla, rápida y fácil de aplicar que posibilite que los principales artefactos generados en el expediente de proyecto de calidad tengan la completitud requerida y cumplan todas las normas establecidas por la metodología del Proceso Unificado del Rational.

Palabras Clave

Verificación, Validación, Técnicas de Evaluación del Software, Técnicas de Lectura, Inspecciones, Lista de Verificación.

Índice

Introducción

En nuestros días la producción del software crece de manera vertiginosa. La meta es desarrollar productos que cumplan con las necesidades del cliente y sean entregados en un tiempo establecido, respondiendo así a una correcta organización y planificación. El software se ha transformado en el pilar de avance de los sistemas y producto informáticos. Para lograr la perfección total o casi total es necesario que el sistema cuente con una calidad requerida, ¿pero qué es calidad?

Las primeras definiciones de calidad datan desde los años setenta. Muchos especialistas, ingenieros, investigadores y comercializadores de software, motivados por la preocupación de la calidad de sus aplicaciones realizaron numerosas investigaciones sobre cómo obtener un software con calidad y cómo evaluar la calidad del software.

Según (IEEE, 1990), la calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario. Se puede definir además como el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. Decir calidad es decir eficiencia, flexibilidad, corrección, confiabilidad, portabilidad, usabilidad, funcionalidad, seguridad e integridad.

La calidad del software puede medirse después de elaborado el producto pero resulta más conveniente hacer esta medición a medida que se vaya creando el software para evitar grandes costos de recursos y pérdida de tiempo.

Nuestro país a pesar de tener un desarrollo incipiente en la producción de sistemas informáticos realiza numerosas actividades y proyectos para desarrollar la industria del software, destacándose así la Universidad de las Ciencias Informáticas como centro que combina la docencia y la producción de software.

La construcción de un sistema de software tiene como objetivo primario satisfacer las necesidades solicitadas por el cliente pero ¿cómo se puede asegurar que el producto se corresponde con exactitud con lo que pidió el cliente? y ¿cómo se puede tener la certeza que el producto va a tener un correcto funcionamiento?

Desgraciadamente nuestra capacidad para medir la fiabilidad del software es muy inferior a lo que sería necesario[1]. Resultaría bastante prudente y grandioso que los informáticos como otros de sus homólogos

[1] W. Wayt Gibbs. Software's Chronic Crisis. Scientific American. Number 218.November 1994

pudiesen demostrar la corrección de sus programas matemáticamente. Otros ingenieros para predecir el comportamiento de sus creaciones recurren al análisis matemático posibilitando así la ocurrencia de errores mucho antes de que el producto esté elaborado. Resulta fatal para el vasto campo del mundo binario que estas matemáticas tradicionales, aptas para la descripción de sistemas físicos (sistemas tratados por otras ingenierías) no sean aplicables. Para este universo binario, el campo de sistemas de software, cuenta con la matemática discreta, ciencia no muy madura pues el estudio de esta materia empieza a penas con el surgimiento de los sistemas de software.

La verificación empírica[2] es el método que tienen los informáticos para preservar la confiabilidad de los programas debido a la imposibilidad de aplicar métodos matemáticos rigurosos. En la trayectoria de este proceso la seguridad de los programas incrementará. Una vez terminado el sistema a construir se pone en marcha y el mismo es observado directamente posibilitando así la eliminación de deficiencias. Sin embargo esta forma no propicia una respuesta adecuada debido a:

- ✓ De encontrar errores graves que afectan a productos anteriores dígase requisitos, diseño y entre otros, se debe volver atrás en el desarrollo, surgen entonces interrogantes como ¿qué debemos hacer?, ¿entregamos el sistema más tarde y repetimos el desarrollo?, ¿le pedimos al cliente un aumento del presupuesto?
- ✓ Por otra parte se plantea que la verificación empírica no es aplicable para asegurar la no existencia de deficiencias en el software pues existe una relación de dependencia entre una parte del software que se está comprobando y de las entradas que se le hayan hecho al código. Por lo tanto existe la probabilidad que haya errores en otra parte del programa que no se esté ejecutando en ese momento o con otras entradas que no se hayan empleado en la prueba.

Se puede afirmar con gran seguridad que el objetivo substancial de un proceso de desarrollo del software es garantizar e incrementar la calidad plena del mismo en cada una de sus fases teniendo un correcto control sobre el proceso. No importa a quien va dirigido el producto lo importante es producir lo esperado, en el tiempo esperado y con el coste esperado.

Por lo tanto es recomendable que al software se le realice una evaluación a medida que se vaya creando, construyendo. Es muy necesario y productivo que se lleve a cabo al unísono con el proceso de desarrollo del software un proceso de verificación, de comprobación de los distintos modelos o

[2] Existen 4 tipos de métodos de experimentación para la ingeniería de software: analítico, científico, ingenieril y empírico. Este último consiste en proponer un modelo como patrón y realizar validaciones una vez que se hayan realizado ciertas observaciones inductivas.

productos que se van generando en el que participarán clientes y desarrolladores.

Durante el proceso de desarrollo se distinguen dos tipos de evaluaciones que según IEEE Std 729-1983 estas se definen como, (Juristo, et al., 2006):

- ✓ Verificación: Proceso que determina que los productos de una cierta fase cumplen con los requisitos de la fase anterior.
- ✓ Validación: Proceso que evalúa el software al final del proceso de desarrollo, propiciando así que cumpla con las necesidades del cliente.

De modo general, la verificación ayuda a determinar si el producto durante el proceso de desarrollo ha sido elaborado adecuadamente mientras que la validación propicia saber si se ha elaborado el producto correcto. Se relaciona con errores al malinterpretar las necesidades del cliente.

A pesar de que se intente lograr un software con calidad actualmente existe una tendencia de no se satisfacer completamente las necesidades y se siguen evidenciando deficiencias durante el proceso de desarrollo del producto como lo muestra la imagen siguiente, (Condori- Fernández).

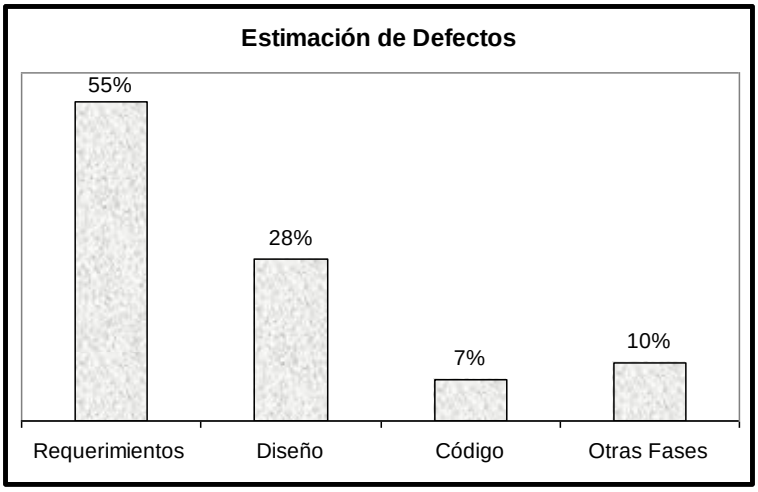

Fig.1: Estimación de Defectos por Fases.

La Universidad de las Ciencias Informáticas (UCI), no está exenta de estos problemas pues en algunos proyectos durante el ciclo de vida del software se violan las principales reglas de la metodología RUP (Racional Unified Process) determinando así la pobre y baja calidad de los proyectos, y esto trae como consecuencia que se atrasen los cronogramas de ejecución porque al entregable no tener la calidad requerida no es liberado por el laboratorio de pruebas y es necesario volver a al flujo en que se

desarrolló y corregir los defectos encontrados. A lo anterior se suma que la UCI no cuenta con un indicador o patrón estándar que posibilite comprobar que los productos creados durante el proceso de desarrollo del software cumplen con las especificaciones de la metodología de RUP.

La siguiente tabla muestra el número de revisiones a la documentación que han sido abortadas desde Enero hasta Mayo de este año, en el laboratorio de Calidad de Software de la Universidad, por existir problemas con la estructura y semántica del documento y no cumplir con los parámetros establecidos por la metodología. De un total de 14 Pruebas Fallidas en los 5 meses, 10 han estado relacionadas con la documentación.

Mes	Cantidad de Pruebas Fallidas
Enero	2
Febrero	1
Marzo	5
Abril	1
Mayo	1

Tabla 1: Cantidad de pruebas a la documentación, que se han abortado, por meses.

Ante tal situación el proceso de verificación del software se convierte en una posible herramienta que puede resultar efectiva para asegurar la calidad de un producto final entregable al cliente pues permite comprobar la completitud de los productos generados en cada fase del proceso de desarrollo.

Teniendo en cuenta tal situación surge el siguiente problema científico ¿cómo verificar que los artefactos (modelos, requisitos y entre otros) están bien elaborados y cumplen con las especificaciones de la metodología RUP?

El objeto de estudio de esta investigación lo constituyen los procesos, metodologías de desarrollo de software y sus artefactos, y el campo de acción se enmarca específicamente en el aseguramiento de calidad a la documentación del software

El objetivo general de esta investigación es: Elaborar listas de chequeos para comprobar la calidad de los principales artefactos ingenieriles que se genera en el expediente de proyecto. Estos artefactos son:

1. Modelo de Dominio
2. Especificación de Requerimientos.

3. Especificación de CU.
4. Documento de Análisis.
5. Documento de Diseño.
6. Documento de Arquitectura del Software.
7. Documento de Arquitectura de Información
8. Diccionario de Datos.
9. Diagrama de Despliegue.
10. Modelo Entidad Relación.
11. Modelo de Objeto.
12. Manual de Usuario.
13. Manual de Instalación.
14. Aplicación Web.
15. Glosario de Términos

Para lograr la presente investigación es necesario llevar a cabo las siguientes tareas de investigación:

1. Analizar las características que deben cumplir los artefactos del expediente de proyecto en la universidad.
2. Identificar la calidad de los artefactos en los proyectos de la Universidad.
3. Identificar los elementos que debe incluir una lista de chequeo por artefactos.
4. Elaborar listas de chequeo para verificar la calidad de los artefactos.
5. Aplicar las listas de chequeo en el laboratorio de calidad de la UCI.

Para lograr una mejor organización del contenido de la tesis, la misma se estructuró por capítulos:

Capítulo 1: Fundamentación Teórica: En este capítulo se abordan las características de las diferentes metodologías, modelos y procesos de desarrollo de software, se resalta la importancia de la calidad durante todo el desarrollo, especificando las distintas técnicas utilizadas para verificar y evaluar la calidad, haciendo énfasis en las listas de chequeo.

Capítulo 2: Propuesta de Listas de Chequeo: En este capítulo se realiza un análisis de la calidad de la

producción en la universidad, se explica en qué consiste el expediente de proyecto y se describen los principales artefactos según la metodología RUP de manera precisa y breve. Se describe la propuesta de las diferentes listas de chequeo elaboradas, especificándose a que artefactos van dirigidos y se establece la forma de evaluar la calidad de los mismos.

Capítulo 3: Validación de la Propuesta de Listas de Chequeo: En este capítulo se describe como validar las listas mediante el criterio de experto y los resultados obtenidos con la aplicación de las mismas en el Laboratorio de Calidad de la Universidad de las Ciencias Informáticas.

Capítulo 1: Fundamentación Teórica

Introducción

El interés por la calidad es un elemento que crece continuamente. Para lograr un producto con calidad es necesario llevar a cabo un buen proceso de desarrollo del software, en este aspecto es importante destacar que el producto a entregar debe considerar la calidad en todos sus estados de evolución a medida que avance el desarrollo de acuerdo al ciclo de vida seleccionado para su construcción (requerimientos, análisis, diseño, entre otros). Existen diferentes procesos de desarrollo por lo que a la hora de realizar un software se debe realizar un estudio para identificar el más adecuado en dependencia de las características del proyecto y es recomendable llevar a cabo un proceso de evaluación determinado por el empleo de técnicas de evaluación estática y dinámica respectivamente.

Es este capítulo se describirán las distintas técnicas de evaluación de software y se abordará además los procesos de desarrollo, metodologías y modelos de desarrollo de software.

1.1 Procesos de Desarrollo de Software

El objetivo fundamental del proceso de desarrollo del software es la producción eficiente y eficaz de un software que reúna los requisitos que el cliente desee. Este proceso se caracteriza por ser completamente intelectual, afectado por la creatividad y el juicio de las personas involucradas (Sommerville, 2002). Este proceso es comparable en muchos aspectos a cualquier otro proceso de ingeniería, pero tiene asociado una serie de desafíos adicionales, relativos esencialmente a la naturaleza del producto obtenido. A continuación se explican algunas particularidades asociadas al desarrollo de software y que influyen en su proceso de construcción.

Un producto software en sí es complejo, es prácticamente inviable conseguir un 100% de confiabilidad de un programa por pequeño que sea. Existe una inmensa combinación de factores que impiden una verificación exhaustiva de las todas posibles situaciones de ejecución que se puedan presentar (entradas, valores de variables, datos almacenados, software del sistema, otras aplicaciones que intervienen, el hardware sobre el cual se ejecuta, entre otros).

Un producto software es intangible y por lo general muy abstracto, esto dificulta la definición del producto y sus requisitos, sobre todo cuando no se tiene precedentes en productos de software similares. Esto hace que los requisitos sean difíciles de consolidar tempranamente, además los cambios en los requisitos son inevitables, no sólo después de entregado el producto sino también durante el

proceso de desarrollo.

El proceso de desarrollo de software no es único. No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo. Debido a esta diversidad, es difícil automatizar todo un proceso de desarrollo de software.

A pesar de la variedad de propuestas de proceso de software, existe un conjunto de actividades fundamentales que se encuentran presentes en todos ellos (Sommerville, 2002):

Especificación de software: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.

Diseño e Implementación: Se diseña y construye el software de acuerdo a la especificación.

Prueba: El software debe ser probado para detectar errores.

Validación: El software debe validarse, para asegurar que cumpla con lo que quiere el cliente.

Evolución: El software debe evolucionar, para adaptarse a las necesidades del cliente.

Además de estas actividades fundamentales, (Pressman, 2000) menciona un conjunto de "actividades protectoras", que se aplican a lo largo de todo el proceso del software. Ellas se señalan a continuación:

1. Seguimiento y control de proyecto de software.
2. Revisiones técnicas formales.
3. Garantía de calidad del software.
4. Gestión de configuración del software.
5. Preparación y producción de documentos.
6. Gestión de reutilización.
7. Mediciones.
8. Gestión de riesgos.

1.2 Modelos de Proceso Software

(Sommerville, 2002) define modelo de proceso de software como "Una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software es una abstracción de un proceso real."

Los modelos genéricos no son descripciones definitivas de procesos de software; sin embargo, son

abstracciones útiles que pueden ser utilizadas para explicar diferentes enfoques del desarrollo de software.

Modelos existentes:

- Codificar y corregir
- Modelo en cascada
- Desarrollo evolutivo
- Desarrollo formal de sistemas
- Desarrollo basado en reutilización
- Desarrollo incremental
- Desarrollo en espiral

1.2.1 ¿Cuál es el modelo de proceso más adecuado?

Cada proyecto de software requiere de una forma particular de abordar el problema. Las propuestas comerciales y académicas actuales promueven procesos iterativos, donde en cada iteración puede utilizarse uno u otro modelo de proceso, considerando un conjunto de criterios (Por ejemplo: grado de definición de requisitos, tamaño del proyecto, riesgos identificados, entre otros).

1.3 Metodologías para Desarrollo de Software

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, entre otros). Adicionalmente una metodología define con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo y demás. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y/o diseño.

La comparación y/o clasificación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas. A grandes rasgos, si tomamos como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, podemos clasificar las metodologías en dos grupos: Metodologías Estructuradas y Metodologías Orientadas a Objetos. Por otra parte, considerando su filosofía de desarrollo, aquellas metodologías con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías

Tradicionales (o peyorativamente denominada Metodologías Pesadas, o Peso Pesado). Otras metodologías, denominadas Metodologías Ágiles, están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

1.3.1 Metodologías más Empleadas

Cada día aumenta en número y variedad los procesos de desarrollo de software. En el mundo entre las metodologías ágiles más empleadas se encuentran FDD (Feature Driven Development \Desarrollo Guiado por Funcionalidad) y XP (eXtreme Programming\Programación eXtrema). El proceso que goza de mayor popularidad entre sus usuarios es RUP (Racional Unified Process\ Proceso Unificado de Rational) a pesar de ser un método “pesado”.

En la UCI existe cierta variedad en el empleo de metodologías, pues como bien se citó anteriormente el establecimiento de un proceso debe ajustarse a las necesidades de cada desarrollador, a las características de la entidad y en ocasiones a las del proyecto.

La Infraestructura Productiva (IP) realizó una encuesta en el presente año 2008 a los desarrolladores y principales roles de cada proyecto y la misma arrojó como la más empleada RUP, como bien los evidencian las figuras 2 y 3.

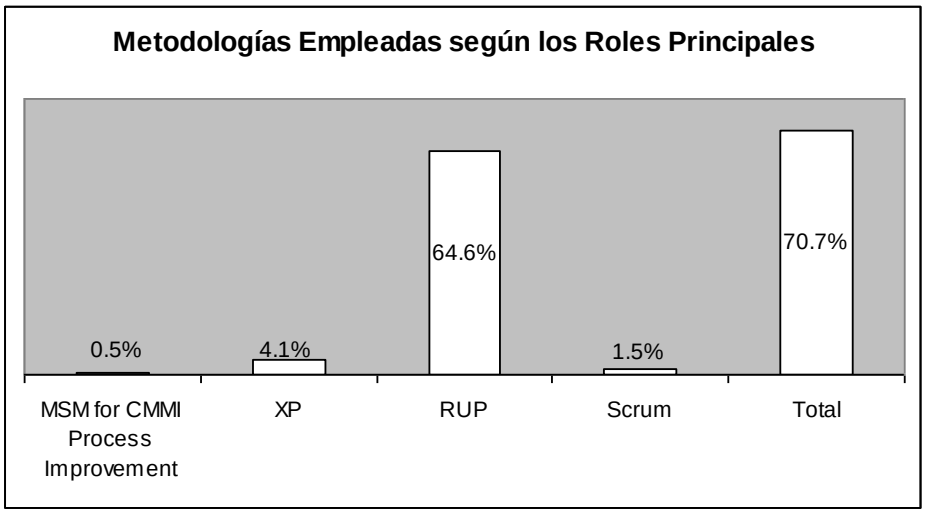

Fig.2: Metodologías de desarrollo utilizadas en la Universidad según el criterio de los diferentes roles.

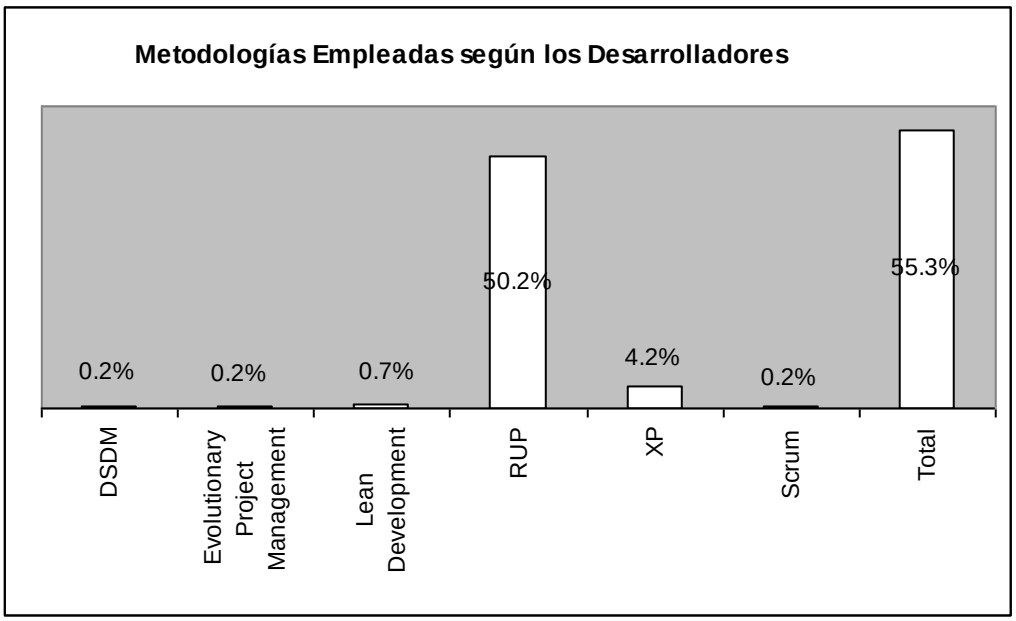

Fig.3: Metodologías de desarrollo utilizadas en la Universidad según el criterio de los desarrolladores.

En la siguiente sección se define de manera breve las metodologías más empleadas en la Universidad, haciendo énfasis en RUP.

1.3.2 Metodologías Empleadas en la Universidad

RUP (Rational Unified Process \Proceso Unificado del Rational)

Es uno de los procesos más generales y grandes de los existentes pues está desarrollado en adaptarse a cualquier proyecto.

En su modelación se definen como elementos, (WEB01):

- Trabajadores (quién): Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- Actividades (cómo): Tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Artefactos (qué): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

- Flujo de Actividades (cuándo): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

Fases y Flujos de Trabajo.

El proyecto que sigue esta metodología se divide en cuatro fases y en cada una de ellas se pueden ejecutar una o varias iteraciones, cada fase tiene asociada varios flujos de trabajo como se muestra en la figura:

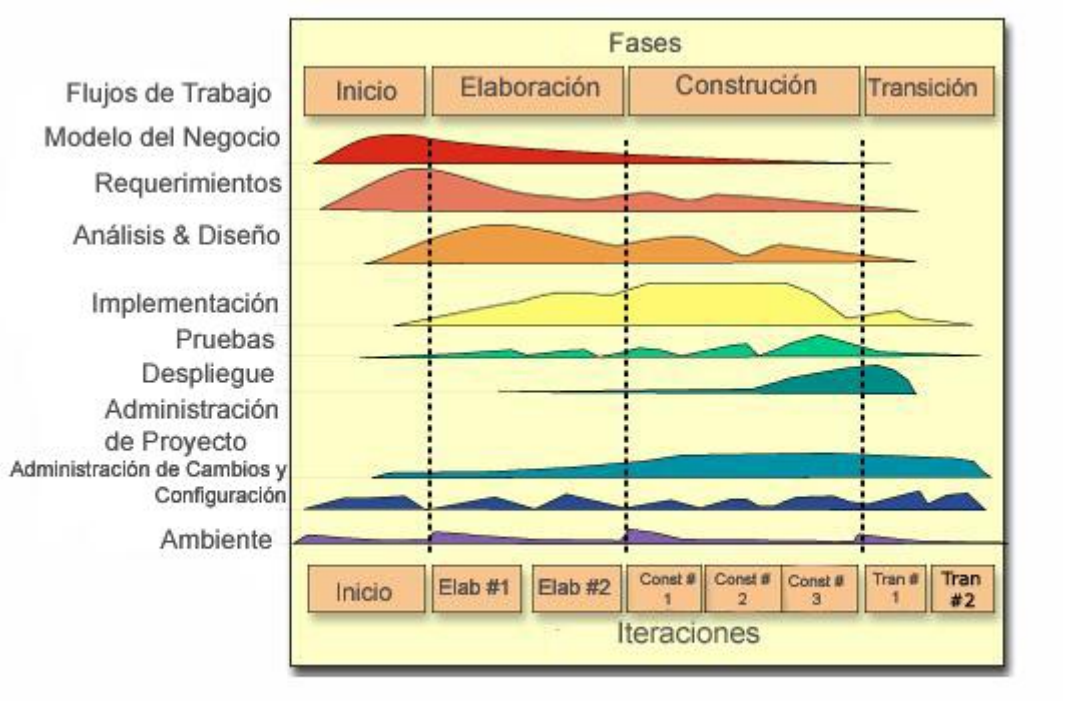

Fig.4: Los Flujos de trabajos y fases de RUP.

Fases

- ✓ **Conceptualización:** (Concepción o Inicio): Describe el negocio y delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- ✓ **Elaboración:** Define la arquitectura del sistema.
- ✓ **Construcción:** Obtiene un producto documentado, con manual de usuario que está listo para su utilización. Se pone a consideración de los usuarios para que lo prueben.
- ✓ **Transición:** El producto ya probado por los usuarios está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Flujos de Trabajo

Existen 6 flujos ingenieriles y 3 de apoyo, los ingenieriles se realizan en un determinado momento en el desarrollo y los de apoyo se llevan a cabo durante todo el proyecto.

Flujos Ingenieriles

- ✓ **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- ✓ **Requerimientos:** Define lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- ✓ **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- ✓ **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- ✓ **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.
- ✓ **Despliegue**: Produce un release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, entre otros) para entregar el software a los usuarios finales.

Flujos de apoyo

- ✓ **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- ✓ **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, entre otros.
- ✓ **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Principales artefactos de los flujos de ingeniería

Modelo del Negocio:

- ✓ Modelo de Casos de Uso
- ✓ Modelo de Objeto del Negocio
- ✓ Glosario de Términos
- ✓ Especificación Complementaria

Requerimientos:

- ✓ Modelo de Casos de Uso
- ✓ Casos de Uso
- ✓ Glosario de Términos
- ✓ Prototipo de Interfaz de Usuario

Implementación:

- ✓ Modelo de Implementación
- ✓ Elementos de Implementación
- ✓ Subsistemas de Implementación

Análisis

- ✓ Documento de Arquitectura
- ✓ Modelo de Análisis
- ✓ Clases de Análisis
- ✓ Realización de los Casos de Uso

Diseño:

- ✓ Modelo de Despliegue
- ✓ Documento de Arquitectura
- ✓ Clases de Diseño
- ✓ Realización de los Casos de Uso
- ✓ Subsistema de Diseño
- ✓ Paquete de Diseño
- ✓ Modelo de Datos

Prueba:

- ✓ Plan de Prueba
- ✓ Caso de Prueba
- ✓ Documento de No Conformidades

Despliegue

- ✓ Manual de Usuario
- ✓ Manual de Instalación

El ciclo de vida de RUP se caracteriza por:

1. Dirigido por Casos de Uso: Enfoca la necesidad de los futuros clientes.
2. Centrado en la Arquitectura: La arquitectura muestra una visión completa del sistema.
3. Iterativo e Incremental: En cada fase se ejecutan iteraciones que involucra los flujos de trabajo generando así productos que se irán incrementado.

XP (eXtreme Programming\ Programación eXtrema)

Enfatiza más la adaptabilidad que la previsibilidad. Intenta reducir la complejidad del software mediante un trabajo orientado únicamente al objetivo, apoyado en las relaciones interpersonales y la velocidad de reacción. Fue creada previendo las siguientes circunstancias:

- ✓ Altas probabilidad de cambios en los requisitos del proyecto. Un ejemplo de esto es que el cliente no sabe exactamente lo que desea o porque el cambio de requisito esta unido al dominio del problema a resolver.
- ✓ Proyectos con altos riesgos. Esto se manifiesta cuando existe un proyecto muy necesario, muy imprescindible, donde su fecha de entrega no se debe dejar de cumplirse bajo ningún motivo.
- ✓ Proyectos con pocos programadores

FDD (Feature Driven Development\ Desarrollo Guiado por Funcionalidad)

Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad, (Molpeceres, 2002).

LD (Lean Development)

Definida por Bob Charette.s a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa. En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios, (Molpeceres, 2002).

DSDM (Dynamic Systems Development Method)

Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación, (Molpeceres, 2002).

Las tres últimas son iterativas, además de existir realimentación a todas las fases.

SCRUM

Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración, (Molpeceres, 2002).

Todas las metodologías ya sean ágiles o robustas definen un grupo de actividades que contribuyen a garantizar la calidad del software y su documentación asociada, debido a la importancia de esta actividad para garantizar el éxito del proyecto.

1.4 Calidad en el Proceso de Desarrollo del Software

"La Calidad no se controla se fabrica"

Anónimo

El término de calidad ha estado evolucionado con el pasar del tiempo. Es el deseo profundo para todas aquellas personas relacionadas con la producción de software. Muchas personas tienen su propia

definición por lo que hay autores que definen calidad como un vocablo subjetivo.

Con respecto a la definición de la calidad del software existe la siguiente interrogante: ¿Es realmente posible encontrar un conjunto de propiedades en un producto software que nos den una indicación de su calidad? Para dar respuesta a estas preguntas aparecen los Modelos de Calidad. En los Modelos de Calidad, la misma se define de forma jerárquica. Resuelven la complejidad mediante la descomposición. La calidad es un concepto que se deriva de un conjunto de sub-conceptos (Anexo 1), (Juristo, et al., 2006).

1.4.1 Tipos de Calidad

En los sistemas de software se puede diferenciar dos tipos de calidad, calidad de diseño y calidad de conformación. Calidad de diseño consiste en las características, procedimientos, especificaciones, entre otros, que prometen producir un servicio vendible que el cliente requiere, calidad de servicio esperada, implica el servicio más útil mientras que la calidad de conformación se cataloga como la medida de la eficiencia en el logro de los resultados por la calidad esperada, prometida. Calidad de servicio resultante implica mayor confiabilidad de trabajo ((DCCAI), 1999).

1.4.2 Control de la Calidad

El control de la calidad está regido por una serie de técnicas dinámicas y estáticas donde estas incluyen métodos para la evaluación del software. El control de la calidad encierra un bucle de realimentación (feedback) del proceso que creó el producto. La combinación de medición y realimentación permite afinar el proceso cuando los productos de trabajo creados fallan al cumplir sus especificaciones. Este enfoque ve el control de calidad como parte del proceso de fabricación (Pressman, 2005).

La detección y corrección de errores que ocurren durante el proceso de desarrollo de un producto constituyen el elemento primordial del control de calidad pues brinda la posibilidad de garantizar que el producto final cumpla con las expectativas del cliente.

1.4.3 Garantía de la Calidad del Software (Software Quality Assurance/Gestión de Calidad de Software) (SQA/GSC).

La Garantía de Calidad del Software (GSC) como algunos autores acostumbran llamarle se define como una actividad que posibilita asegurar, proteger todo el proceso de ingeniería de software.
Es un conjunto de procedimientos, técnicas y herramientas, aplicados por profesionales, durante el ciclo de desarrollo de un producto, para asegurar que el producto satisface o excede los estándares o niveles de calidad preestablecidos (IEEE, 1990).

Grupo de actividades de planificación, estimación y supervisión de las actividades de desarrollo, que se realizan de forma independiente al equipo de desarrollo, de tal forma que los productos software resultantes cumplen los requisitos establecidos (Company, 2006).

Es la guía de los preceptos, de gestión y de las disciplinas de diseño para el espacio tecnológico y la aplicación de la ingeniería del software (IEEE, 2004).

Según (Pressman, 2005) es un patrón de acciones planificado y sistemático que se requiere para asegurar la calidad del software.

Actividades para el aseguramiento de calidad del software:

- ✓ Métodos y herramientas de análisis, diseño, codificación y prueba.
- ✓ Revisiones Técnicas Formales que se aplican en cada paso.
- ✓ Estrategia de Prueba.
- ✓ Control de la documentación del software y de los cambios realizados.
- ✓ Procedimiento que aseguren un ajuste a los estándares de desarrollo.
- ✓ Mecanismo y medidas de Información.

En resumen, la GCS son todas aquellas actividades que aseguran la calidad del producto durante su ciclo de vida, donde este conjunto de actividades incluyen técnicas o métodos, procedimientos, herramientas para lograr efectividad, donde además existe estrecha relación con los requisitos, con los estándares de desarrollo y los atributos implícitos que se espera del software.

1.5 Evaluación del Software y el Proceso de Desarrollo

A medida que se va desarrollando el software es necesario llevar a cabo un proceso de comprobación. Anteriormente se abordaba que en la trayectoria de su evolución se destaca dos tipos de evaluación. Según

(ISO, 1994).

Verificación: El proceso de evaluar el producto de una fase dada, para asegurar la corrección y la consistencia con respecto a los productos, así como normas proporcionadas como elementos de entrada a esa fase.

(ISO, 1994)

Validación: El proceso de evaluar el software para asegurar el cumplimiento con los requisitos

especificados.

Estos métodos de evaluación agrupan dos tipos de técnicas, técnicas de evaluación estática y técnicas de evaluación dinámica. La primera consiste en buscar deficiencias sobre el sistema en reposo, o sea, estudia los diferentes modelos que componen el sistema, escudriñando posibles faltas en los mismos. Esta técnica se caracteriza además por la detección de fallas, la corrección es mucho más directa. Las técnicas de evaluación dinámica generan entradas al sistema posibilitando la detección de fallos cuando son ejecutadas dichas entradas. Este tipo de técnica es también conocida como pruebas de software o testing y son aplicadas generalmente sobre el código.

Es bueno destacar que el proceso de evaluación además de detectar deficiencias conlleva a la refinación de los mismos, corregir las fallas encontradas.

A medida que se va desarrollando los artefactos en el proceso de desarrollo se empieza a poner en práctica la técnica de evaluación estática, también conocida como revisiones, una vez que finaliza la elaboración del código fuente la técnica que se emplea es la de evaluación dinámica como bien lo muestra la imagen posterior. Después de la fase de implementación del código se pasa a la realización de pruebas unitarias y antes de efectuar estas pruebas se debe aplicar la evaluación estática en los programas del igual modo que se ha hecho con los artefactos.

En síntesis, las actividades de evaluación estática constituyen elementos cruciales para comprobar la calidad de los artefactos y el progreso del proyecto.

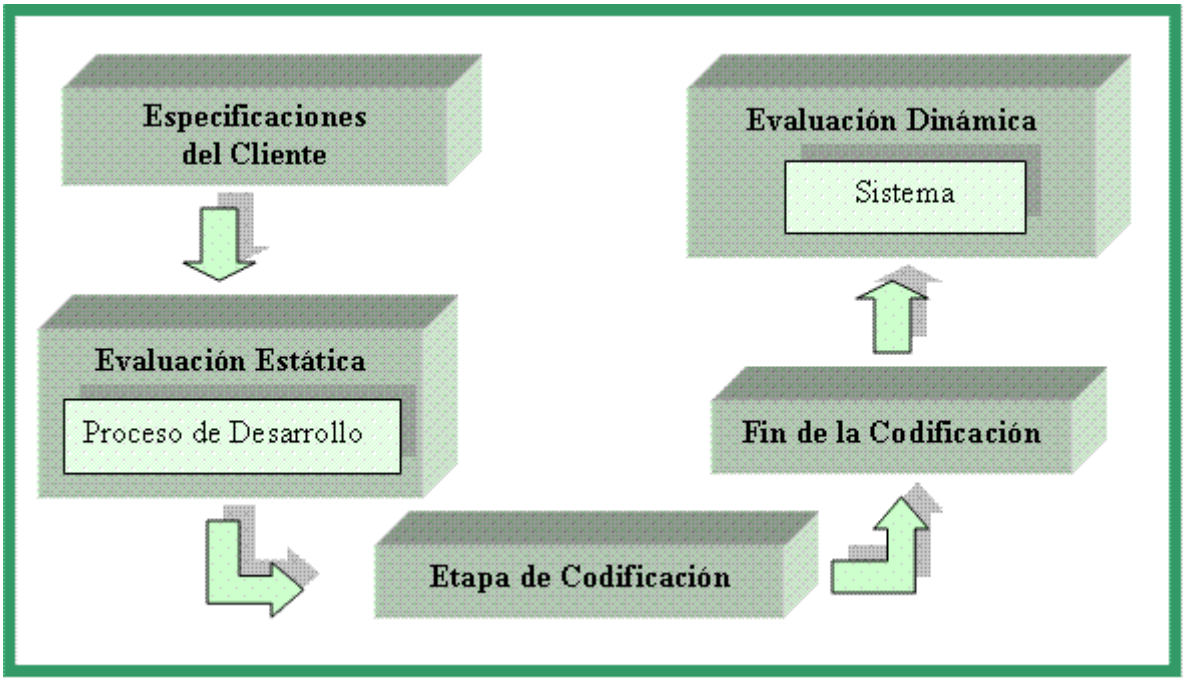

Fig.5: Abstracción de la Técnica de Evaluación Estática y Dinámica.

1.6 Técnica de Evaluación Estática

Las técnicas de evaluación estática son conocidas generalmente como Revisiones. Las revisiones pretenden divisar manualmente deficiencias en cualquier producto o artefacto del desarrollo, o sea, que un determinado artefacto está impreso en un papel y los revisores están examinando ese artefacto a través de la lectura sin ejecutarlo.

La siguiente imagen muestra en síntesis, las diferentes técnicas que se pueden llevar a cabo durante la evaluación estática:

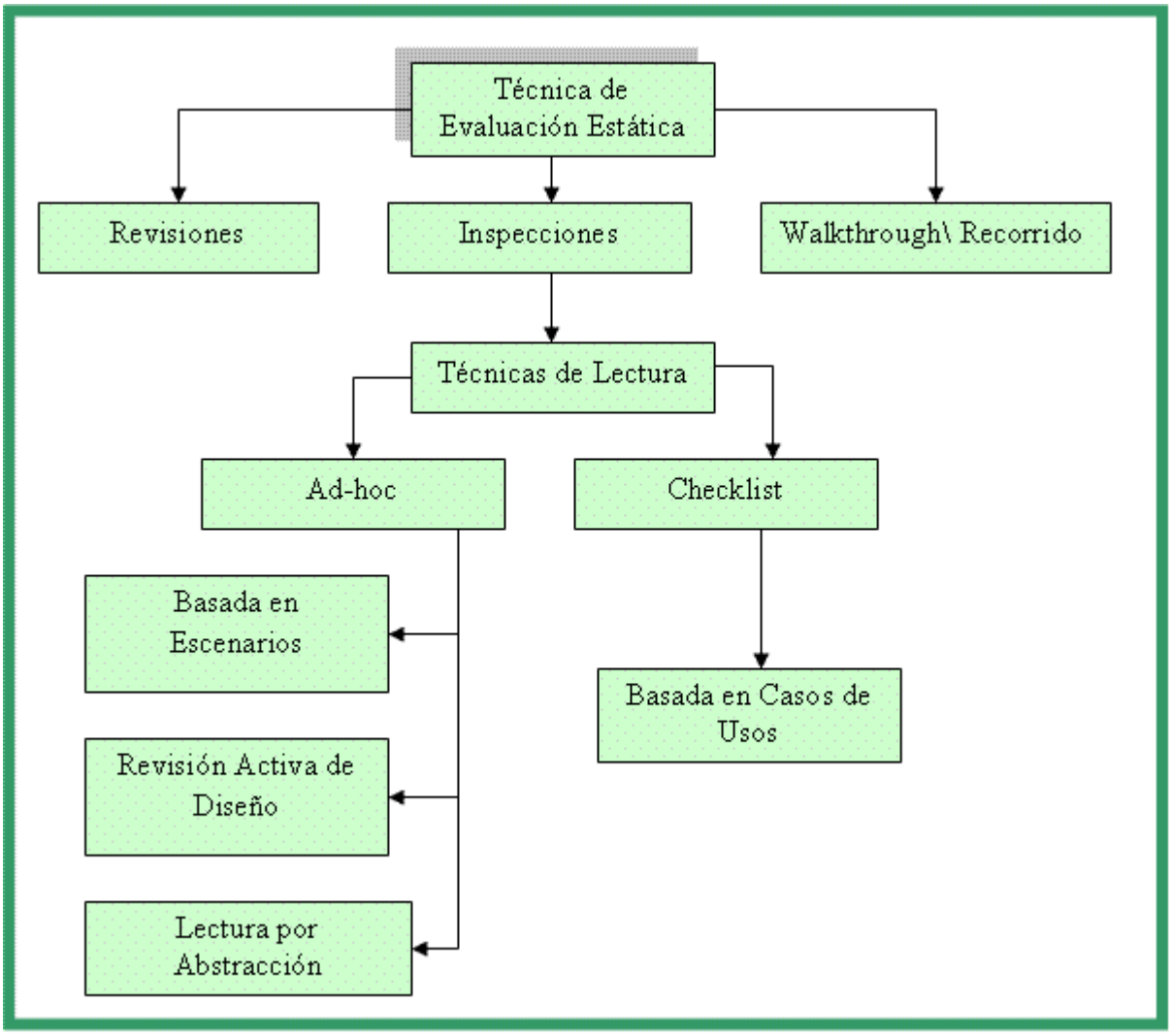

Fig.6: Clasificación de la Técnica de Evaluación Estática.

Existen diferentes tipos de técnicas de evaluación estática, las cuales se puede nombrar como:

- ✓ Revisiones Informales: Son llamadas informalmente Revisiones. Se caracteriza por intercambiar opinión entre los participantes. No emplean técnicas de lectura.
- ✓ Revisiones Formales o Inspecciones: Los participantes son los responsables de la fiabilidad de la evaluación, generando además un informe que muestra tal revisión.
- ✓ Walkthrough\Recorrido: Consiste en simular la ejecución de los casos de pruebas para el programa que se está evaluando. Con este tipo de revisión se recorre el programa imitando lo que haría una computadora. Es conocido como recorrido.

Las Revisiones Formales o inspecciones se destacan como el tipo de técnica de más importancia pues sus participantes deben estar aptos para analizar el producto, empleando algún tipo de técnica de lectura con el objetivo de detectar deficiencias.

1.6.1 Inspecciones

Las inspecciones se caracterizan como un método de análisis estático para verificar y validar un producto manualmente. Es un proceso bien definido y disciplinado. Su propósito principal es hallar deficiencias antes de que se comience la fase de prueba. Emplea algún tipo de técnica de lectura.

El proceso de inspección se puede delimitar en cuatro fases:

- ✓ Inicio: Su objetivo es preparar la inspección y proporcionar la información necesaria sobre el artefacto en cuestión.
- ✓ Detección de Defectos: Cada integrante del equipo debe realizar individuamente la lectura del material, debe comprender el artefacto y detectar fallas en el mismo. Las técnicas de lectura resultan un elemento clave en este paso.
- ✓ Colección de Defectos: Las faltas encontradas por cada integrante del equipo es llevada a un solo documento con el propósito de ser empleado como base para una discusión en grupo
- ✓ Corrección y Seguimiento: El desarrollador del artefacto que se está evaluando tiene la obligación de rectificar las deficiencias encontradas e informar de las correcciones hechas a modo de seguimiento.

Durante el proceso de inspección las personas involucradas deben tener un alto grado de experiencia y conocimiento. El candidato apropiado para desempeñar el rol de inspector debe ser aquella persona implicada en el desarrollo del producto. El equipo de inspección no debería sobrepasarse de cinco miembros. La siguiente imagen muestra en síntesis el rol que jugaría cada miembro del proyecto durante el proceso de inspección:

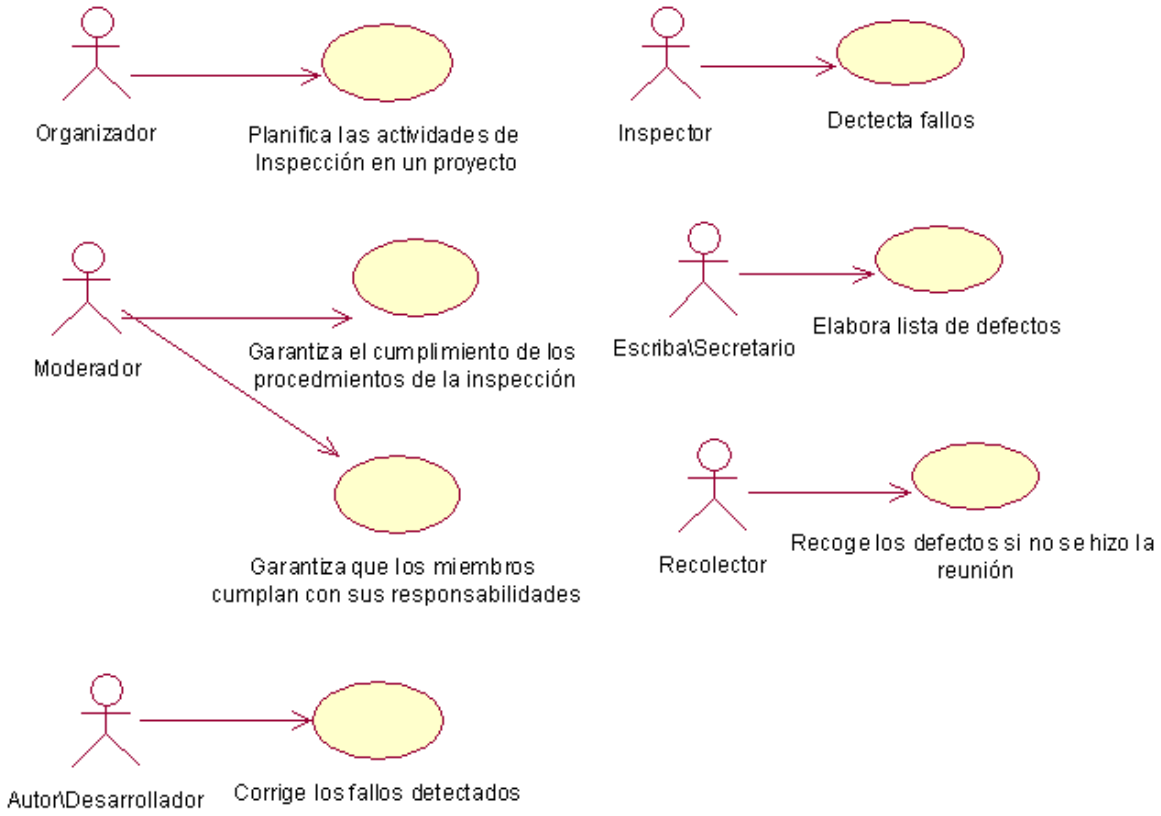

Fig.7: Roles en el proceso de una inspección.

Técnicas de Lectura

Las técnicas de lectura son "supervisoras" que facilitan detectar defectos en los productos de software elaborados.

Una técnica de lectura se define como una serie de pasos o procedimientos con el propósito de brindarle un vasto conocimiento a la persona, o sea, al inspector que la realiza. Típicamente, para la realización de la inspección es necesario como precondición tener conocimiento del producto que se está inspeccionando para lograr efectividad en la detección de defectos ya sean sencillos o complejos. . En cierto sentido, una técnica de lectura puede verse como un mecanismo para que los inspectores detecten defectos en el producto inspeccionado.

Existen diferentes tipos de técnicas de lectura. Entre las más conocidas se destacan la lectura Ad-hoc,

la lectura basada en Listas de Comprobación o Listas de Chequeo y Lectura Basada en Casos de Usos. Ambas técnicas son aplicables sobre cualquier artefacto de software.

Técnicas más populares.

Lectura Ad-hoc

Es conocida como Lectura sin Checklist. En esta técnica el producto software se entrega a los inspectores sin ninguna indicación o guía sobre cómo proceder con el producto, sin embargo, aunque los participantes no cuenten con guías de qué buscar no significa que no escudriñen sistemáticamente el producto inspeccionado, ni tampoco que no tengan en mente el tipo de defecto que están buscando. Típicamente, el inspector deberá buscar secuencialmente los defectos típicos del producto que esté leyendo Por ejemplo, si se está inspeccionando unos requisitos, el inspector, buscará sistemática y secuencialmente defectos de corrección, de completitud, de ambigüedad, etc.

Lectura basada en Lista de Comprobación

Proporciona un apoyo mayor mediante preguntas que los inspectores deben de responder mientras leen el artefacto. Es decir, esta técnica proporciona listas que ayudan al inspector a saber qué tipo de faltas buscar. Entre sus ventajas se encuentran:

- ✓ Las listas de chequeo que son desarrolladas para una actividad de control específica, y que además son usadas correctamente en la ejecución de la misma permite:
  - ✓ Una mejor planificación.
  - ✓ Asegura una consistente aproximación de la actividad, respecto al alcance definido.
  - ✓ Actúa como una guía o plan de ejecución, incluso del tiempo establecido.
- ✓ Constituyen un repositorio de información que puede ser usada posteriormente.

Lectura basada en Casos de Usos

Está técnica es utilizada en la actualidad en los desarrollos orientados a objetos. Su objetivo es poder comprobar que cada objeto puede responder de buena manera a todo posible camino en donde pueda ser usado. Es por eso que es importante realizar un análisis de los casos de uso en que el objeto participa, y así poder verificar: que se llama a los métodos correctos y que los cambios de estado dentro de los métodos son los correctos. El accionar básico de la técnica es definir un grupo de escenarios y poder ver durante la inspección como los objetos son tratados durante los escenarios. El inspector se centra o focaliza en esos escenarios por lo que su contexto de inspección es acotado. Está técnica puede ser apoyada con otras técnicas como la Técnica de Listas de Comprobación.

1.7 Técnica de Evaluación Dinámica

Son conocidas como Prueba de Software o Testing. Se caracteriza por generar entradas al sistema y este ejecuta dichas entradas con el propósito de encontrar fallas.

La prueba de Software se define como un proceso donde el sistema o parte del sistema ejecuta entradas previamente especificadas (configuración de prueba), registrándose los resultados obtenidos. Luego se realiza una evaluación que consiste en comparar los resultados obtenidos con los esperados con el objetivo de detectar fallas dando origen a la actividad de depuración en el que es necesario identificar la falta asociada con cada fallo y corregirla, que puede dar lugar a una nueva prueba. El resultado final puede brindar cierto grado de confianza en el software ya probado

El objetivo fundamental del proceso de prueba es la detección de fallas y su contrapartida es brindarle a la organización toda la información de los defectos detectados relacionados con los requerimientos del sistema.

Las pruebas no mejoran directamente la calidad del sistema, pero sí pueden prever un ámbito de errores y secuelas que pueden producir a la organización. Las pruebas brindan la posibilidad a la organización de tomar decisiones sobre la asignación de recursos para mejorar la calidad del sistema.

Para obtener un mayor nivel de detección de errores las pruebas deben realizarlas un equipo de desarrolladores distinto al que realizó el sistema pues inconscientemente los realizadores del sistema probarán que el software funciona y no que no lo hace dando lugar al poco éxito de la prueba.

1.7.1 Características de una buena prueba y actividades a realizar para probar el sistema

Para la realizar una buena prueba es necesario tener en cuenta los siguientes elementos:

- ✓ El responsable de esta actividad debe comprender el sistema e imaginarse cómo podría fallar.
- ✓ Se debe probar que el sistema hace lo que debe hacer.
- ✓ Se debe probar que el sistema no hace lo que no debe hacer.
- ✓ Como se cuenta con poco tiempo y recursos las pruebas deben caracterizarse por no ser redundantes y tener propósitos diferentes.
- ✓ Se debe probar con aquella prueba que tenga la mayor probabilidad de encontrar una clase completa de fallas.
- ✓ Para no cubrir errores se debe evitar combinar pruebas a la misma vez. Las pruebas no deben ser ni muy sencillas ni muy complejas.

Cuando se está realizando el proceso de pruebas se llevan a cabo distintas tareas. Una vez realizadas las mismas se comienza a depurar las deficiencias relacionadas con las fallas encontradas. A continuación se muestra las actividades para probar el sistema:

- ✓ Diseño de las pruebas: Se identifican los métodos o técnicas que se emplearán para probar el software. Distintas técnicas de prueba ejercitan diferentes criterios como guía para realizar las pruebas.
- ✓ Generación de los casos de prueba: Los casos de prueba se definen como las entradas de datos para ejecutar el sistema. Durante esta actividad en dependencia de la técnica que se emplee se elaborará los casos de prueba.
- ✓ Definición de los procedimientos de la prueba: Esta actividad agrupa un "cómo, cuándo, quién", o sea ¿cómo se hará el proceso de prueba, quién lo realizará y cuándo lo hará? Se define como la especificación del proceso de prueba.
- ✓ Ejecución de la prueba: Probar el sistema a partir de los casos de prueba generados previamente e identificar los fallos producidos.
- ✓ Realización de un informe de prueba: Registrar lo ocurrido durante el proceso de prueba ¿Qué fallos se encontraron, cuáles casos de prueba pasaron satisfactoriamente, cuáles no?

Para llevar a cabo las pruebas es necesario tener claro conceptos como: Estrategia, Niveles, Tipos y Métodos de pruebas.

1.8 Listas de Verificación. Herramienta Básica para el Aseguramiento de la Calidad

Para asegurar la calidad existen disímiles métodos, técnicas y herramientas. Las listas de chequeo son contempladas como una de las siete herramientas básicas de calidad, las otras seis son: histograma, diagrama de Pareto, diagrama de causa efecto, estratificación, diagrama de scadter o dispersión y gráfica de control. En la práctica, las listas de chequeo requieren ser integradas a otras técnicas como las entrevistas, las encuestas y entre otras. Correctamente empleadas pueden garantizar el objetivo deseado.

Las herramientas básicas han sido adoptadas en las actividades de mejora de la calidad y utilizadas como soporte para el análisis y solución de problemas en los distintos ámbitos de una entidad.

Muchas personas tienden a desdeñar dichas herramientas pues aparentan ser simples y fáciles de aplicar y prefieren las herramientas o técnicas más avanzadas, pero según la experiencia de algunos especialistas con estas herramientas básicas, señalan que bien aplicadas y utilizando un método

estandarizado de solución de problemas pueden ser capaces de resolver hasta el 95% de los problemas.

Las listas de chequeo hacen su aparición en el año 1970 aunque algunos autores consideran que su uso es más antiguo. En distintas fuentes bibliográficas son conocidas también como lista de comprobación o verificación, hoja de control o de recogida de datos, hoja de registro, pero el término más empleado es checklist (lista de chequeo).

Una lista de chequeo se define como un formulario de preguntas, las cuales dependen del objetivo para el cual son usadas. Es fácil de aplicar, resumir y comparar. Es flexible. Su análisis es rápido, pues consiste en verificar si existe o no existe un control que es aplicable al sistema en análisis.

El empleo de las listas de verificación se ha extendido en diversos ámbitos que van desde comprobar el nivel potencial de mercados extranjeros o las aptitudes del personal de cierta entidad hasta medir la fiabilidad de sistemas informáticos. Para ejemplificar se puede citar la evaluación de un portal web que incluye como criterios de valoración, la usabilidad, accesibilidad, y navegación del portal; tan solo para mencionar algunos criterios.

La elaboración de una lista de control en algunos casos requiere de un personal calificado. Por ejemplo, para evaluar artefactos generados por un proyecto que diseñe usando la metodología RUP, específicamente un diagrama de despliegue, la lista debe ser confeccionada por alguien que tenga un previo conocimiento básico de UML (Unified Modeling Language) y suficiente experiencia en la creación de diagramas de despliegue. Además las listas deben ser clasificadas de acuerdo a la urgencia de los criterios a evaluar y su importancia.

Las listas de comprobación aparecen en diversos medios de soporte. Existen algunas que pueden hacer cálculos básicos basados en los resultados obtenidos de los criterios a evaluar-una lista en una hoja Excel- que puede brindar el total de respuestas positivas y negativas.

Se pueden realizar de diferentes maneras. Las preguntas en forma de cuestionario sirven como guía, como "supervisora" que obliga a la persona que la responda a meditar sobre el cumplimiento de ciertas normas. Las listas de verificación enumeran una serie de criterios (muchos o pocos) que deben ser verificados uno a uno para garantizar la calidad esperada del producto.

Entre las más prácticas y sencillas se destacan aquellas que son en forma de cuadro pues posibilitan una rápida realización. Se pueden contestar con sí o no, marcar la casilla que se está verificando o dejar la casilla en blanco en caso que no se cumpla. En este tipo de formato es conveniente dejar un espacio bien amplio para poder anexar las observaciones que se hallan hecho en el transcurso de la

verificación.

Otro formato de listas es un listado de preguntas con espacios libres al final que deben ser respondidas con frases sencillas y precisas. El otro diseño cuenta con los descritos anteriormente, alterna los dos formatos colocando en algunas partes casillas y en otras espacios libres.

El factor dinamismo es muy importante en las listas pues muchas veces se aprecian situaciones o criterios que deben ser explicados. Esto se logra a través de los comentarios que se colocan debajo de los elementos a verificar. Este dinamismo repercute en la realización de una nueva versión de la lista pues debe reflejar los "requisitos" que se deriven de las anteriores.

Es conveniente que las listas tengan un formato sencillo, práctico y fácil de visualizar para que el personal encargado de esta actividad se familiarice de forma rápida y la añada a su rutina de manera natural.

El contenido y la extensión de la lista también resultan ser variada pues las hay breves y muy extensas, complejas y sencillas. La complejidad y la integridad de las listas de comprobación no siempre garantizan el éxito de tal proceso. Un contenido muy extenso y difícil puede provocar rechazo en quien la efectúa ya sea por el tiempo que se invierte en hacerlas o por el tiempo en comprenderlas. También puede traer como consecuencia que no se responda o de hacerlo se realice de manera rápida e incorrecta, con el solo fin de cumplir con la obligación, sin prestar atención a lo que se dice. Estos casos pueden exponer al sistema que se evalúa de una escasa calidad.

Las hojas de control son respondidas generalmente por personas que tienen muchas otras actividades por realizar, por lo tanto, deben ser generadas con el fin de ayudar a las personas que las ejecuta. Su fin no es agobiar, ni recargar el trabajo sino todo lo contrario, brindar rapidez y calidad al sistema en análisis. Es aconsejable generar listas de comprobación con un cuestionario breve, conciso y fácil de responder que contemple los criterios más necesarios y representativos que brinden la certeza que serán atendidos y bien respondidos.

Las hojas de control tienen más de un destinatario y usuario pues sirven para verificar el trabajo que efectuó una determinada persona o entidad y por otra parte para que el individuo que realizó el trabajo tenga conocimiento del cumplimiento de ciertas normas ineludibles.

Existen algunas características de las listas de comprobación que puede garantizar su éxito en cuanto a aceptación e incorporación para su empleo:

- ✓ Debe ser de fácil compresión.

- ✓ No consumir demasiado tiempo para responderlas. Sólo lo necesario.
- ✓ Evitar las redacciones confusas.

Las listas existen de manera abundante y libre. Para su uso se pueden utilizar las que ya están elaboradas o diseñar una propia, esta última opción es mucho más factible pues contempla con precisión los intereses de quienes van usarlas y las particularidades de la entidad donde será aplicada. Su redacción debe ser encargada a la persona idónea.

1.9 Calidad de la Producción de Software en Cuba

En Cuba el desarrollo de una Industria Nacional de Software es una tarea de gran prioridad, debido a la alta perspectiva económica que posee, así como para el aseguramiento de un grupo importante de actividades del país. A pesar de ello, los resultados alcanzados no cubren las expectativas, ya que la productividad es baja, la cantidad real de recursos a consumir -en tiempo principalmente- es casi impredecible y el trabajo realizado casi nunca tiene la calidad y profesionalidad requerida. Los proyectos están excesivamente tarde y los beneficios que pudieran obtenerse al utilizar los mejores métodos e instrumentos en las distintas etapas no se detectan en este medio indisciplinado y caótico de desarrollo, (Febles Estrada, 2000).

En estudios realizados por el Centro de Estudios de Ingeniería de Sistemas (CEIS) perteneciente al Instituto Superior Politécnico “José Antonio Echeverría” (CUJAE), en empresas cubanas se detectaron problemas relacionados con (ÁLVAREZ, 2000):

- ✓ El personal disponible en estas empresas es aún escaso aunque tiene un alto nivel de preparación.
- ✓ Existe una gran desorganización en las empresas lo que no permite aplicar técnicas, modelos o estándares que ayudarían al desarrollo de ésta.
- ✓ Existen pocos clientes y no se ha creado una disciplina para el intercambio con ellos.
- ✓ El mercado externo es aún muy pobre, pues prácticamente no se obtienen ingresos por concepto de exportación de software.
- ✓ No existe una cultura de producción de software bajo parámetros de terminación y calidad, donde se actúe bajo conceptos y estudios técnicamente fundamentados por equipos multidisciplinarios y competentes dirigidos a la creación de un producto orientado a determinado mercado.
- ✓ Hay mala calidad en gran parte del software que se produce en el país y es indispensable la

solución de este problema lo más brevemente posible, pues la calidad del producto que desarrollan las empresas nacionales es clave para mejorar su competitividad, y teniendo la calidad del producto como elemento distintivo, éstas pueden encontrar nuevos mercados.

Muchas empresas o entidades han aplicado métodos de evaluación para garantizar la calidad de los sistemas informáticos durante su ciclo de vida. El Centro de Estudios de Ingeniería de Sistemas (CEIS) no está exenta de esta actividad evaluativa pues ha desarrollado un sistema que permite hacer un seguimiento de las inspecciones a los proyectos de la empresa incluyendo las posibles listas de chequeo y terminando con la documentación final de la inspección que se almacena para que pueda ser consultada y así comparar los resultados que produce la revisión en la calidad final de producto de software, (Delgado, 2000).

Esas listas de comprobación tienen la función de evaluar la existencia de algunos artefactos al concluir la etapa final del proyecto, o sea, vela por la existencia de un glosario de términos, por la documentación de todos los requisitos funcionales, que estén documentados los riesgos, tan sólo para citar ejemplos.

En el caso de la UCI, a lo largo de estos seis años se ha ido trabajando en crear una cultura de la calidad en los diferentes proyectos y con este objetivo surge la dirección de Calidad de Software en la Universidad, así como los asesores de calidad en las facultades. Se han establecido requisitos mínimos de Calidad que cada proyecto debe cumplir, se realizan auditorías y pruebas de liberación a las aplicaciones y se emplean las listas de verificación para la revisión de la documentación pero existe el inconveniente que no se han elaborado todas las listas de chequeo para cada artefacto del expediente de proyecto y los proyectos no conocen los parámetros que estas evalúan.

Conclusiones Parciales del Capítulo

Las temáticas abordadas en este capítulo sustentan con profundidad la presente investigación. Inicialmente se abordó sobre los procesos de desarrollo y la necesidad que estos se adapten a las necesidades del cliente o del proyecto. De igual manera se expusieron las metodologías más empleadas en el mundo y en la universidad, haciendo énfasis en RUP. También se trató el tema de la calidad donde se analizaron los tipos de calidad, así como el control y la garantía de la misma. Se enfatizó sobre la evaluación del software en etapas tempranas mediante el empleo de técnicas y herramientas básicas, destacando entre éstas la lista de chequeo, que a pesar de ser despreciada por muchas personas está considerada como la herramienta más conveniente e idónea para la revisión de la documentación.

Capítulo 2: Propuesta de Listas de Chequeo

Introducción

En este capítulo se realiza un análisis de la producción en la universidad, se referencia en qué consiste el expediente de proyecto de calidad y se describen los principales artefactos según la metodología RUP de manera precisa y breve. Además se explica la estructura de las listas de chequeo que se proponen, a qué artefactos van dirigidas y la forma establecida para evaluar la calidad de los mismos.

2.1 La Producción en la Universidad

"Un lugar para crear ideas y conocimientos"

Fidel Castro Ruz

La Universidad de las Ciencias (UCI) es una universidad productiva, cuya misión es producir software y servicios informáticos a partir de la vinculación estudio – trabajo como modelo de formación, (WEB02).

La producción se concentra en el desarrollo de proyectos en más de 30 Polos Productivos y se destacan resultados en las esferas de salud, educación, software libre, teleformación, sistemas legales, realidad virtual, automatización, bioinformática, procesamiento de imágenes y señales, entre otras, (WEB02).

Promueve el desarrollo de productos y servicios informáticos en aquellas ramas donde Cuba tiene un reconocido prestigio en el mundo a través del concurso de los mejores especialistas del país para lograr una solución de calidad y de impacto internacional, (WEB02).

La producción está regida por la Infraestructura Productiva más conocida como IP. La IP es la encargada de brindar servicios relacionados con la producción de sistemas informáticos en la universidad, entre los más importantes se destacan la calidad de software llevada a cabo por un grupo de personas que aplica ciertas herramientas y técnicas como las revisiones, inspecciones y auditorías en laboratorio de calidad. Por otra parte está el servicio de arquitectura y tecnología, servicios legales y diseño de comunicación visual.

2.2 Expediente de Proyecto

Para lograr estandarizar los entregables de cada proyecto la Dirección de Calidad estableció un expediente de proyecto que contempla diferentes áreas del proceso de desarrollo, como se muestra en la Figura 8. La mayoría de los artefactos que se recogen en el expediente siguen las pautas impuestas

por la metodología RUP, por ser esta precisamente la más usada en la Universidad e impartida en la docencia.

Expediente de Proyecto			
Ingeniería	**Gestión de Riesgos**	**Soporte**	**Legal**
Requisitos	Plan de Proyecto	Aseguramiento de la Calidad	Organización
Arquitectura y Diseño	Riesgos	Gestión de Configuración	
Implementación y Pruebas	Recursos		
Despliegue e Instalación	Acuerdos de Trabajo		
	Información del Cliente		
	Informes		
	Reuniones		

Fig.8: Descripción de los recursos del expediente del proyecto.

Los artefactos en que se enfoca esta investigación están relacionados con el área de Ingeniería, se escoge esta área porque a pesar de haberse definido qué debe llevar cada documento todavía persiste gran cantidad de errores en los mismos.

A continuación se da una breve explicación de los elementos que se debe tener en cuenta para lograr la calidad de cada artefacto.

2.2.1 Descripción de los Artefactos

Modelo del Dominio

Es también conocido como Modelo Conceptual. Es una representación de conceptos en un dominio del problema (Fowler, 1996), o sea es una descripción del dominio de un problema real y no una descripción del diseño del software. Contiene conceptos y asociaciones entre conceptos.

Modelo de Objetos

Se lleva a cabo durante el modelado del negocio. El modelo de objeto describe cómo interactúan los trabajadores y las entidades del negocio. Las entidades del negocio representan todos los objetos que evalúan y manipulan los trabajadores del negocio. Generalmente estos objetos representan elementos tangibles, como un documento, o elementos intangibles como el conocimiento acerca de algo o de alguien.

Diagrama de Actividades

Un diagrama de actividad describe un proceso que explora el orden de las tareas o actividades que logran los objetivos del negocio. Es similar a un diagrama de estados en el cual todos o la mayoría de los estados son estados de actividad y en la cual todas o la mayoría de las transiciones se disparan al completarse las acciones en los estados fuentes precedentes.

Glosario de Términos

Es un documento simple en el cual se definen términos. El glosario o diccionario modelo registra y precisa todos los términos que requieren explicación, con el objetivo de mejorar la comunicación y mermar malos entendimientos. Se construye desde la fase de Inicio y va perfeccionándose consecutivamente en cada ciclo de desarrollo al aparecer nuevos vocablos. Los términos registrados están organizados alfabéticamente y en ocasiones pueden formarse grupos de términos, los cuales se deben organizar según este mismo criterio. También registra las reglas del dominio de cierta entidad y las restricciones.

Especificación de Casos de Uso

La especificación de casos de uso es una descripción del uso del sistema y se centra en la intención del actor. Su objetivo radica en describir los requisitos funcionales y originar un resultado de valor para un actor en particular.

La especificación de casos de uso puede estar dada por una descripción textual o gráfica. En la descripción textual se evalúa que se describa detalladamente cada uno de los escenarios del caso de uso que se analiza, que la descripción se realice en tiempo presente y otras cuestiones. En la descripción gráfica, o sea, en el diagrama de casos de uso se controla que todos los casos de uso estén conectados con al menos un actor, que capturen detalladamente lo que el sistema debe hacer, que el diagrama en general pueda ser bien comprendido, esté bien detallado y entre otros aspectos.

Documento de Arquitectura del Software

El documento de arquitectura posibilita controlar el desarrollo del sistema en cuestión. Está centrado en diferentes vistas, vista de casos de uso, vista de procesos, vista lógica, vista de implementación y vista de despliegue. La descripción de la arquitectura describe las partes del sistema más significativas para los desarrolladores y otros interesados.

La vista de casos de uso integra elementos esenciales que se deben tener en cuenta durante la modelación de los casos de uso, incluyendo además aspectos del diagrama de actividades. La vista lógica abarca sobre elementos de la modelación de diseño y la vista de implementación sobre el modelado de componentes.

Especificación de Requerimientos

La especificación de requisitos de software es un documento que agrupa los requerimientos del sistema. Constituye el resultado de las actividades desarrolladas durante el flujo de trabajo de Requisitos.

Su representación debe ser precisa pues especifican lo que el sistema debe hacer, no debe existir ningún requisito contradictorio, o con ambigüedades, que dé lugar a diferentes interpretaciones y deben ser claros, ya que se debe entender lo que está especificado.

La especificación de requisitos del software (ERS) traza los siguientes objetivos (Chalmeta, 1999):

1. Ayudar a los clientes a describir claramente lo que se desea obtener mediante un determinado software: El cliente debe participar activamente en la especificación de requisitos, ya que éste tiene una visión mucho más detallada de los procesos que se llevan a cabo. Asimismo, el cliente se siente partícipe del propio desarrollo.
2. Ayudar a los desarrolladores a entender qué quiere exactamente el cliente: En muchas ocasiones el cliente no sabe exactamente qué es lo que quiere. La ERS permite al cliente definir todos los requisitos que desea y al mismo tiempo los desarrolladores tienen una base fija en la que trabajar. Si no se realiza una buena especificación de requisitos, los costes de desarrollo pueden incrementarse considerablemente, ya que se deben hacer cambios durante la creación de la aplicación.
3. Servir de base para desarrollos de estándares de ERS particulares para cada organización: Cada entidad puede desarrollar sus propios estándares para definir sus necesidades.

Este documento no debe describir ningún detalle de diseño, modo de implementación o gestión del proyecto, pues los requisitos se deben reflejar de manera que el cliente del sistema pueda comprenderlo.

Documento de Análisis

La evaluación de este documento está dada por la construcción del modelo de análisis. El modelo de análisis a su vez va contener las clases de análisis y sus relaciones y los diagramas de interacción (secuencia y colaboración).

Las clases de análisis se pueden estereotipar de tres maneras diferentes: de interfaz, de control o de entidad. Las clases interfaz se usan para modelar la interacción entre el usuario y el sistema.

Las clases de entidad modelan información de larga duración y que en ocasiones es persistente y las clases de control representan la coordinación de las actividades de los objetos.

Documento de Diseño

Este documento contempla elementos que se deben tener en cuenta durante la modelación del diseño. De modo general abarca aspectos como el comportamiento entre las clases de diseño, elementos que se debe tener en cuenta a la hora de diseñar una base de datos y realizar un diagrama de clases persistentes y demás.

Modelo de Despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos y las conexiones entre estos. Puede ser empleado también para visualizar la distribución de los componentes de software en los nodos físicos.

El modelo de despliegue contiene nodos, que son los elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos; los dispositivos, nodos sin capacidad de procesamiento tal como una impresora u otro periférico y los conectores que establecen las relaciones entre los nodos y dispositivos.

Diseño Web

Las WebApps posibilitan que los usuarios dispongan de una gran variedad de contenido y funcionalidad. Es una mezcla de de publicación impresa y desarrollo de software, de marketing e informática, de comunicaciones internas y relaciones externas, y de arte y tecnología (Powell, 1998).

Es un sistema de software con estado de negocio, y por tanto puede elaborarse a través de la metodología de trabajo que propone RUP empleando para la modelación los estereotipos apropiados.

Documento de Arquitectura de Información

La Arquitectura de Información (AI) es una disciplina relativamente nueva que nace tras la definición que hace Richard Saul Wurman en 1975, en su libro titulado “Information Architects” publicado al año siguiente, (WEB03).

La arquitectura de información es una disciplina que organiza conjuntos de Información, permitiendo que cualquier persona los entienda y los integre a su propio conocimiento, de manera simple. Desde el punto de vista de quienes la utilizan para la construcción de sitios o aplicaciones web, se define como el conjunto de prácticas que entendiendo el objetivo de un sitio o aplicación web, organiza el contenido en subconjuntos de nombres comprensibles para el usuario final, facilitando las operaciones de búsqueda y uso de la información que contienen (WEB03).

Modelo Entidad Relación

El modelo de entidad relación se basa en la percepción del mundo real. Contiene objetos de datos llamados entidades; atributos, que definen las propiedades de las entidades y los conectores, que muestran las relaciones entre las entidades. Las relaciones pueden ocurrir de diferentes formas.

Los atributos se pueden clasificar en simples o compuestos, en univaluados o multivaluados, en derivados e identificadores. Las entidades pueden ser débiles o no.

Diccionario de Datos

El diccionario de datos es un listado organizado de de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que permiten que el usuario y el analista del sistema tengan una misma comprensión de las entradas, salidas, de los componentes de almacenamiento y también de los de cálculos intermedios, (Yourdon, 1990).

Manual de Instalación

El documento refleja los lineamientos a seguir para instalar el sistema. Contiene información sobre la infraestructura de instalación e instrucciones para la instalación y actualización del software.

Manual de Usuario

El manual de usuario describe las características técnicas y de funcionamiento de la aplicación. Es un documento técnico de comunicación, que brinda ayuda a las personas que usan un sistema en

particular. Generalmente son asociados al género electrónico, hardware de la computadora y software. Una vez familiarizado con el mismo pasa a ser un material de consulta para cuando se ha olvidado algún aspecto o se presenta incertidumbre.

2.2.2 Los Modelos en su generalización

Los modelos ofrecen información substancial acerca del sistema que se está construyendo. La revisión de éstos proporciona un aseguramiento en cuanto a consistencia, compleción y exactitud.

La consistencia de los modelos está basada en las relaciones entre sus objetos, la exactitud se basa en dos aspectos, en el sintáctico y semántico. La exactitud sintáctica se juzga en el uso apropiado de la simbología; cada modelo debe mantener las convenciones propias del modelado según la herramienta empleada. La exactitud semántica está dada en la correspondencia del modelo con el dominio del problema en el mundo real. La compleción de los modelos es juzgada en que sólo se especifique lo que el sistema debe hacer.

2.3 Propuesta de Lista de Chequeo

Con el objetivo de brindarle a cada desarrollador una guía para verificar que los artefactos generados siguen los criterios definidos por la *metodología RUP,* y que se no se ha obviado ninguna información, así como posibilitarles al grupo de calidad una guía para la revisión de dichos artefactos se realiza una propuesta de listas de chequeo a los artefactos ingenieriles. Como se dijo anteriormente van a existir dos tipos de listas de chequeo en dependencia de a quién van dirigidas:

1. Dirigida a los desarrolladores.
2. Dirigida al grupo de calidad.

En algunos casos las listas pueden coincidir porque los criterios a comprobar son los mismos, en la tabla se muestra para cuáles artefactos el equipo de desarrollo usa una lista y el grupo de calidad usa otra y para cuáles es la misma lista.

Artefactos	**Lista de chequeo dirigida a los desarrolladores**	**Lista de chequeo dirigida al grupo de calidad**
Modelo del Dominio	Coincide	
Diagrama de Actividades	x	x
Modelo de Objeto	Coincide	

Glosario de Términos	Coincide	
Especificación de Casos de Uso	x	x
Documento de Arquitectura del Software	Coincide	
Especificación de Requisitos del Sistema	x	x
Documento de Análisis	Coincide	
Documento de Diseño	x	x
Modelo de Despliegue	Coincide	
Diseño Web	Coincide	
Documento de Arquitectura de Información	Coincide	
Modelo Entidad Relación	x	x
Manual de Instalación	Coincide	
Manual de Usuario	Coincide	

Tabla 2: Abstracción de listas de chequeo dirigida a los desarrolladores, al personal de calidad o ambos.

2.3.1 Propuesta de Plantilla

La dirección de Calidad ya había definido con anterioridad una plantilla para las listas de chequeo. La misma contaba con los siguientes campos:

- ✓ Nivel: Define los criterios de mayor importancia que otros.
- ✓ Criterio de Evaluación: Campo que define el aspecto que será evaluado.
- ✓ Evaluación: Define la evaluación de los aspectos en un rango de 0 a 5.
- ✓ No Procede: Campo que indica que ese aspecto no es factible a valoración.
- ✓ Observaciones: Define cualquier opinión que desee dar la persona que aplica la lista.

- ✓ Métricas, Submétricas y % Cumplimiento: Es para aplicar métricas a las iteraciones que sean realizadas.
- ✓ Tabla de Registro de Defectos y dificultades detectadas: En esta tabla se registran las no conformidades encontradas en la revisión del artefacto.

Con el objetivo de realizar una evaluación detallada a los diferentes artefactos se propone una nueva plantilla, como bien lo evidencia la tabla 3, con las siguientes secciones:

Estructura del Documento: Contienen todos los aspectos definidos por el expediente de proyecto o el formato establecido por el mismo.

Elementos definidos por la Metodología: Comprende todos los indicadores a evaluar según la metodología.

Semántica del Documento: Define todos los indicadores a evaluar respecto a la ortografía, redacción, formato, entre otros.

Cada una de estas secciones contiene los siguientes campos:

- Peso: Define si el indicador a evaluar es crítico o no.
- Indicadores a Evaluar: Campo que define el aspecto que será evaluado.
- Evaluación: Es la forma de evaluar la lista. La misma se evalúa de 0 en caso de mal y 1 en caso que esté bien.
- Cantidad de Elementos Afectados: Especifica la cantidad de errores encontrados.
- Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.
- No Procede: Campo que indica que ese aspecto no es factible a valoración.
- Tabla de Registro de Defectos y dificultades detectadas: Es la misma que emplea el Departamento de Calidad.

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**

Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**

Tabla 3: Plantilla propuesta.

2.3.2 El Proceso de Evaluación de las Listas de Chequeo

Con el objetivo de brindar una evaluación final del artefacto que se está chequeando se empleó los criterios de criticidad (Anexo 2) que se adecuaban a la revisión de la documentación. Estos criterios fueron propuestos por el MSc. Michael González Jorrín perteneciente a la Dirección de Calidad,

Luego si el artefacto que se está evaluando cumple con los criterios de criticidad la revisión del documento es abortada. Los criterios utilizados son:

- ✓ El promedio de las No Conformidades críticas por casos de uso es superior a uno.

Este criterio es empleado únicamente para los casos de uso. En caso de otro artefacto la revisión se aborta si:

- ✓ Existen al menos dos indicadores críticos evaluados de mal.
- ✓ Existe más de una falta de ortografía por página o pantalla en caso de ser una interfaz.
- ✓ Incumple con más del 50 % de los indicadores a evaluar de la sección Estructura del Documento que posee la lista de chequeo.
- ✓ Se mantienen las No Conformidades de una revisión a otra.

Se da una evaluación de regular si cumple con los siguientes criterios:

- ✓ Existe una No Conformidad crítica.
- ✓ La cantidad de elementos afectados de un indicador evaluado de mal es superior a tres.

Estos criterios se cumplen para todas las secciones que tiene la lista de chequeo.

La lista es evaluada de bien si:

- ✓ No existe ninguna No Conformidad relacionada con indicadores con peso crítico.
- ✓ Si la cantidad de elementos afectados de un indicador que no sea crítico no es mayor que dos.

En las listas dirigidas a los desarrolladores ellos no abortan la revisión del artefacto. Estos criterios simplemente le indican que su artefacto está muy mal elaborado.

A continuación se muestra la lista de chequeo correspondiente al artefacto Modelo de Objeto.

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. Para garantizar mejor la claridad del modelo de objeto ¿ha confeccionado un diagrama por cada caso de uso?				
	2. ¿Ha identificado y representado las relaciones (asociación, dependencia) entre los objetos?				
Crítico	3. ¿Ha agrupado los objetos en trabajadores del negocio y				

	entidades del negocio?				
Crítico	4. ¿Los trabajadores, entidades y eventos del negocio participan en al menos una realización de caso de uso?				
Trabajadores del Negocio (NT)					
	1. ¿Han sido identificados los Trabajadores del Negocio?				
	2. ¿Representan los Trabajadores del Negocio un sistema o persona que interactúa directamente con el negocio?				
	3. ¿Ha asociado el nombre de los Trabajadores del Negocio con sus responsabilidades?				
	4. ¿Ha descrito los Trabajadores del Negocio?				
	5. ¿El nombre y la descripción de los Trabajadores del Negocio es clara y legible?				
	6. ¿Los Trabajadores del Negocio se relacionan con al menos un caso de uso?				
Entidades del Negocio (ET)					
	1. ¿Han sido identificados como contenedores de información?				
	2. ¿Es el nombre claro y legible?				
	3. ¿Se ha relacionado con al menos un caso de uso?				
Reglas den Negocio (RN)					
	1. ¿Han sido identificadas las RN?				

	2. ¿Ha identificado de manera correcta las categorías de las RN?				
	3. ¿Están las RN identificadas por una pequeña descripción o número?				
	4. ¿Está completamente claro el contexto de las RN?				
	5. ¿Ha evitado las ambigüedades en las RN?				
	6. ¿Ha capturado (reflejado) las RN en un documento o modelo?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Tabla 4: Lista de chequeo correspondiente al artefacto Modelo de Objeto.

Las demás listas de chequeo pueden ser encontradas en el Anexo 4

Conclusiones Parciales del Capítulo

El presente capítulo tuvo como objetivo exponer la importancia de la producción en nuestra universidad, qué aspectos contendrían las listas, cuál sería la plantilla a emplear y el modo de evaluación.

La propuesta definida en este trabajo proporcionará a los desarrolladores y al el equipo de calidad una herramienta eficiente y fácil de emplear que les posibilite determinar si los artefactos que se han creado cumplen en completitud y normas según establece el Proceso Unificado de Racional.

Capítulo 3: Validación de la Propuesta de Listas de Chequeo

Introducción

Con el objetivo de comprobar la efectividad de la investigación, la propuesta mencionada en el capítulo anterior se validó a través del criterio de un panel de expertos. Este proceso requiere tener en cuenta ciertos pasos, tales como la selección de los expertos, la elaboración del cuestionario y el análisis cualitativo y cuantitativo del resultado obtenido de los cuestionarios aplicados. Estos pasos serán explicados detalladamente en las secciones de este capítulo.

3.1 Introducción al Criterio de Expertos

Para la validación y aceptación de dicha propuesta se empleó el criterio de un panel de expertos y el empleo de técnicas que propone el método de Delphi.

El método Delphi también denominado como Delfos, nombre cuyo origen proviene del oráculo de la antigua Grecia, fue creado alrededor de los años 1963-1964 por la Rand Corporation, específicamente por Olaf Helmer y Dalkey Gordon, con el objetivo de elaborar pronósticos a largo plazo, referentes a posibles acontecimientos en varias ramas de la ciencia, la técnica y la política. En resumen, este método es el uso consecuente del juicio intuitivo de un conjunto de expertos para lograr un consenso de opiniones.

3.1.1 ¿Qué es el Criterio de Expertos?

(POLLÁN, 2003) y (URIZARRI, 2004) definen experto al individuo en sí como a un grupo de personas u organizaciones capaces de ofrecer valoraciones conclusivas de un problema, y hacer recomendaciones respecto a sus momentos fundamentales con un máximo de competencia.

3.2 Objetivo del Empleo del Criterio de un Panel de Expertos

La aplicación de este método pretende los siguientes objetivos:

- ✓ Obtener una valoración sobre la utilidad de la propuesta.
- ✓ Identificar aspectos erróneos y recomendaciones que posibiliten mejorar la propuesta.
- ✓ Determinar si es necesario agregar o eliminar aspectos evaluativos a las listas.
- ✓ Determinar si los artefactos generados en el expediente de proyectos son necesarios y suficientes.

- ✓ Determinar qué aspecto evaluativo de cada lista es de suma importancia basándose en las listas de chequeo de RUP que propone las características fundamentales que debe cumplir cada artefacto.

3.3 Selección de los Expertos

La selección de los expertos resulta una tarea de de gran importancia pues estos individuos deben tener un amplio conocimiento con relación a la propuesta a evaluar, además debe apreciarse profesionalismo, capacidad de análisis y que posean amplitud de enfoques para examinar la propuesta presentada (ASTIGARRAGA, 2002).

Se tuvo en cuenta en este proceso de selección el área de Dirección de Calidad del Software y los laboratorios de proyectos productivos de la Universidad.

3.3.1 Aprobación de los Expertos en la Validación de la Propuesta

Tras el proceso de selección se hizo una invitación formal por correo explicándoles de manera detallada el objetivo de la propuesta, la realización de la encuesta, el plazo y orden de ejecución de la misma.

3.3.2 Cantidad de Expertos a Seleccionar

Se recomienda como mínimo un total de siete expertos pues el error por cada experto adicionado reduce hasta llegar a los siete. Además no es aconsejable recurrir a más de 30 expertos, pues la mejora en la previsión es muy pequeña y normalmente el incremento en coste y trabajo de investigación no compensa la mejora.

3.4 Elaboración de la Encuesta

Con el objetivo de evaluar la competitividad de los expertos se elaboró una encuesta (ver Anexo 3) para determinar el nivel de conocimiento acerca de la metodología RUP, pruebas e ingeniería del software y una vez seleccionados los expertos se empleó otra encuesta (Anexo 4) para conocer las posibles insuficiencias de la propuesta y después rectificarlas.

3.4.1 Determinación del Coeficiente de Competencia

El método consiste en calcular el coeficiente de competencia (k) del experto a partir de su autovaloración sobre el conocimiento o información que tiene sobre el tema (kc) y el coeficiente de argumentación o valoración (ka) empleando la siguiente ecuación:

$k = (kc + ka)/2$

La estrategia de interpretación de los coeficientes de competencias es como sigue:

Si 0,8 < = k < =1,0 coeficiente de competencia alto.

Si 0,5 < = k < 0,8 coeficiente de competencia medio.

Si 0< = k < 0,5 coeficiente de competencia bajo.

¿Cómo calcular kc?

Para calcular el conocimiento o información que tiene sobre el tema se multiplica por 0,1 el valor del rango escogido por encuestado.

Expertos	**1**	**2**	**3**	**4**	**5**	**6**	**7**
kc	0.7	0.6	0.9	0.7	0.6	0.6	0.9

Tabla5: Resultados del nivel de competencia de los expertos.

¿Cómo calcular ka?

Para calcular el coeficiente de argumentación o valoración al experto se le presenta una tabla sin cifras para que marque con una (x) el grado de influencia de las fuentes, de acuerdo a los niveles ALTO, MEDIO y BAJO. Luego empleando los valores que aparecen en la tabla patrón, tabla 5, se determina el valor de ka para cada aspecto.

Si Ka = 1, esto implica una influencia alta del total de fuentes.

Si Ka = 0.8, influencia media del total de fuentes.

Si Ka = 0.5 influencia baja del total de fuentes.

Fuentes de Argumentación	**Grado de influencia de cada una de las fuentes**		
	ALTO	MEDIO	BAJO
Análisis teóricos realizados	0.3	0.2	0.1
Experiencia obtenida	0.5	0.4	0.2
Trabajos o investigaciones de autores nacionales	0.05	0.05	0.05
Trabajos o investigaciones de autores internacionales	0.05	0.05	0.05
Conocimiento del estado actual del problema	0.05	0.05	0.05
Intuición	0.05	0.05	0.05
Total	**1**	**0.8**	**0.5**

Tabla 5: Tabla Patrón del grado de influencias de las fuentes de argumentación.

Los resultados de los expertos sobre el grado de influencias de las fuentes de argumentación y el nivel de competencia se muestran a continuación:

Fuentes de Argumentación	**Expertos**						
	E1	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**
1	*0.2*	*0.3*	*0.3*	*0.2*	*0.2*	*0.2*	*0.3*
2	*0.4*	*0.5*	*0.5*	*0.2*	*0.2*	*0.4*	*0.4*
3	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*
4	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*
5	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*
6	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*	*0.05*
Ka	**0.8**	**1**	**1**	***0.6***	***0.6***	***0.8***	**0.9**

Tabla 6: Resultados del los expertos sobre el grado de influencias de las fuentes de argumentación.

Experto	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**
K	*0.75*	*0.8*	*0.95*	*0.65*	*0.6*	*0.7*	*0.9*
Interpretación	**medio**	**alto**	**alto**	**medio**	**medio**	**medio**	**alto**

Tabla 7: Resultados e interpretación del coeficiente de competencia de los expertos.

3.5 Opiniones Emitidas por los Expertos

Después de haber aplicado las encuestas se analizaron las respuestas de las interrogantes del Anexo 4. A continuación se resume cada una de estas opiniones.

El empleo de las listas resulta una herramienta bastante efectiva, sencilla y rápida para garantizar la calidad del producto. Asegura, de haber sido bien elaborada, la revisión detallada del producto y no lo deja a la intuición de la persona que realiza la revisión. Posibilita además localizar de una manera vertiginosa los posibles errores que puede tener el producto.

La propuesta contiene solamente los artefactos ingenieriles del expediente de proyecto. Es importante resaltar que las metodologías de desarrollo se modifican, ajustándose a las condiciones de progreso, pero que siempre van a existir un conjunto de artefactos que serán de carácter obligatorio por el grado de importancia que posee. Por lo tanto, es de vital importancia revisarlos y que exista con qué.

Con relación a agregar o eliminar criterios evaluativo a las lista se emitió que la lista se enfoca bien en cada elemento, toca cada uno de los puntos esenciales de cada artefacto pero debería abarcar aspectos relacionados a la ortografía, formato, contenido del texto y demás. En aras de mejorar las listas se hicieron los siguientes señalamientos:

- ✓ Algunas preguntas englobaban otras y por lo tanto incumple con unas de las características más importantes que debe cumplir las listas; evitar a todo costo las ambigüedades.
- ✓ Ciertos aspectos deberían ser más detallados pues el nivel de detalle de cada criterio de la lista elevará enormemente el entendimiento de la lista y del artefacto que está siendo chequeado.

En base a las recomendaciones, se realizaron cambios a las plantillas de las listas de chequeo. Todos los expertos emitieron que la propuesta abarca muy bien su objetivo, pues el mismo es brindarle al desarrollador una guía que le permita verificar la completitud de los artefactos siguiendo la metodología RUP y brindarle al equipo de calidad dicha guía para revisar tales artefactos. Además valoraron que la propuesta ofrece aportes significativos pues los mismos se centran en:

- ✓ El diseño de dos listas de chequeo, una para el desarrollador y otra para el departamento de calidad.
- ✓ La creación de una nueva plantilla que contiene las secciones tales como: estructura y semántica del documento y elementos de la metodología propiciando dicha creación un elevado nivel de detalles.
- ✓ El logro de adecuar los criterios de criticidad (Anexo 2) a los aspectos evaluativos de las listas pues debido a sus complejidades no todos los criterios se ajustan a los aspectos de las listas.

En una escala del 1 al 3 siendo el 1 un nivel bajo, el 2 un nivel medio y el 3 un nivel alto de creatividad y efectividad respectivamente, la propuesta fue valorada como alta y media, predominando esta última opinión pues a pesar de estar muy bien fundamentada se considera que debe seguir mejorándose.

A continuación se muestra un resumen sobre la valoración general de la propuesta:

La propuesta del trabajo de tesis es buena por el nivel de importancia y la relación estrecha que tiene sobre los artefactos que se generan en el expediente de proyecto que necesitan ser evaluados.

Además de estar enfocada a los proyectos, también se centra en gran escala a las conferencias de Ingeniería del Software pues orienta de una manera muy bien detallada cómo se debe completar cada uno de los artefactos siguiendo la metodología RUP.

Es muy importante contar con listas de chequeo como un instrumento más de evaluación, pues contribuye enormemente a mejorar la calidad en el desarrollo de software.

Con el objetivo de mejorar la propuesta se opinó que se debería incluir listas de chequeo para revisión los códigos de los productos en diferentes lenguajes ya que estos en ocasiones también es un entregable al usuario.

3.6 Aplicación de las Listas de Chequeo en el Laboratorio de Calidad de la Universidad.

Las listas de chequeo propuestas fueron aplicadas en los laboratorios de calidad al módulo de Administración del proyecto Registro y Notarías y entre otros proyectos tales como CCV, Identidad, Menpet y Sacgir .A continuación se muestra los resultados de tal aplicación evidenciando la cantidad de no conformidades Significativas (S) y No Significativas (NS):

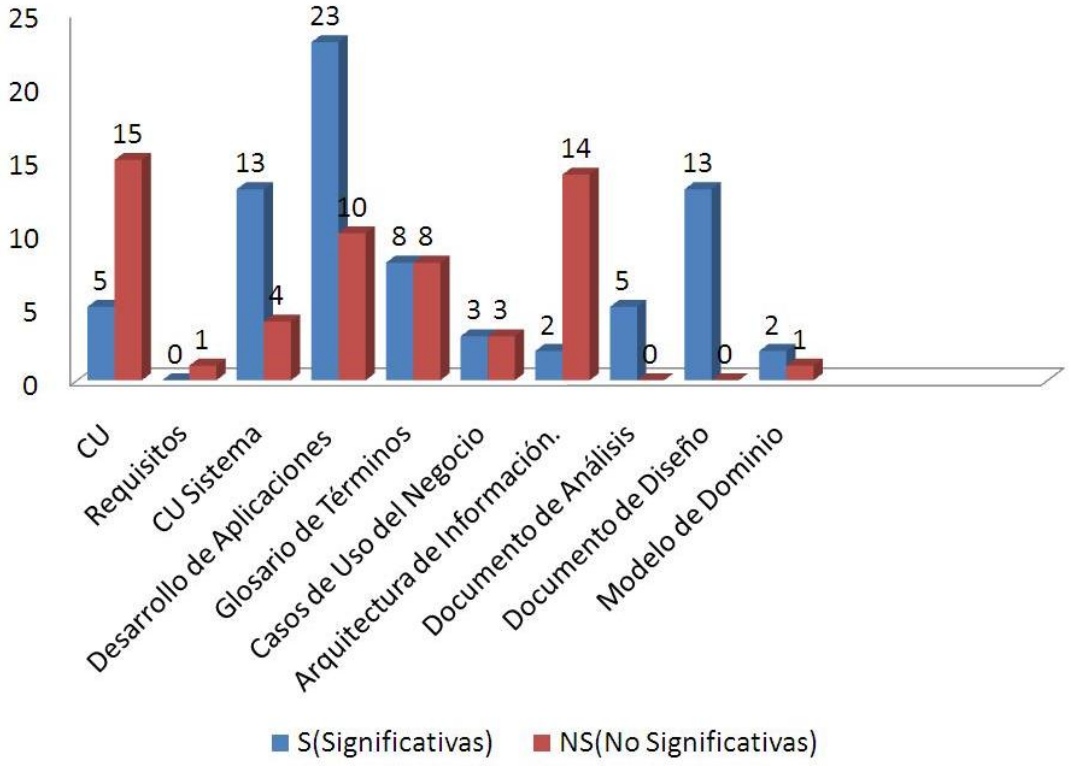

Fig.9: Resultados de la aplicación de las listas en los laboratorios de calidad.

Conclusiones Parciales del Capítulo

La aplicación de las listas reafirmó que la localización temprana de errores evita que el personal que las aplica pierda su tiempo y principalmente que se derroche grandes costes de recursos. La validación de la propuesta a razón del criterio de expertos favoreció obtener desde una perspectiva más confiable una evaluación de los aspectos comprendidos en la misma. Por otra parte determinó cuáles eran de suma importancia auxiliándose de las listas propuestas por RUP que contienen las características esenciales que debe cumplir cada artefacto. Además consideró qué aspecto debía ser eliminado o adicionado con el objetivo de perfeccionar las listas de chequeo y lograr el grado de eficacia una vez que sea aplicada.

Conclusiones Generales

- ✓ Para obtener un sistema con calidad no basta con realizar pruebas una vez que el mismo esté terminado, es necesario, verificar y controlar la calidad de los artefactos generados durante todo el ciclo de vida del software, porque todos los artefactos ingenieriles afectarán de manera positiva o negativa al desarrollo de la aplicación, en dependencia de la calidad de su diseño.
- ✓ Con la aplicación de las listas de chequeo propuestas en esta investigación se logra detectar errores manera rápida y sencilla en la documentación y a la vez sirven de guía al equipo de desarrollo para no cometerlos.
- ✓ Con la propuesta se logra estandarizar la forma de evaluar la documentación y se tiene el criterio de evaluación de la calidad de cada proyecto.

Recomendaciones

Luego de haber desarrollo el presente trabajo de diplomado se recomienda:

- ✓ Llevar a cabo una propuesta que incluya los restantes artefactos que genera el expediente de proyecto.
- ✓ Elaborar una lista de chequeo para el modelo de procesos pues existe tendencia de modelar el negocio empleando dicho modelo.
- ✓ Validar las listas de chequeo de los desarrolladores en los laboratorios de producción de la Universidad.
- ✓ Realizar nuevas versiones de listas de chequeo con el objetivo de una lograr una total perfección.
- ✓ Emplear las listas de los desarrolladores como guía útil para los estudiantes iniciados en la materia de Ingeniería del Software.
- ✓ Publicar las listas de chequeo en el sitio de Calidad de la Universidad.

Bibliografía

1. **(DCCAI), Departamento de Control de Calidad y Auditoría Informática. 1999.** Control de calidad en los sistemas. [En línea] 1999. [Citado el: 11 de 2 de 2008.] http://sistemas.dgsca.unam.mx/publica/pdf/califormat.PDF.
2. **—. 2004.** *Standard Glossary of Software Engineering Terminology, Guide to the Software Engineering Body of Knowledge.* 2004.
3. **ÁLVAREZ, S. 2000.** *Aplicación del modelo CMM a la Empresa Segurmática.* La Habana : Instituto Superior Politécnico "José Antonio Echeverría", 2000.
4. **Amador, Toro Durán. 2000.** *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información.* s.l. : Departamento de Lenguajes y Sistemas Informáticos Universidad de Sevilla, 2000.
5. **ASTIGARRAGA, E. 2002.** *El método Delphi.* s.l. : Universidad de Deusto. Facultad de CC.EE. y Empresariales, 2002.
6. **Canós, José H., Letelier, Patricio y Penades, Maria Carmen.** *Métodologías Ágiles en el Desarrollo de Software.*
7. **Center, Baufest Development.** *Baufest.* [En línea] [Citado el: 12 de 05 de 2008.] http://www.baufest.com/spanish/scrum/scrumconference2006/Que_es_scrum.pdf.
8. **Chalmeta, R. 1999.** *DSI II. 2º Boletín de transparencias.* s.l. : UJI, 1999.
9. **Condori- Fernández, Nelly.** *"Medición y Calidad del Software".* s.l. : Universidad Politécnica de Valencia.
10. **Delgado, Marta. 2000.** *Calidad de los Proyectos de Software: Revisiones utilizando Razonamiento Basados en Casos.* 2000.
11. **Duque, Jorge García. 2000.** [En línea] 2000. [Citado el: 23 de 1 de 2008.] http://idtv.det.uvigo.es/tesispdfs/tesis-jorge.pdf.
12. **Febles Estrada, Ailyn. 2000.** *Calidad de Software y la empresa, enseñanza de un tema imprescindible para el Ingeniero Informático.* s.l. : Instituto Superior Politécnico "José Antonio Echeverría, 2000.
13. **Fernández Carrasco, Oscar M., García León, Delba y Beltrán, Alfa. 1995.** Revista ACIMED. [En línea] 1995. [Citado el: 15 de 3 de 2008.] http://bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm.
14. **Fernández, Nelly Condori. 2007.** *Universidad Politécnica de Valencia.* [En línea] 2007. [Citado el: 1 de 6 de 2008.]

http://inf.ucsp.edu.pe/summerschool/Formal%20Method%20in%20Software%20Development_files/Q&S-Nelly(fINAL).pdf.

15. **Fowler, M. 1996.** *Analysis Patterns: Reusable Object Models.Reading, MA.: Addison-Wesley.* 1996.
16. **IEEE. 1990.** *IEEE Standard Glossary of Software Engineering Terminology.* 1990.
17. **Informática, Departamento de Control de Calidad y Auditoría. 2006.** Control de calidad en los sistemas. [En línea] 2006. [Citado el: 12 de 6 de 2008.] http://sistemas.dgsca.unam.mx/publica/pdf/califormat.PDF.
18. **ISO, 8402. 1994.** [En línea] 1994. [Citado el: 1 de 02 de 2007.] http://fabetsia.dmpa.upm.es/solo_alumnos/sp2/Tablon_sp2/Transparencias CALIDAD06.pdf.
19. **Juristo, Natalia, Moreno, Ana M. y Vegas, Sira. 2006.** TÉCNICAS DE EVALUACIÓN DE SOFTWARE. [En línea] 2006. [Citado el: 12 de 5 de 2008.] http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion_Evaluacion_7.pdf.
20. **Lovelle, Juan Manuel Cueva.** [En línea] [Citado el: 16 de 5 de 2008.] http://di002.edv.uniovi.es/~cueva/asignaturas/doctorado/2001/Calidad.pdf.
21. **Microsoft Corporation.** Métodos Heterodoxos en Desarrollo de Software. [En línea] [Citado el: 20 de 6 de 2008.] http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.mspx.
22. **Molpeceres, Alberto. 2002.** javahispano. [En línea] 2002. [Citado el: 12 de 11 de 2007.] http://www.willydev.net/descargas/articulos/general/cualxpfddrup.PDF.
23. **POLLÁN, A. 2003.** *Particularidades de la función docente-metodológica del profesor de Inglés en las secundarias básicas urbanas del municipio Bartolomé Masó Márquez.* 2003.
24. **Powell, T. A. 1998.** *Web Site engineering,Pretincell-Hall.* 1998.
25. **Pressman, R. 2000.** *Ingeniería del Software: Un enfoque práctico, McGraw Hill 1997.* 2000.
26. **Pressman, R. 2005.** *Ingeniería del Software, Un enfoque práctico.* La Habana : s.n., 2005.
27. **Sommerville, I. 2002.** *Ingeniería de Software, Pearson Educación, 2002.* 2002.
28. **The Reuse, Company. 2006.** *Presente y Futuro de la Reutilización de Software.* 2006.
29. **Universidad Politécnica de Valencia.** [En línea] [Citado el: 16 de 4 de 2008.] https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20Proceso%20de%20Desarrollo%20de%20SW.doc.
30. **URIZARRI, L. 2004.** *Algunas consideraciones acerca del Método de Evaluación por Criterio de Expertos.* 2004.
31. **WEB01.** Entorno Visual de Aprendizaje. [En línea] [Citado el: 5 de 2 de 2008.] http://teleformacion.uci.cu.

32. **WEB02.** Universidad de las Ciencias Informaticas. [En línea] [Citado el: 14 de 6 de 2008.] http://www.uci.cu.
33. **WEB03.** *Arquitectura de Información-Chile.* [En línea] [Citado el: 12 de 1 de 2008.]
34. **Yourdon, E. N. 1990.** *Modern Structured.* 1990.

Glosario de Términos

Artefacto: Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

Aseguramiento: Acción y efecto de asegurar.

Calidad de software: Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con características implícitas que se espera de todo software desarrollado profesionalmente.

Casos de uso: Fragmentos de funcionalidad del sistema.

Control: es el acto de delimitar responsabilidad y autoridad con el fin de liberar la gerencia de detalles innecesarios, conservando los medios para asegurarse de que los resultados sean satisfactorios.

Expediente de Proyecto: Término utilizado para referirse a toda la documentación de un proyecto productivo.

Infraestructura Productiva (IP): Estructura central encargada de dirigir la producción de la UCI y supervisar los proyectos productivos.

Ingeniería de Software: Es una tecnología multicapa en la que se pueden identificar: los métodos que indican cómo construir técnicamente el software, el proceso que es el fundamento de la Ingeniería de Software y las herramientas que son un soporte para el proceso y los métodos. Tiene varios modelos o paradigmas de desarrollo en los cuales se puede apoyar para la realización de software.

ISO: Organización Internacional de Estándares

Iteración: conjunto de periodos de tiempo dentro de un proyecto, en el cual usted produce una versión ejecutable del producto, estable, junto con cualquier otra documentación de soporte, instalación de scripts o similar, necesarios para usar esta liberación.

Paradigma: Modelo o Patrón que se utiliza como guía.

Rol: Define el comportamiento y responsabilidad de un individuo o grupo de individuos en un sistema.

RUP: Proceso Unificado de Rational.

UML (*Unified Modeling Language*): Lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

Anexos

Anexo 1: Características y Subcaracterísticas de Calidad.

FUNCIONALIDAD: La capacidad del producto software para proporcionar las funciones que satisfacen las necesidades establecidas e implícitas cuando el software se utiliza bajo condiciones especificadas.				
Apropiabilidad	Exactitud	Interoperabilidad	Seguridad	Conformidad con la funcionalidad
CONFIABILIDAD: La capacidad del producto software para mantener su nivel de desempeño cuando es utilizado bajo condiciones especificadas (durante un determinado período de tiempo).				
Madurez	Tolerancia a fallas	Recuperabilidad	Conformidad con la confiabilidad	
FACILIDAD DE USO: La capacidad del producto software para ser comprendido, aprendido, utilizado y que sea atractivo para el usuario, cuando es utilizado bajo condiciones especificadas.				
Comprensibilidad	Facilidad de aprendizaje	Operabilidad	Atractivo	Conformidad del uso
EFICIENCIA: La capacidad del producto software para proporcionar un desempeño apropiado, en relación con la cantidad de recurso utilizado, bajo condiciones establecidas.				
Comportamiento en el tiempo	Utilización de los recursos	Conformidad con la eficiencia		

MANTENIBILIDAD: La capacidad del producto software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software a cambios de ambiente y en requisitos y especificaciones funcionales.				
Analizabilidad	Facilidad de cambio	Estabilidad	Facilidad de prueba	Conformidad con la mantenibilidad
PORTABILIDAD: La capacidad del producto software para ser transferido de un ambiente a otro.				
Adaptabilidad	Instalabilidad	Co-existencia	Reemplazabilidad	Conformidad con la portabilidad

Anexo 2: Criterios de Criticidad

Establecen los parámetros para declarar un producto del desarrollo software en estado crítico de terminación. Son aplicables en tres momentos fundamentales: Cuando se hace la revisión de la solicitud (RS); Durante el proceso de pruebas exploratorias (PE) y Durante las pruebas formales. (PF). Los criterios de criticidad son los siguientes:

- ✓ No se presenta la última versión del producto de trabajo comprobada y avalada por la etapa anterior.
- ✓ No están presentes todos los elementos componentes del sistema, producto o entregable. (Hardware, Software, Partes y Documentación). (RS) (PE).
- ✓ No se corresponden totalmente los requisitos funcionales documentados con los implementados. (RS) (PE).

- ✓ Supera el producto la taza de: 1 defecto significativo por CU para (PF) ó ½ defecto significativo por CU para (PE) y en caso de otros artefactos al menos 2 defectos significativos que estén relacionados con las pautas esenciales definidas por la metodología que sigue el proyecto. (PF).
- ✓ Excede el producto el número máximo de iteraciones establecidas por plan (entre 2 y 3 iteraciones). (PF).
- ✓ Se incumple más del 50% de las reglas para la documentación establecidas por el proyecto y/o expediente del proyecto. (RS) (PE) (PF).
- ✓ Existe incoherencia significativa entre los artefactos que tienen relación o dependencia entre sí. (RS) (PE) (PF).
- ✓ Existe incoherencia significativa entre lo documentado y lo implementado. (RS) (PE) (PF)
- ✓ La cantidad de faltas de ortografía excede la cantidad de páginas o pantallas que tiene el artefacto en cuestión. (RS) (PE) (PF).
- ✓ Durante las pruebas de regresión persisten al menos 2 defectos significativos de iteraciones anteriores. (PE) (PF).
- ✓ Se observan los mismos tipos de defectos, ya señalados en iteraciones anteriores, en otros o los mismos lugares donde fueron detectados, para (PE): ½ de la cantidad de defectos tipo encontrados en etapa anterior, o para (PF): ¼ de la cantidad de defectos tipo encontrados en etapa anterior.
- ✓ Si se aplica alguno de estos criterios de criticidad, se considera la prueba fallida y se para la prueba hasta el día siguiente como mínimo.

Anexo 3: Determinación del Coeficiente de Competencia de los Expertos.

1. Marque con una X, en una escala del 1 al 10 el valor que se corresponde con el grado de conocimiento o información que usted considera que tiene respecto a la Ingeniería del Software, la metodología RUP y las Pruebas.

1	2	3	4	5	6	7	8	9	10

2. Señale con una X el nivel de influencia que ha tenido cada una de las fuentes indicadas en su conocimiento sobre el Proceso Unificado del Rational.

No.	Fuentes de argumentación	Grado de influencia		
		Alto	**Medio**	**Bajo**
1.-	Análisis realizado por Ud.			
2.-	Experiencia.			
3.-	Trabajos de autores nacionales.			
4.-	Trabajos de autores extranjeros.			
5.-	Su propio conocimiento del tema.			
6.-	Su intuición.			

Anexo 4: Validación de los principales aspectos de la propuesta.

Propuesta de Listas de Chequeos para verificar la calidad de los principales artefactos generados en el expediente de proyecto de calidad**.**

1. ¿Considera Ud. significativo el empleo de las listas de chequeo como método que puede garantizar la calidad de los artefactos?

Sí___ No___ ¿Por qué?

2. ¿Considera Ud. qué la propuesta contemplan todos los artefactos esenciales que se generan en el expediente de proyecto de calidad?

Especificación de Casos de Uso

Especificación de Requisitos

Documento de Arquitectura de Software

Documento de Arquitectura de Información

Documento de Análisis

Documento de Diseño

Aplicación Web

Diccionario de Datos

Modelo de Despliegue

Modelo Entidad Relación

Modelo de Objeto del Negocio

Manual de Instalación

Manual de Usuario

Glosario de Términos

Sí___ No___ ¿Por qué?

3. ¿Considera Ud. qué la propuesta abarca todos o la mayoría de los aspectos necesarios para la elaboración correcta de los artefactos?

Sí___ No___ ¿Por qué?

4. Señale que aspecto de verificación UD. considera muy importante.

5. ¿Agregaría o eliminaría UD algún aspecto de verificación descrito en la propuesta? Mencione la lista y el aspecto en cuestión.

Sí___ No___ ¿Por qué?

6. ¿Considera UD. qué la propuesta cumplen con su objetivo?

Sí___ No___ ¿Por qué?

7. ¿En que escala del 1 al 3 colocaría UD la propuesta desarrollada? ¿Por qué?

    - 1: nivel bajo de creatividad y efectividad
    - 2: nivel medio de creatividad y efectividad.
    - 3: nivel alto de creatividad y efectividad.

8. Elabore un comentario general sobre el procedimiento que está siendo evaluado que aporte elementos a la mejora del mismo.

Anexo 4 Listas de Chequeo

Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto dónde se referencia las observaciones en cuanto al aspecto a evaluar.

Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del *[nombre del artefacto]* de los proyectos de la universidad.

Esta plantilla ha sido confeccionada para guiar a desarrolladores, especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del *[nombre del artefacto].*

Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.

Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del *[nombre del artefacto]* que se desarrollen.

Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

[Documento para la confección de Listas de Chequeo]
*[**Forma de Uso:***
***Peso*:** *Define si el indicador a evaluar es crítico o no.*

Evaluación (Eval): *Es la forma de evaluar la lista. La misma se evalúa de 0 en caso de mal y 1 en caso que esté bien.*
Cantidad de elementos afectados: *Especifica la cantidad de errores encontrados.*
Comentario: *Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.*
Estructura del Documento: *Contienen todos los aspectos definidos por el expediente de proyecto o el formato de plantilla que establecido por el mismo.*
Elementos definidos por la metodología: *Contienen todos los indicadores a evaluar según la metodología.*
Semántica del documento: *Contemplan todos los indicadores a evaluar respecto a la ortografía, redacción, formato y demás*
N.P. *Significa No Procede y es para el caso de la aplicación que ese aspecto no sea factible su valoración.*

Control de versiones

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

Registro de defectos y dificultades detectados

No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Re Eq
Descripción e la No Conformidad>	*<Descripción del Aspecto correspondiente>*	*<Etapa de detección del error>*	*<X>*	*<X>*	*<X>*	*[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta PD:Pendiente NP:No Procede*	*[Es co pa ite re eq qu co en ca no po*

Modelo del Dominio

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Modelo Conceptual					
	1. ¿Se ha creado un modelo conceptual preliminar?				
Crítico	2. ¿Representan los elementos características del mundo real y no componentes del SW?				
	3. ¿Ha identificado los conceptos u objetos (elementos anteriores) mediante la descripción de los casos de uso o una lista de categoría de				

	conceptos?				
	4. ¿Ha identificado las asociaciones necesarias que contribuyen a la comprensión del modelo?				
Crítico	5. ¿Ha asignado nombre a la asociación, preferentemente una forma verbal?				
	6. ¿Ha definido la multiplicidad?				
	7. ¿Son los atributos simples y válidos?				
Crítico	8. ¿Se ha identificado los términos que requieren explicación?				
	8.1. ¿Se han registrado en el glosario de términos?				
Crítico	9. ¿Sean definido los atributos de las entidades?				
Semántica del documento					

Peso	Indicadores a Evaluar	Eval	(NP)	Cantidad de elementos afectados	Comentarios
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Glosario de Términos

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	¿Todos los términos tienen una definición breve y clara?				
	¿Todos los términos del glosario están incluidos en alguna parte de las descripciones de los casos de uso del negocio?				
	¿Son usados los términos consistentemente en las descripciones breves de los actores del negocio?				
crítico	¿Las definiciones de los términos no son generales y sí específicas para el proyecto?				
	¿El comienzo de la definición de todos los términos está escrito de forma uniforme?				
crítico	¿Todas las palabras y				

	expresiones que se utilizan para definir los términos están claras? En caso de que no lo estén, ¿están definidos en el glosario?				
crítico	¿Se utilizan abreviaturas para definir los términos que no están especificadas en el Glosario?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Diagrama de Actividades-Desarrolladores

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Aspectos Generales					
	1. ¿Has identificado todos los trabajadores?				
	2. ¿Has identificado todos actores del negocio?				
	3. ¿Para una mejor organización y visualización todos los trabajadores del negocio están en un paquete?				
	4. ¿Para una mejor organización y visualización todos los actores del negocio están en un paquete?				
Crítico	5. ¿Has hecho un diagrama				

	de actividades por cada CU descrito en el negocio?				
	6. ¿Ha colocado en el diagrama primero los actores y después los trabajadores?				
	7. ¿El diagrama comienza con su estado de inicio?				
	8. ¿El diagrama termina con su estado de fin?				
Actividades					
	1. ¿Has identificado y detallado todas las actividades?				
	2. ¿Se ha representado con el estereotipo correcto?				
	3. ¿Las relaciones entre las actividades están representadas con una flecha con línea continua?				
	4. ¿El nombre de las actividades empieza con un infinitivo?				
Crítico	5. ¿Las actividades corresponden con el rol (actor o trabajador)?				

Entidades					
	1. ¿Has identificado todas las entidades?				
Crítico	1.1. ¿Representan las entidades objetos con larga vida de duración? Ejemplo: carnet.				
	2. ¿Las entidades se han representado con el estereotipo correcto?				
	3. ¿Ha indicado el estado de la entidad?				
Alternativas					
	1. ¿Ha identificado todas las alternativas de las actividades?				
	2. ¿Han sido representadas con una bifurcación?				
Actividades y Entidades					
	6. ¿Todas las entidades y actividades tienen nombre?				
	7. ¿Las relaciones entre actividades y entidades están denotadas con línea discontinua?				

	7.1. ¿Se ha representado el sentido de navegabilidad?				
	8. ¿Las relaciones entre entidades y decisiones están una flecha con línea continua?				
Crítico	9. ¿Cuando hay una relación entre una entidad y una actividad y la entidad se puede modificar el sentido de navegabilidad se inicia en la actividad y termina en la entidad?				
Crítico	10. ¿Cuando hay una relación entre una entidad y una actividad donde la entidad sólo es consultada y no es modificada el sentido de navegabilidad se inicia en la entidad y termina en la actividad?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				

	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Diagrama de Actividades-Calidad

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿El diagrama se entiende con claridad?				

	2. ¿Existe un diagrama de actividades por cada CU descrito en el negocio?				
Crítico	3. ¿El diagrama contiene solamente un único estado inicial?				
	4. ¿Las actividades y entidades están representadas con el estereotipo correcto?				
Crítico	5. ¿Representan las entidades objetos con larga vida de duración? Ejemplo: carnet.				
	6. ¿Se ha indicado el estado de la entidad?				
	7. ¿Las relaciones entre actividades y entidades están denotadas con línea discontinua?				
	7.1. ¿Se ha representado el sentido de navegabilidad?				
Crítico	8. ¿Cuando hay una relación entre una entidad y una actividad y la entidad se puede modificar el sentido de				

	navegabilidad se inicia en la actividad y termina en la entidad?				
Crítico	9. ¿Cuando hay una relación entre una entidad y una actividad donde la entidad sólo es consultada y no es modificada el sentido de navegabilidad se inicia en la entidad y termina en la actividad?				
	10. ¿Comienza el nombre de la actividad con un infinitivo?				
Crítico	11. ¿Las calles sólo se corresponden con Unidades Organizativas, Trabajadores de Negocio o Actores de Negocio?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				

	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Diseño Web

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**

Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Todos los formularios (Form) han sido identificados y representados?				
crítico	2. ¿Todas las páginas cliente (Client Page) han sido identificadas y representadas?				
crítico	3. ¿Todas las páginas servidoras (Server Page) han sido identificadas y representadas?				
crítico	4. ¿Ha establecido las posibles relaciones de asociación o descomposición entre todos estos elementos?				
crítico	5. ¿Ha identificado las relaciones de asociación entre páginas cliente como link?				
crítico	6. ¿Ha identificado las relaciones de asociación entre páginas servidoras y páginas cliente como build?				

crítico	6.1. ¿El sentido de navegabilidad se inicia en las páginas servidoras y finaliza en las páginas cliente?				
crítico	7. ¿Ha identificado las relaciones de asociación entre las páginas servidoras y los formularios como submit?				
crítico	7.1. ¿El sentido de navegabilidad se inicia en los formularios y finaliza en las páginas servidoras?				
crítico	8. ¿Ha representado la relación de composición entre las páginas clientes y los formularios?				
	9. ¿El nombre de las páginas servidoras siguen la sintaxis sp_nombre?				
	10. ¿El nombre de las clases cliente siguen la sintaxis cp_nombre?				
	11. ¿El nombre de los formularios siguen la sintaxis fr_nombre?				
crítico	12. Si la página servidor				

	incluye otra página servidor ¿ha representado ésta relación como include?				
crítico	13. Si la página servidor redirecciona el procesamiento de datos a otra página ¿ha representado ésta relación como redirect?				
crítico	14. ¿Ha especificado en detalles los atributos de los formularios?				
crítico	15. (Cada atributo debe indicar si es de selección, de entrada o es un botón. Ejemplo: <<input >> nombre, <<selection>>provincia, <<button>>Ok)				
crítico	16. ¿Ha especificado las operaciones de las páginas cliente y páginas servidoras teniendo en cuenta las variables de su entorno?				
crítico	17. ¿Si las funciones de validación se hicieron en un fichero javascript ¿existe una relación de inclusión entre la página				

	cliente y dicho fichero?				
crítico	17.1 ¿El sentido de navegabilidad se inicia desde la página cliente al fichero?				
crítico	18. ¿De existir una relación entre la clase controladora y la clase interfaz ha representado tal relación como inclusión?				
crítico	18.1 ¿La navegabilidad se inicia en la clase controladora y finaliza en la clase interfaz?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				

crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Arquitectura de la Información

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Objetivos del Producto					
crítico	1. ¿Dan respuesta a qué se pretende lograr con el producto?				
	2. ¿Son frases iniciadas con verbos que indican las acciones a realizar?				
	3. ¿Son alcanzables con el desarrollo del producto?				

Audiencia del Producto					
	4. ¿Quiénes serán los usuarios del producto?				
	5. ¿Cuáles son los usuarios principales?				
crítico	6. ¿Cuáles son las características de la audiencia principal?				
crítico	7. ¿Cuáles son las necesidades de esa audiencia que se pretenden satisfacer con el producto?				
	8. ¿Cuáles son sus expectativas?				
Contenido del Producto					
crítico	9. ¿Están inventariados los contenidos a incluir en el producto?				
crítico	10. ¿Los contenidos del producto están agrupados y etiquetados? (la taxonomía)				
crítico	11. ¿El mapa de navegación representa las secciones, niveles y contenidos relacionados?				
	12. ¿El sistema de navegación es consistente,				

	uniforme y visible?				
crítico	13. ¿Se representa el diseño de la estructura de todas las pantallas tipo?				
	14. ¿Se identifican las diferentes áreas en la estructura de las pantallas?				
	15. ¿En los casos donde se realizan transacciones se incluye un diagrama de flujo sencillo que ejemplifica las posibles interacciones y sus resultados con las pantallas correspondientes?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
crítico	3. ¿Ha identificado errores				

	ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Diagrama de Despliegue

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Nodos					
	1. ¿Ha identificado los nodos y sus configuraciones de red?				
crítico	2. ¿Representa cada nodo un recurso de cómputo?				
	3. ¿Ha especificado claramente el nombre?				
crítico	4. De reflejar los nodos algunos componentes, ¿ha establecido las				

	relaciones entre ellos (componentes)?				
crítico	5. De realizar la descripción del nodo, ¿ha contemplado la capacidad de memoria y almacenamiento respectivamente u otras informaciones relacionadas con las capacidad?				
	6. ¿Existe una Aplicación Web? ¿Ha sido especificado?				
	7. ¿Existe una Base de Datos? ¿Ha sido especificada?				
Dispositivos					
	8. ¿Están representados como un ortoedro?				
	9. ¿Ha especificado claramente el nombre?				
crítico	10. ¿Representa un recurso de cómputo que no tiene capacidad de almacenamiento?				
Conectores					
	11. ¿Están los				

	dispositivos y los nodos correctamente conectados?				
	12. ¿Ha tenido en cuenta el protocolo de comunicación para establecer las conexiones entre ellos?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				
	5. ¿Está el documento				

	completo?				

Diccionario de Datos

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Todos los datos utilizados o producidos por el sistema han sido registrados y descritos en detalles?				
	2. De ser necesario, ¿ha incluido otro nombre del elemento de datos? (Alias)				
crítico	3. ¿Cada descripción del elemento de datos es clara y detallada?				
crítico	4. ¿Cada uno de los elementos o				

	actividades del sistema tiene un único significado?				
	5. ¿Han sido registrados todos los detalles adicionales relacionados al flujo de datos del sistema?				
crítico	6. ¿Ha especificado si el elemento de datos contiene valor numérico, caracteres o alfabético?				
crítico	7. ¿Ha especificado el máximo de caracteres o dígitos que puede tener un elemento de datos?				
	8. ¿Ha indicado cómo se presenta el dato al mostrarse en pantalla o al imprimirse en un reporte?				
	9. ¿Ha registrado que tipo de valor específico debe tener el elemento de datos?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos**	**Comentarios**

				afectados	
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Documento de Análisis

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					

Peso	Indicadores a Evaluar	Eval	(NP)	Cantidad de elementos afectados	Comentarios
Diagrama de clases de Análisis.					
Crítico	1. ¿El nombre de la clase es único?				
	2. ¿Ha enfocado las clases a las capacidades o condiciones del sistema?				
Crítico	3. ¿Ha identificado las responsabilidades (atributos y métodos) de las clases?				
	4. ¿Ha verificado que no exista más de una clase con la misma responsabilidad?				
Crítico	5. ¿Ha determinado las relaciones entre las clases?				
	6. ¿Los atributos definen				

	completamente al objeto en el contexto del problema?				
Crítico	7. ¿Ha detallado y especificado claramente el tipo de clase?				
	8. ¿Si existe dificultad en la comprensión de la clase de análisis debido a que esta tiene muchos atributos o los atributos son complejos ya separó algunos de esos atributos en clases independientes?				
Crítico	9. ¿Están las clases organizadas por paquetes, de existir complejidad en el sistema que se está tratando?				

	10. ¿Cada caso de uso tiene asociado un diagrama de clases del análisis?				
	11. ¿Ha aplicado patrones (GRASP o GoF) para el mejor entendimiento de las clases?				
Interacción entre las clases					
	1. ¿Ha identificado los actores que operan directamente con el sistema?				
	2. ¿Ha identificado la sucesión de los eventos mediante la descripción textual y detallada de los casos de uso?				
	3. ¿Están expresados los eventos a nivel de propósito del				

	objeto?				
Crítico	4. ¿Comienza el nombre de los eventos con un verbo?				
	5. En dependencia del número de escenarios de un caso de uso ¿ha realizado el diagrama de interacción de cada uno de estos escenarios?				
	6. ¿Ha colocado a la izquierda en el diagrama de interacción la instancia del objeto que inicia la interacción?				
Crítico	7. De haber empleado el diagrama de colaboración ¿todos los enlaces entre los objetos están acompañados de eventos?				

	8. ¿Las instancias de los objetos siguen el orden de clase interfaz, clase controladora, clase entidad?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Documento de Arquitectura del Software

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Vista de Casos de Uso					
crítico	1. ¿Se ha determinado cuáles son los actores y los casos de uso más significativos?				
crítico	2. Todas las relaciones entre los casos de uso (generalización-especialización, extensión e inclusión) ¿han sido especificada?				
crítico	3. ¿Todos los casos de uso son una traza directa con los requisitos funcionales?				
	4. ¿El nombre del caso de uso capta en esencia lo que el sistema debe llevar a cabo?				

crítico	5. ¿Todos los actores están al menos relacionados con un caso de uso?				
	6. ¿Es el nombre del caso de uso único?				
	7. ¿Ha elaborado una descripción textual de cada caso de uso?				
	7.1 ¿Se ha descrito en detalles cada escenario del caso de uso?				
crítico	8. ¿Todos los requisitos No funcionales (requisitos especiales) han sido descrito detalladamente?				
	9. ¿Es el modelo de casos de uso fácil de entender?				
	10. ¿Se ha empleado correctamente los patrones de casos de usos?				
Vista lógica					
crítico	11. ¿Se han identificado las clases de diseño más importante arquitectónicamente?				
crítico	12. ¿Todas las relaciones de asociación,				

	dependencia, generalización-especialización entre las clases han sido identificadas?				
crítico	13. ¿Se ha establecido la cardinalidad entre las clases?				
	14. ¿Se ha establecido la navegabilidad en la relación de asociación entre las clases?				
	15. ¿Las descripciones y el nombre de las clases se corresponden con el rol que juegan?				
crítico	16. ¿Se ha establecido la visibilidad de los atributos, las operaciones y los parámetros?				
	17. ¿Los elementos de los subsistemas están estrechamente asociados?				
	18. ¿Existe poca dependencia entre los subsistemas?				
	19. ¿Existe una traza directa entre los paquetes de análisis y los				

	requerimientos funcionales del sistema?				
	20. ¿Existe una traza directa entre los subsistemas de diseño y los paquetes de análisis?				
Interacción entre las clases					
	21. ¿Ha descrito la interacción entre las clases de análisis?				
	22. ¿Ha descrito la interacción entre las clases de diseño?				
crítico	23. ¿Ha identificado los actores que operan directamente con el sistema?				
crítico	24. ¿Están expresados los eventos a nivel de propósito del objeto?				
	25. ¿Comienza el nombre de los eventos con un verbo?				
	26. En dependencia del número de escenarios de un caso de uso ¿ha realizado el diagrama de interacción de cada uno de estos escenarios?				

Vista de Implementación					
crítico	27. ¿Han sido identificados todos los componentes arquitectónicamente significativos?				
crítico	28. ¿Todos los estereotipos de componentes han sido correctamente identificados?				
crítico	29. ¿Cada componente implementa la funcionalidad que se especifica en las clases de diseño?				
	30. ¿Son correctos los elementos que conforman los subsistemas de implementación?				
crítico	31. ¿Se ha asignado a los nodos de configuraciones de red componentes ejecutables?				
	32. ¿Se han identificados todos los componentes ejecutables?				
crítico	33. ¿Se ha establecido las relaciones de dependencia entre los componentes?				

crítico	34. ¿Se ha generado el código fuente de todas las clases de diseño que son arquitectónicamente relevantes?				
	35. ¿Ha valorado las relaciones de agregación y asociación en la generación del código fuente?				
crítico	36. ¿Todos los componentes implementados han sido sometidos a prueba de unidad?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores				

	ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Documento de Diseño-Desarrolladores

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Diagrama de clases de diseño					
	1. ¿Ha identificado las clases de análisis más relevantes arquitectónicamente?				
Crítico	2. ¿Ha especificado la visibilidad de los atributos, parámetros y las operaciones?				
	3. ¿Ha agregado a las clases de diseño todos los nombres de los métodos que aparecen en el diagrama de interacción de las clases de				

	análisis?				
Crítico	4. ¿Ha indicado las relaciones de asociación, dependencia, generalización-especialización entre las clases de diseño?				
Crítico	5. ¿Ha indicado la navegabilidad de las asociaciones entre las clases?				
Crítico	6. ¿Ha indicado la cardinalidad entre las clases?				
Crítico	7. ¿Son las clases de diseño una traza directa de las clases de análisis?				
	8. ¿Cada operación de la clase de diseño esta recogida en al menos una realización de caso de uso del análisis?				
	9. ¿Las clases de diseño contienen los eventos o mensajes de las realizaciones de caso de uso del análisis?				
Crítico	10. ¿De ser necesario ¿ha creado otras clases para representar clases innatas				

	del lenguaje de programación?				
Modelo de Despliegue					
	11. ¿Ha modelado la configuración de los elementos de procesamiento y las conexiones de estos elementos del sistema?				
Nodos					
	1. ¿Ha identificado los nodos y sus configuraciones de red?				
crítico	2. ¿Representa cada nodo un recurso de cómputo?				
	3. ¿Ha especificado claramente el nombre?				
crítico	4. De reflejar los nodos algunos componentes, ¿ha establecido las relaciones entre ellos (componentes)?				
crítico	5. ¿De realizar la descripción del nodo, ¿ha contemplado la capacidad de memoria y almacenamiento respectivamente u otras informaciones relacionadas con las capacidad?				
	6. ¿Existe una Aplicación Web? ¿Ha sido				

	especificado?				
	7. ¿Existe una Base de Datos? ¿Ha sido especificada?				
Dispositivo					
	8. ¿Están representados como un ortoedro?				
	9. ¿Ha especificado claramente el nombre?				
crítico	10. ¿Representa un recurso de cómputo que no tiene capacidad de almacenamiento?				
Conectores					
	11. ¿Están los dispositivos y los nodos correctamente conectados?				
	12. ¿Ha tenido en cuenta el protocolo de comunicación para establecer las conexiones entre ellos?				
Interacción entre las clases					
	12. ¿Ha descrito la interacción entre las clases de diseño?				
	13. ¿Están expresados los eventos a nivel de propósito				

	del objeto?				
	14. ¿Comienza el nombre de los eventos con un verbo?				
	15. ¿Los diagramas de interacción de las clases de diseño son una traza directa con los diagramas de clases de análisis?				
	16. ¿Ha colocado a la izquierda del diagrama la instancia del objeto que inicia la interacción?				
	17. De haber empleado el diagrama de colaboración ¿todos los enlaces entre los objetos están acompañados de eventos?				
	18. ¿Las instancias de los objetos siguen el orden de clase interfaz, clase controladora, clase entidad?				
Aspectos para diseñar una base de datos					
Crítico	19. Tras examinar las clases del análisis ha determinado ¿cuáles clases de diseño serán persistentes?				
	19.1. ¿Ha excluido las clases de control?				

	20.En las clases de generalización-especialización si la clase hija no hereda todo lo definido por el padre ¿ha incluido una nueva clase que describa lo común de ambas clases? (Refinar clases)				
	21.De existir clases con comportamientos y atributos comunes ¿ha definido para cada clase análoga una nueva clase con lo común de éstas?(Refinar clases)				
Aspectos para construcción del diagrama de clases persistente					
Crítico	22. ¿Ha realizado un diagrama con todas las clases persistentes?				
Crítico	23.¿Ha indicado en cada clase cuál es el atributo que identifica la clase ? (llave)				
Crítico	24. ¿Ha establecido la relaciones entres las clases persistentes?				
	25. ¿Ha revisado la relación de generalización-especialización?				
	25.1 ¿Ha indicado el atributo				

	del padre y el valor que toma cuando éste se especializa en cada hija?				
	25.2 De no existir el atributo en la clase padre ¿ha añadido dicho atributo en la clase persistente?				
	26. ¿Ha indicado si la herencia de las clases hijas (herencias que derivan de la clase padre) es total o con o sin solapamiento?				
	27. ¿Ha indicado la cardinalidad entre las clases compuestas? ¿Ha tenido presente los siguientes criterios?: • En el extremo de la clase compuesta como máximo la cardinalidad es 1. Para la agregación puede ser mayor que 1. • En el extremo de la clase componente la cardinalidad mínima por defecto es 1, la máxima depende de: i. Si no es agrupación es 1. ii. Si es una agrupación				

	es m. iii. Si la clase compuesta tiene más de un atributo de igual tipo la relación es con una clase y con la cardinalidad máxima igual a la cantidad de atributos iguales.				
	28. ¿Ha indicado la cardinalidad de entre las clases complejas?				
	28.1. ¿Ha tenido presente los siguientes criterios?: • En el extremo de la clase compleja la cardinalidad mínima debe ser 0 ó 1. • En el extremo de la clase componente la cardinalidad mínima por defecto es 1, la máxima depende de: i. Si no es agrupación es 1. ii. Si es una agrupación es m. iii. Si la clase compleja tiene más de un atributo de				

	igual tipo la relación es con una clase y con la cardinalidad máxima igual a la cantidad de atributos iguales.				
Crítico	29. ¿Ha generado el modelo de datos partiendo del diagrama de clases persistentes?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Documento de Diseño- Calidad

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Clases					
	1. ¿El nombre de la clase de diseño se corresponden con la descripción de la clase?				
Crítico	2. ¿Se ha indicado la visibilidad de los atributos, parámetros y operaciones?				
	3. ¿Las operaciones, atributos, parámetros y tipos de datos se han especificado utilizando un lenguaje de programación?				
Crítico	4. ¿Los estereotipos de las clases de diseño se corresponden con el lenguaje de programación?				
	5. ¿Las clases de diseño contienen los requisitos funcionales y no funcionales del				

	sistema?				
	6. ¿Se ha establecido las relaciones de generalización-especialización, asociación y dependencia entre todas las clases de diseños?				
	7. ¿Se ha indicado la navegabilidad en la asociación de las clases?				
Modelo de despliegue					
	8. ¿Se ha modelado la configuración de los elementos de procesamiento y las conexiones de estos elementos del sistema?				
Nodos					
	1. ¿Ha identificado los nodos y sus configuraciones de red?				
crítico	2. ¿Representa cada nodo un recurso de cómputo?				
	3. ¿Ha especificado claramente el nombre?				
crítico	4. De reflejar los nodos algunos componentes, ¿ha establecido las relaciones entre ellos (componentes)?				
crítico	5. De realizar la descripción del nodo, ¿ha contemplado la capacidad de memoria y				

	almacenamiento respectivamente u otras informaciones relacionadas con las capacidad?				
	6. ¿Existe una Aplicación Web? ¿Ha sido especificado?				
	7. ¿Existe una Base de Datos? ¿Ha sido especificada?				
Dispositivo					
	8. ¿Están representados como un ortoedro?				
	9. ¿Ha especificado claramente el nombre?				
crítico	10. ¿Representa un recurso de cómputo que no tiene capacidad de almacenamiento?				
Conectores					
	11. ¿Están los dispositivos y los nodos correctamente conectados?				
	12. ¿Ha tenido en cuenta el protocolo de comunicación para establecer las conexiones entre ellos?				
Operaciones					
	9. ¿El nombre de cada operación de las clases de diseño es descriptible y				

	“legible”?				
Interacción entre las clases					
	10. ¿Se ha representado la interacción entre las clases?				
	10.1. ¿Es fácil de entender y leer tal representación?				
	11. ¿Los objetos instanciados en el diagrama de interacción siguen el orden de clase interfaz, clase controladora y clase entidad?				
	12. ¿Ha revisado en el diagrama de interacción que la clase entidad no emita ningún mensaje?				
	13. ¿Ha revisado en el diagrama de interacción que la clase interfaz no emita ningún mensaje?				
Clases Persistentes					
Crítico	14. ¿Se ha identificado correctamente las clases persistentes?				
	15. ¿Se ha elaborado un diagrama de clases persistentes?				
	16. ¿Se ha representado las relaciones entre las clases				

	persistentes?				
	17.Se ha indicado cuál es el atributo que identifica a la clase ? (llave)				
Cardinalidad					
	18. ¿Se ha especificado la cardinalidad de las clases?				
	18.1. ¿Se ha tenido en cuenta los criterios elementales de cardinalidad?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Documento de Especificación de Requisitos del Software-Desarrolladores

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Ha identificado a la (s) persona(s) que lo ayudará a especificar los requisitos?				
	2. ¿Ha solicitado la participación de diferentes personas para poder definir los requisitos en diferentes puntos de vista?				
	2.1 ¿Se ha asegurado de capturar lo esencial de cada requisito registrado?				
	3. ¿Están todos los requisitos escritos en el lenguaje del usuario?				
	3.1 ¿Piensan eso los usuarios?				
	4. ¿Ha definido el flujo de información de manera				

	adecuada para el problema de dominio? (entrevistas, encuestas, equipos de discusión, grupos de trabajo)				
crítico	5. ¿Ha identificado todas las funciones que el usuario debe hacer?				
	6. ¿Ha definido los límites del sistema?				
crítico	7. ¿Todos los requisitos identificados se centran en lo que el sistema debe hacer y no como el sistema debe hacerlo?				
crítico	8. ¿Ha referenciado los requisitos incluso los que se derivan de otros requisitos?				
crítico	9. ¿Cada requisito es verificable? (Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina pueda chequear que el software satisface dicho requerimiento).				
	10. ¿Los criterios de validación se han declarado detalladamente?				

	10.1 ¿Son los criterios suficientemente adecuados para declarar un sistema exitoso?				
Crítico	11. ¿Han sido definidos todos los datos de entrada y salida?				
	12. ¿Cada funcionalidad del sistema ha sido representada gráficamente?				
crítico	13. ¿Ha identificado los requerimientos de software y de hardware?				
crítico	14. ¿Han sido identificadas las restricciones de diseño e implementación?				
crítico	15. ¿Han sido identificadas las restricciones de interfaz externa?				
crítico	16. ¿Lo requerimientos de soporte y usabilidad se han identificados?				
crítico	17. ¿Ha identificado los requerimientos de seguridad (con fidelidad, integridad, disponibilidad)?				
	18. ¿Ha definido que requisitos serán incluidos				

	en cada iteración del desarrollo del sistema?				
	19.1 ¿Ha tenido en cuenta el criterio del equipo de proyecto y la importancia que representa el requisito para el cliente?				
crítico	19. ¿Todos los cambios en los requisitos han sido controlados?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Documento de Especificación de Requisitos del Software-Calidad

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Están todos los requisitos escritos en el lenguaje del usuario?				
crítico	2. ¿Están los requisitos a un nivel bastante consistente?				
	2.1 ¿Debería especificarse algún requisito con más detalle?				
	2.2 ¿Debería especificarse algún requisito con menos detalles?				
crítico	3. ¿Son los requisitos lo suficientemente claros para poderse enviar a un grupo independiente para su implementación y que lo entiendan?				
crítico	4. ¿Cada requisito establece lo				

	que el sistema debe cumplir?				
	5. ¿Se ha utilizado diagramas para un mejor entendimiento?				
	6. ¿Los requisitos, incluyen aspectos relacionados con la funcionalidad, rendimiento de las limitaciones de diseño, atributos (atributos de datos y estructura de datos), interfaces o externa?				
crítico	7. ¿Han sido abordadas e identificadas los valores de entradas y salidas?				
	8. ¿Han sido incluidos las respuestas válidas y no válidas de los valores de entrada?				
	9. ¿Existe correspondencia entre el modelo de caso de uso, las Especificaciones Suplementarias y las especificaciones de requerimientos?				
	10. ¿Las figuras, tablas y diagramas incluyen plenamente las etiquetas y las referencias y las definiciones de todos los términos y unidades de medida?				
	11. ¿Cada requisito ha sido				

	identificado para indicar ya sea la importancia (proyecto y/o cliente) o la estabilidad de ese requisito?				
	12. ¿Se puede verificar cada requisito?				
	13. ¿Es cada requisito relevante al problema y a su solución?				
	14. ¿Se puede trazar cada uno al origen en el entorno del problema?				
	15. ¿Se han especificado todos los posibles cambios en los requisitos, incluyendo la probabilidad de cambio?				
	16. No aparece un mismo requisito en más de un lugar del documento de especificación.				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				

	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Modelo Entidad Relación- Desarrolladores

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Entidades Débiles					
	1. ¿Has representado las entidades débiles mediante un rectángulo dibujado con una línea doble?				
	2. ¿La relación entre una entidad débil y otra entidad es mediante un rombo dibujado con líneas dobles?				
crítico	3. ¿Sean identificado como identidades débiles aquellas que tienen una dependencia de existencia de otra entidad?				
Aspectos Generales					
	1. ¿Ha determinado qué información debe almacenarse?				

	2. ¿Has identificado todos los objetos?				
	3. ¿Son los objetos una traza directa a las clases de diseño?				
	¿Ha especificado cada uno de los atributos que contiene el objeto?				
crítico	4. ¿Has identificado la relación de dependencia entre entidades?				
	5. ¿Las asociaciones recursivas (a una misma entidad) tienen bien definida la cardinalidad de la misma?				
	6. ¿La cardinalidad entre las relaciones ternarias son de muchos a muchos?				
	7. ¿Ha escrito la cardinalidad entre paréntesis?				
	8. ¿Está bien representada la herencia que existe en relaciones de generalización-especialización?				
	9. ¿Sea representado la agregación como un rectángulo englobando a la interrelación que la conforma?				

	10. ¿Todas las relaciones de agregaciones están compuestas por más de una entidad?				
	11. ¿Las relaciones ternarias están compuesta por tres entidades?				
crítico	12. ¿Todas las entidades tienen su identificador (llave)?				
	13. ¿Todas entidades del MER están relacionadas mediante una línea?				
	14. ¿Todas las llave de las entidades están subrayados o con un fondo en negro?				
	15. ¿Los atributos multivaluados (multivaluados son los que tienen más de un valor para una misma persona o entidad un ejemplo en el caso de persona, cuando tiene muchos teléfonos)están bien representados(estés e representa con un ovalo grueso y oscuro)?				
crítico	16. ¿Están bien definidas todas las asociaciones recursivas?				
	17. ¿Están bien representadas				

	todas las especializaciones?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Modelo Entidad Relación- Calidad

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Entidades Débiles					
	4. ¿Has representado las entidades débiles mediante un rectángulo dibujado con una línea doble?				
	5. ¿La relación entre una entidad débil y otra entidad es mediante un rombo dibujado con líneas dobles?				
crítico	6. ¿Sean identificado como identidades débiles aquellas que tienen una dependencia de existencia de otra entidad?				
Aspectos Generales					
crítico	18. ¿Has identificado la relación de dependencia entre entidades?				
	19. ¿Las asociaciones recursivas (a una misma entidad) tienen				

	bien definida la cardinalidad de la misma?				
	20. ¿La cardinalidad entre las relaciones ternarias son de muchos a muchos?				
	21. ¿Ha escrito la cardinalidad entre paréntesis?				
	22. ¿Esta bien representada la herencia que existe en relaciones de generalización-especialización?				
	23. ¿Sea representado la agregación como un rectángulo englobando a la interrelación que la conforma?				
	24. ¿Todas las relaciones de agregaciones están compuestas por más de una entidad?				
	25. ¿Las relaciones ternarias están compuesta por tres entidades?				
crítico	26. ¿Todas las entidades tienen su identificador (llave)?				
	27. ¿Todas entidades del MER están relacionadas mediante una línea?				
	28. ¿Todas las llave de las				

	entidades están subrayados o con un fondo en negro?				
	29. ¿Los atributos multivaluados (multivaluados son los que tienen mas de un valor para una misma persona o entidad un ejemplo en el caso de persona, cuando tiene muchos teléfonos) están bien representados (estés e representa con un ovalo grueso y oscuro)?				
crítico	30. ¿Están bien definidas todas las asociaciones recursivas?				
	31. ¿Están bien representadas todas las especializaciones?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				

Crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Especificación de Casos de Uso-Desarrolladores

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Aspectos Generales					
	1. ¿Ha identificado cuáles son las tareas y responsabilidades de cada actor del sistema?				
	2. ¿Se ha especificado todos lo requisitos no funcionales del sistema?				
	3. ¿Ha identificado si algún actor creará, almacenará, modificará o borrará información del sistema? ¿Ha especificado que casos de uso harán estas funciones?				

	4. ¿Se ha especificado si es necesario que un actor le informe al sistema sobre los cambios externos?				
	5. ¿Se ha identificado los casos de uso que darán soporte y mantenimiento al sistema?				
	6. ¿Ha discutido con los futuros usuarios la propuesta de los casos de uso?				
	7. ¿Ha descrito con precisión todas las alternativas o excepciones?				
	8. ¿Ha capturado en detalles lo que desea el futuro usuario del sistema?				
	9. ¿Ha clasificado los casos de uso que definen la arquitectura básica del sistema?				
	10. ¿Ha clasificado los casos de uso que sirven de apoyo a los caso de uso que cubren las principales funciones que el sistema debe realizar?				
	11. ¿Ha clasificado los casos de uso que no son claves para la arquitectura?				
	12. ¿Cada caso de uso registra claramente lo que el sistema debe hacer?				
Nombre del Caso de Uso					

crítico	1. ¿Está en infinitivo y refleja de manera clara el objetivo del usuario sobre el sistema?				
crítico	2. ¿Es nombre del caso de uso único?				
	3. ¿Es nombre del caso de uso intuitivo?				
	4. ¿Es nombre del caso de uso explicativo?				
	5. ¿El nombre del caso de uso capta en esencia lo que el sistema debe llevar a cabo?				
Descripción					
	1. ¿El resumen dice como se inicia las operaciones principales que realiza el sistema?				
Precondición					
	1. ¿Se escribe una precondición si y solo si a partir de la ocurrencia de un suceso determinado comienza el caso de uso?				
	2. ¿La precondición es válida tanto para flujos básicos como flujos alternativos?				
Poscondición					
	1. ¿La poscondición plasma cambios que suceden en el sistema al terminarse de ejecutar el caso de uso?				
Complejidad del CU					
	1. ¿Se especificarse la complejidad del caso de uso?				

Forma de presentar la información					
crítico	1. ¿Está descrito el Caso de uso en presente?				
crítico	2. ¿Se describe de manera comprensible y detallada las acciones del actor frente al sistema? ¿Esta lo más parecido a un manual de ayuda?				
crítico	3. ¿Está correctamente reflejadas las inclusiones y extensiones del caso de uso?				
	4. ¿Se describe el momento exacto cuando se imprime o exporta a .pdf y qué entidad? ¿En caso de ser en cualquier momento se coloca como un flujo alterno?				
Actores del CU					
crítico	1. ¿El Caso de Uso está relacionado con al menos un actor?				
crítico	2. ¿Si hay dos actores interactuando con el caso de uso está generalizado en uno solo?				
crítico	3. ¿Si el caso de uso es abstracto (include, extend, generalización-especialización), no lo inicializa ningún actor?				
Flujo Básico					
	1. ¿Comienza diciendo “El caso de uso se inicia cuando el actor…”? (Esto puede variar en cada proyecto)				

	2. ¿Termina diciendo en un evento independiente “El caso de uso termina”?				
	3. ¿No existen abreviaturas?				
crítico	4. ¿Las partes del flujo de eventos que se repiten en otro caso de uso se especifican como un Caso de Uso incluido?				
crítico	5. Si las alternativas que se describen casi nunca ocurren o son alternativas comunes a otros casos de uso se especifican como un Caso de Uso extendido.				
Flujo Alterno					
crítico	1. ¿Los eventos que ocurren en cualquier momento están ubicados antes de los demás eventos?				
	2. ¿La condición refleja el evento inmediato que lo condiciona? Ej.: Debería ser “Cierre de sesión por el usuario” o “El usuario cierra la sesión” y no “El usuario desea cerrar la sesión”.				
crítico	3. ¿En todos los CU que se introducen datos tienen un flujo alterno donde el sistema Valide la integridad de los datos que se introducen y muestra un mensaje en caso de que los datos estén incompletos?				
	4. ¿Los flujos alternativos se nombran con el número del paso que lo generó en el flujo básico, una letra, ordenados alfabéticamente, y la condición que lo produjo? (Esto puede variar en cada proyecto)				

crítico	5. ¿En la sección flujos alternativos se describen todas las excepciones que existan por muy evidentes que parezcan?				
Casos de uso incluidos y extendidos					
crítico	1. ¿Se mencionan todos los casos de Uso Incluidos en el Caso de Uso?				
crítico	2. ¿Se mencionan todos los casos de Uso que Extienden en el Caso de Uso?				
crítico	3. ¿La descripción de los Casos de Uso incluidos y extendidos se realiza aparte?				
Reglas del negocio					
	1. ¿Si los datos cumplen alguna regla determinada están reflejadas en el documento de reglas del negocio ó hacen una referencia en el caso de uso a dicho documento ó se especifica en el caso de uso?				
Navegabilidad					
crítico	1. ¿La navegabilidad en los caso de uso de inclusión se inicia desde el caso uso base hasta el caso de uso incluido?				
crítico	2. ¿La navegabilidad en los caso de uso de extensión se inicia desde el caso uso extendido hasta el caso de uso base?				
Relaciones					
crítico	1. ¿Las relaciones de inclusión y extensión entre los caso de uso se ha representado con				

	línea discontinua?				
Información General					
crítico	1. ¿El diagrama de casos de uso expresa en detalles y claramente lo que debe hacer el sistema?				
	2. De existir la generalización-especialización entre los caso de uso ¿se ha representado?				
	3. Si la modelación de las interacciones con el sistema es muy extensa ¿ha empleado los paquetes de caso de uso?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				

Especificación de Casos de Uso-Calidad

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Nombre del caso de uso					
crítico	¿Está en infinitivo y refleja de manera clara el objetivo del usuario sobre el sistema?				
crítico	¿Es nombre del caso de uso único?				
	¿Es nombre del caso de uso intuitivo?				
	¿Es nombre del caso de uso explicativo?				
	¿El nombre del caso de uso capta en esencia lo que el sistema debe llevar a cabo?				
Descripción					
	¿El resumen dice como se inicia las operaciones principales que realiza el sistema?				
Precondición					
	¿Se escribe una precondición si y solo si a partir de la ocurrencia de un suceso determinado comienza el caso de uso?				
	¿La precondición es válida tanto para flujos básicos como				

	flujos alternativos?				
Poscondición					
	¿La poscondición plasma cambios que suceden en el sistema al terminarse de ejecutar el caso de uso?				
Complejidad del CU					
	¿Se especificarse la complejidad del caso de uso?				
Forma de presentar la información					
crítico	¿Está descrito el Caso de uso en presente?				
crítico	¿Se describe de manera comprensible y detallada las acciones del actor frente al sistema? ¿Esta lo más parecido a un manual de ayuda?				
crítico	¿Está correctamente reflejadas las inclusiones y extensiones del caso de uso?				
	¿Se describe el momento exacto cuando se imprime o exporta a .pdf y qué entidad? ¿En caso de ser en cualquier momento se coloca como un flujo alterno?				
Actores del CU					
crítico	¿El Caso de Uso está relacionado con al menos un actor?				
crítico	¿Si hay dos actores interactuando con el caso de uso está generalizado en uno solo?				
crítico	¿Si el caso de uso es abstracto (include, extend, generalización-especialización), no lo inicializa ningún actor?				
Flujo Básico					

	¿Comienza diciendo “El caso de uso se inicia cuando el actor…”? (Esto puede variar en cada proyecto)				
	¿Termina diciendo en un evento independiente “El caso de uso termina”?				
	¿No existen abreviaturas?				
crítico	¿Las partes del flujo de eventos que se repiten en otro caso de uso se especifican como un Caso de Uso incluido?				
crítico	Si las alternativas que se describen casi nunca ocurren o son alternativas comunes a otros casos de uso se especifican como un Caso de Uso extendido.				
Flujo Alterno					
crítico	¿Los eventos que ocurren en cualquier momento están ubicados antes de los demás eventos?				
	¿La condición refleja el evento inmediato que lo condiciona? Ej.: Debería ser “Cierre de sesión por el usuario” o “El usuario cierra la sesión” y no “El usuario desea cerrar la sesión”.				
crítico	¿En todos los CU que se introducen datos tienen un flujo alterno donde el sistema Valide la integridad de los datos que se introducen y muestra un mensaje en caso de que los datos estén incompletos?				
	¿Los flujos alternativos se nombran con el número del paso que lo generó en el flujo básico, una letra, ordenados alfabéticamente, y la condición que lo produjo? (Esto puede				

	variar en cada proyecto)				
crítico	¿En la sección flujos alternativos se describen todas las excepciones que existan por muy evidentes que parezcan?				
Casos de uso incluidos y extendidos					
crítico	¿Se mencionan todos los casos de Uso Incluidos en el Caso de Uso?				
crítico	¿Se mencionan todos los casos de Uso que Extienden en el Caso de Uso?				
crítico	¿La descripción de los Casos de Uso incluidos y extendidos se realiza aparte?				
Reglas del negocio					
	¿Si los datos cumplen alguna regla determinada están reflejadas en el documento de reglas del negocio ó hacen una referencia en el caso de uso a dicho documento ó se especifica en el caso de uso?				
Navegabilidad					
crítico	¿La navegabilidad en los caso de uso de inclusión se inicia desde el caso uso base hasta el caso de uso incluido?				
crítico	¿La navegabilidad en los caso de uso de extensión se inicia desde el caso uso extendido hasta el caso de uso base?				
Relaciones					
crítico	¿Las relaciones de inclusión y extensión entre los caso de uso se ha representado con línea discontinua?				
Información General					

crítico	¿El diagrama de casos de uso expresa en detalles y claramente lo que debe hacer el sistema?				
	De existir la generalización-especialización entre los caso de uso ¿se ha representado?				
	Si la modelación de las interacciones con el sistema es muy extensa ¿ha empleado los paquetes de caso de uso?				
	¿Ha identificado cuáles son las tareas y responsabilidades de cada actor del sistema?				
	¿Se ha especificado todos lo requisitos no funcionales del sistema?				
	¿Ha identificado si algún actor creará, almacenará, modificará o borrará información del sistema? ¿Ha especificado que casos de uso harán estas funciones?				
	¿Se ha especificado si es necesario que un actor le informe al sistema sobre los cambios externos?				
	¿Se ha identificado los casos de uso que darán soporte y mantenimiento al sistema?				
	¿Ha discutido con los futuros usuarios la propuesta de los casos de uso?				
	¿Ha descrito con precisión todas las alternativas o excepciones?				
	¿Ha capturado en detalles lo que desea el futuro usuario del sistema?				
	¿Ha clasificado los casos de uso que definen la arquitectura básica del sistema?				

	¿Ha clasificado los casos de uso que sirven de apoyo a los casos de uso que cubren las principales funciones que el sistema debe realizar?				
	¿Ha clasificado los casos de uso que no son claves para la arquitectura?				
	¿Cada caso de uso registra claramente lo que el sistema debe hacer?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto?				
	2. ¿Está organizado correctamente?				
	2.1 ¿Es conforme con la normativa aplicada?				
	2.2 ¿Contiene las secciones obligatorias?				
crítico	3. ¿Ha identificado errores ortográficos?				
	4. ¿Se entiende claramente lo que se ha especificado en el documento?				