UNIVERSIDAD CARLOS RAFAEL RODRIGUEZ.

Facultad de Informática.

Carrera de Ingeniería Informática.

Tesis presentada en opción al grado de Ingeniero en informática

Título: Guía para facilitar el aseguramiento de la calidad en el proceso de desarrollo de software en la facultad de informática de la UCF.

Autora:

Leticia Quesada Candelario.

Tutoras:

Ing. Anabel Martín Aparicio.

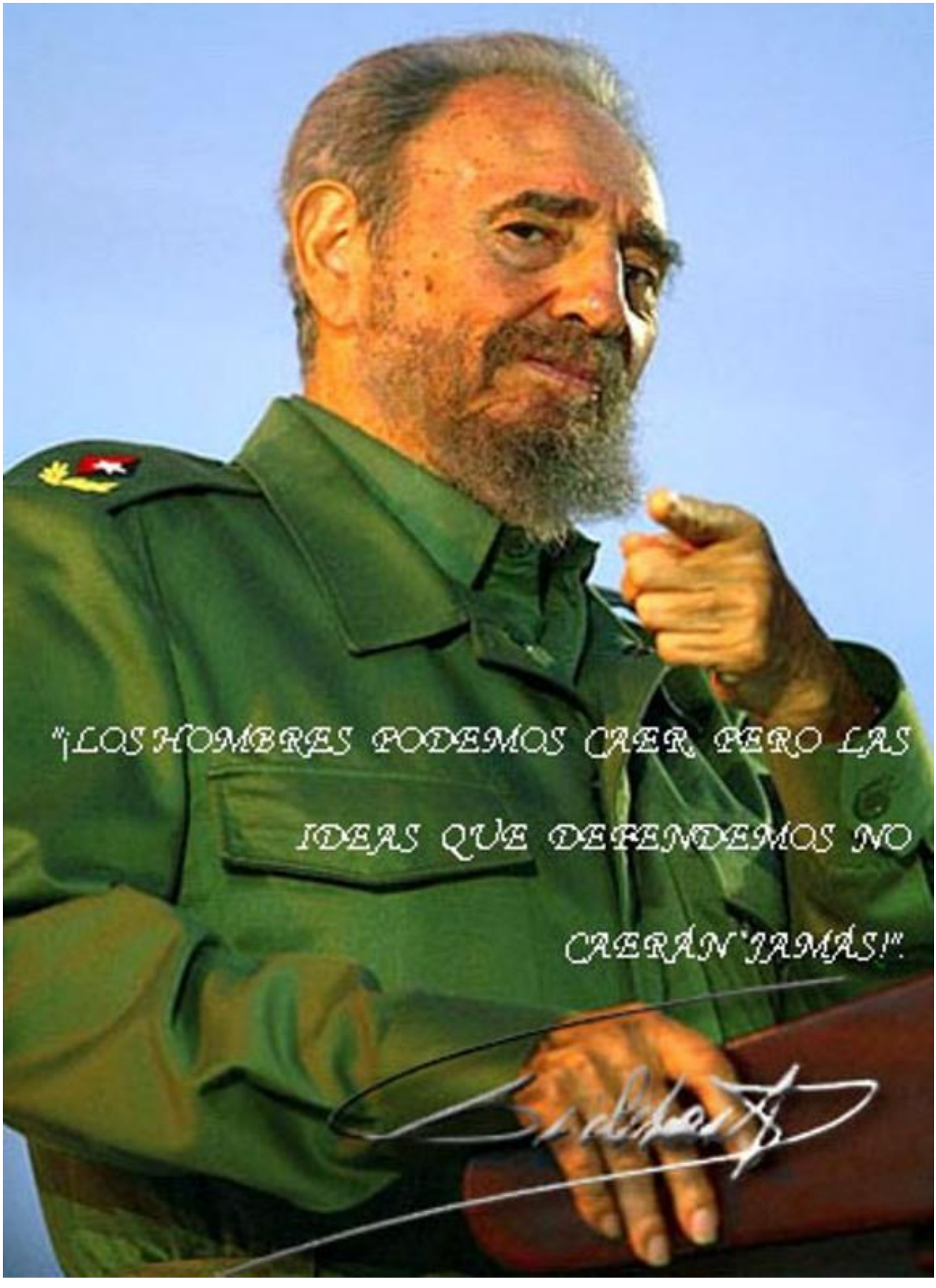
MSc. Daimarelys Acevedo Cardoso.

Consultante:

Ing. Yllen Sardiñas Torres.

Cienfuegos, Cuba

Curso 2009-2010.


"¡LOS HOMBRES PODEMOS CAER, PERO LAS
IDEAS QUE DEFENDEMOS NO
CAERÁN JAMÁS!".

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Departamento de Informática de la Facultad de Informática en la Universidad de Cienfuegos “Carlos Rafael Rodríguez”, para que hagan el uso que estimen pertinente con el trabajo de diploma.

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

_____
Nombre completo del autor

Los abajo firmantes certificamos que el presente trabajo ha sido revisado según acuerdo de la dirección de nuestro centro y el mismo cumple los requisitos que debe tener un trabajo de esta envergadura referente a la temática señalada.

---	---
Firma Tutor	Firma Tutor
---	---
Firma ICT	Firma Vicedecano

Agradecimientos

Agradezco a todas aquellas personas que han ayudado de una forma u otra con el desarrollo de este trabajo, las cuales saben de sobra quienes son. En especial a mi familia, mis padres, mis tutoras y mejores amigos.

Dedicatoria

Dedicado: A mis padres por estar siempre que los necesité.

A mi hermana por estar siempre a mi lado y brindarme su apoyo.

A mis dos grandes amigas Anabel e Yllen por estar siempre en las buenas y en las malas.

A mis tíos y a mi primo Ernestico por la ayuda y el apoyo que me brindaron.

A mis compañeros de aula por pasar tantos momentos juntos.

A mis familiares, amigos y a todos los que pusieron un granito de arena para que este trabajo fuese posible.

Resumen

En esta investigación se hizo un análisis exhaustivo para la elaboración de una guía que permita a cualquier desarrollador de software llevar el aseguramiento de la calidad de manera eficiente y eficaz. Además se realizó un estudio de las principales metodologías, modelos, herramientas y estándares utilizados en el proceso de desarrollo de software para escoger así las mejores técnicas a utilizar para la investigación. Se tuvo en cuenta para la validación de la misma el uso de encuestas, entrevistas y consultas a expertos para asegurar la excelencia de la guía. Por último se tomó una selección de la mejor bibliografía luego de haber hecho un análisis profundo de la misma. Quedando así conformada esta guía para garantizar que se cumpla con el aseguramiento de la calidad en cualquier proceso de desarrollo de software.

Summary

In this investigation had made an exhaustive analysis for elaboration a guide that permit anyone creator of software to secure the Quality of manner efficient and efficacious, beside it had made study the main methodologies, models, tools and standards using in the process of developing of software for choose the best technician using in de investigation. It had been account for the validation the use of inquiry, interview and expert's opinions to secure guide's Excellency. Finally it had been a selection of the best bibliography after a deep analysis of it. In that way it being create this guide to guarantee the quality assurance in any process of software's develop.

Índice

Índice de tablas

Índice de figuras

Introducción

El aseguramiento de la calidad comienza en la época de los años 50. Además del mejoramiento del proceso, se percibe la necesidad de asegurar el mejoramiento introducido. La calidad como estrategia competitiva comienza a utilizarse en la década de los 80. La administración redefine su papel con el propósito de que la calidad sea la estrategia a utilizar para tener éxito frente a los competidores.

A nivel mundial uno de los principales problemas con los que se encuentra la actividad de aseguramiento de la calidad en el software es la falta de apoyo por parte de la alta dirección de las organizaciones. Este apoyo es fundamental para que el aseguramiento de la calidad y el software en desarrollo tengan éxito. Los costos económicos de la función de aseguramiento de la calidad en el software se han estimado que varían entre un 2.5 y 5 porciento del costo total de un proyecto de desarrollo de un producto de software (Qualitrain, 2008).

El aseguramiento de la calidad de software es uno de los temas más actuales e importantes dentro de la ingeniería de software, y casualmente, es uno de los más olvidados en los programas de estudios universitarios y en el desarrollo empresarial. Cada vez más las organizaciones e instituciones de software están empezando a reconocer la importancia de contar con ingenieros de software especializados y capacitados en el área de calidad, y esencialmente con conocimientos en áreas tales como administración de la calidad del software, administración de procesos de software, administración de proyectos, métricas de software, administración de la configuración, y pruebas de software.

La función de aseguramiento de la calidad tiene como primera finalidad el determinar si las necesidades de los usuarios están siendo satisfechas adecuadamente, o sea, si se cumple con los requerimientos especificados por el cliente. Otra de sus funciones es la de determinar los costos que puede causar el añadir ciertas peculiaridades al producto, ya que a largo plazo, la economía resulta ser un factor decisivo para obtener un producto de calidad (Kynetia, 2007).

Actualmente, la responsabilidad del aseguramiento de calidad del software corresponde a ingenieros del software, gestores de proyectos, clientes, comerciales y grupos de SQA[1] (Software Quality Assurance). Los grupos de SQA son los encargados de velar por los intereses del cliente y asegurar que exista y se mantenga la calidad del software, por lo que se puede asegurar que las actividades de SQA están presentes en todo el proceso de desarrollo del software.

Actividades de SQA:

- Métodos, herramientas de análisis, diseño, programación y prueba.
- Inspecciones técnicas formales.
- Estrategias de prueba multiescala.
- Control de la documentación del software y de los cambios realizados.
- Procedimientos para ajustarse a los estándares (y dejar claro cuándo se está fuera de ellos).
- Mecanismos de medida (métricas).
- Registro de auditorías y realización de informes.

El aseguramiento de la calidad de software se ha convertido en una necesidad prioritaria para las organizaciones que desarrollan software, ya sea para uso interno o para implementaciones externas en clientes, porque cada vez más los errores en el software repercuten directa o indirectamente en graves consecuencias para la organización. Una de las empresas especializadas en el aseguramiento de la calidad de software es GreenSQA[2].

GreenSQA es un equipo humano con conocimiento y experiencia en la implementación de estándares, metodologías y modelos usados en la Industria del software a nivel mundial para las Pruebas y el Aseguramiento de Calidad tanto de los productos de software como del proceso productivo para su desarrollo.

[1] Aseguramiento de la calidad de software.

[2] Software Quality Assurance: Empresa especializada en el aseguramiento de la calidad de software.

Desde su inicio, a finales del año 2002, GreenSQA ha sido responsable de la implementación de la Estrategia de Calidad de ParqueSoft[3] Colombia, gracias a lo cual ha expuesto la metodología de trabajo a múltiples condiciones (tecnología, paradigmas de programación, infraestructura y comunicaciones, sectores de la economía, complejidad, tamaño de productos, equipos humanos) y adicionalmente ha recibido el beneficio de proyectos de cooperación nacional e internacional con Colciencias, el Banco Mundial y Carana, con el único propósito de hacer su servicio de clase mundial.

IBM[4] también es una de de las empresas que maneja el aseguramiento de la calidad y las herramientas de administración de la calidad del software que utiliza dan soporte a pruebas de unidades de software, a pruebas de sistema y al aseguramiento de la calidad del software a través de pruebas, ejecución, análisis de resultados y administración de QA (Aseguramiento de la Calidad). Ejemplos: Rational Functional Tester, Rational PurifyPlus, Web Sphere Studio Workload Simulator for z/OS and OS/390.

Las empresas cubanas en la actualidad no tienen mucho conocimiento con respecto al aseguramiento de la calidad. Se encuentran muchas empresas como SOFTEL[5], DESOFT[6], AVANTE[7], SOFCAL[8], CITMATEL[9] que han hecho algunos avances

3 Parque Tecnológico de Software: Empresa de base tecnológica especializada en la industria del conocimiento, a través de modelos de procesos de producción de productos y servicios basados en las mejores prácticas.

4 International Business Machines: es una empresa multinacional que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

5 Soluciones Informática: Empresa que proporcionar soluciones informáticas que eleven la eficiencia de los servicios de salud con personal y tecnologías de avanzada.

6 Empresa Nacional de Software: Empresa encargada de la automatización de la sociedad cubana en la elaboración de software.

7 Consultoría de Negocios del ministerio de la Informática y las comunicaciones en Cuba.

8 Centro de Desarrollo Informático en Cuba.

basándose en las normas ISO 9000 del 2000 pero les corresponde seguir perfeccionándose pues las exigencias a nivel mundial son cada vez mayores y la competencia juega un papel fundamental donde la satisfacción del cliente tiene una función excepcional (Desoft, 2008).

La producción de software en CITMATEL está compuesta por varias etapas o niveles, en primer lugar tienen el proceso de planificación del proyecto de software, luego cuentan con el proceso de contratación, proceso de diseño y desarrollo del producto, proceso de evaluación de la calidad, proceso de registro del producto, proceso de resguardo del producto y por último el proceso de reproducción y comercialización (Llerena, 2008).

CITMATEL realiza el proceso de evaluación de la calidad basándose en que debido a investigaciones realizadas han llegado a la conclusión de que el mayor número de errores se encuentra en el proceso de definición y requerimientos del sistema. Así como que también el mayor costo en la vida y operación de un producto informático está en la detección y corrección de errores. Por las razones antes expuestas esta entidad realiza las revisiones primero que todo por personal de las divisiones que realizan el software, ya que un error que es detectado en etapas tempranas de su ciclo de vida es más fácil de erradicar que si se detecta al final del mismo. Luego de esto se evalúa por el área de calidad según cuestionarios definidos en la metodología Qualysoft[10] en base a normas desarrolladas para la elaboración de productos de software y en base a las métricas definidas por la norma cubana ISO 9126-1:2004. De estas revisiones se extraen experiencias que se debaten en seminarios con los desarrolladores para futuros proyectos y para el mejoramiento de las metodologías utilizadas (Llerena, 2008).

En la universidad de Cienfuegos Carlos Rafael Rodríguez (UCF), específicamente en la facultad de informática se realizan software para diversas entidades de la

9 Empresa de Tecnologías de la Información y Servicios Telemáticos Avanzados: se distingue en el desarrollo científico y tecnológico de la sociedad cubana.

10 Sistema para la evaluación y certificación del software.

provincia, a dichas entidades solamente les es entregado el producto y no se verifica en ningún momento que se cumpla el SQA en el proceso de desarrollo del software, tampoco existe una posterior gestión de configuración con el sistema entregado. Además, los desarrolladores de software en general no se encuentran preparados para situar esta función dentro del proceso de desarrollo, ya sea por falta de conocimiento o por falta de personal que se dedique a las actividades de aseguramiento de la calidad. Los desarrolladores tienen el concepto erróneo de que con la programación del sistema están haciendo todo lo que se debe hacer, y tratan de evitar todo el proceso de documentación que implica el SQA.

Las razones expuestas anteriormente dan lugar a que el **problema científico** a solucionar sea: ¿Cómo introducir el aseguramiento de la calidad en el proceso de desarrollo del software dentro de la facultad de informática de la Universidad de Cienfuegos Carlos Rafael Rodríguez?

Para dar solución al problema científico se ubica el **objeto de estudio** el aseguramiento de la calidad del software.

Se define como **campo de acción** el aseguramiento de la calidad en el proceso de desarrollo del software.

Para el desarrollo de este trabajo y dar solución a la interrogante planteada, se trazó el siguiente **objetivo general**: Elaborar una guía para el aseguramiento de la calidad en el proceso de desarrollo de software en la facultad de informática de la Universidad de Cienfuegos Carlos Rafael Rodríguez.

Para dar cumplimiento al objetivo general planteado se han definido una serie de **tareas de investigación**:

- Realización de un estudio de las normas y estándares de calidad ISO 9126, 15 504.
- Realización de un estudio de los modelos CMM (“Capability Maturity Model”) y CMMI (Capability Maturity Model Integration).

- Realización de un estudio del estado del arte sobre las Metodologías de Desarrollo de Software, de forma tal que permita fundamentar la elección de la Metodología a utilizar.
- Análisis de las herramientas relacionadas con el aseguramiento de la calidad, para fundamentar la utilización de alguna de ellas.

Teniendo en cuenta toda la información antes planteada se define la siguiente **idea a defender:** la elaboración de una guía para el aseguramiento de la calidad en el proceso de desarrollo de software posibilitará que en la facultad de informática de la UCF se obtengan productos con mayor calidad y menor tiempo de entrega.

Aporte:

- Esta investigación de forma teórica aporta una guía para facilitar el aseguramiento de la calidad en el proceso de desarrollo de software en la facultad de informática de la Universidad de Cienfuegos Carlos Rafael Rodríguez.
- Como aporte práctico servirá para facilitar el trabajo de los desarrolladores de software y permitirá que se lleve el aseguramiento de la calidad como es debido. También brindará los argumentos teóricos y prácticos que la fundamentan.

La investigación realizada está estructurada por los siguientes capítulos:

Capítulo 1: Fundamentación Teórica. En este capítulo se realiza un estudio sobre el aseguramiento de la calidad de software, herramientas que utilizan, rol y modelos que la contienen. Se estudian diferentes metodologías, la gestión de la configuración y el proceso de V&V (verificación y validación) dentro del proceso de desarrollo de software.

Capítulo 2: Guía de aseguramiento de la calidad de software para la facultad de Informática (SQAFINF). En este capítulo se analizan las actividades de SQA, gestión de la configuración y V&V, dándole paso al desarrollo de la guía SQAFINF mediante una tabla resumen con estas actividades.

Capítulo 3: Proceso de evaluación del contenido de la guía SQAFINF. En este capítulo se analizarán los resultados obtenidos de las encuestas realizadas a especialistas, procesadas por el paquete estadístico SPSS y se podrá arribar a conclusiones del análisis de los mismos.

CAPÍTULO 1: Fundamentación Teórica.

1.1 Introducción.

En aras de incentivar el uso del aseguramiento de la calidad en el desarrollo del software en la facultad de informática de la Universidad de Cienfuegos, se abordarán temas relacionados con la misma. Se analizarán metodologías de desarrollo y se estudiarán los modelos CMM, CMMI y algunas de las normas y estándares, ISO 9126, 15 504, así como algunas herramientas que permitan la automatización del aseguramiento de la calidad a lo largo del desarrollo del software.

1.2 Aseguramiento de la calidad de software.

1.2.1 ¿Qué es la calidad?

La calidad es un aspecto a tener en cuenta en todas las organizaciones desarrolladoras de software, es el eslabón fundamental para que el software o producto final tenga las especificaciones requeridas por el cliente. El término calidad es definido por diversos autores y organizaciones ejemplo de ellos son los siguientes:

- Según Drae: calidad es la propiedad o conjunto de propiedades inherentes a una cosa, que permiten apreciarla con o igual, mejor o peor que las restantes de su especie (Carlos, 2007).
- Según Pressman: se afirma que la calidad es la “concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario” (Pressman, 2005).
- Totalidad de las características de un producto o servicio que le confieren su aptitud para satisfacer unas necesidades expresadas o implícitas, es el concepto brindado en la Norma UNE 66-001-92 traducción de ISO 8402 (Carlos, 2007).
- Grado en el que un conjunto de características inherentes cumple con los requisitos. Planteado por la ISO 9000:2000 (Carlos, 2007).

- Según Deming es la conformidad con requisitos y confiabilidad en el funcionamiento (Carlos, 2007).
- **[11]Calidad no es más que el conjunto de características con el que debe contar un producto para que sea aceptado por el cliente. Es decir el producto debe cumplir con todas las necesidades del mismo y no presentar errores.

1.2.2 ¿Qué es el aseguramiento de la calidad?

- Es el grupo de actividades planificadas y sistemáticas necesarias para brindar la confianza adecuada en que el software satisfará los requisitos establecidos de calidad. El aseguramiento pretende dar la confianza de que el producto tiene la calidad requerida (Qualitrain, 2008) (Lovelle, 2005).
- Algunos desarrolladores de software continúan creyendo que la calidad del software es algo en lo que empiezan a preocuparse una vez que se ha generado el código. ¡Nada más lejos de la realidad! La garantía de calidad del software (SQA software quality assurance, GCS gestión de calidad del software) es una actividad de protección que se aplica a lo largo de todo el proceso del software. La SQA engloba: 1)un enfoque de gestión de calidad; 2)tecnología de ingeniería del software efectiva(métodos y herramientas); 3)revisiones técnicas formales que se aplican durante el proceso de software; 4)una estrategia de prueba multiescalada; 5)el control de la documentación del software y de los cambios realizados; 6)un procedimiento que asegure un ajuste a los estándares de desarrollo del software(cuando sea posible),y 7)mecanismos de medición y de generación de informes (Pressman, 2005).
- **[12]SQA (Software Quality Assurance) no es más que la comprobación sistemática para que el producto o software sea entregado al cliente lo más fiel posible a como él lo solicitó. Un producto que no contenga errores, que sea amigable para el cliente y que cumpla con todos los requisitos establecidos.

[11] Concepto propio de calidad.

[12] Concepto propio aseguramiento de la calidad.

1.3 Modelos y normas que determinan el aseguramiento de la calidad.

Existen diversos modelos y normas que se pueden utilizar para determinar el aseguramiento de la calidad. Estos normalmente te indican qué hacer y no cómo hacerlo porque en realidad el cómo hacerlo depende del tipo de metodología a utilizar o de los objetivos del negocio que se realice. En este epígrafe se hará énfasis en los modelos ISO/IEC 9126, SPICE, CMM y CMMI. Donde se abordará a grandes rasgos sus principales características y sus niveles de capacidad y madurez.

1.3.1 ISO 9126

Los modelos de calidad basados en la ISO9126 (ISO/IEC 9126[13]), inicialmente se inspiraron en el modelo de Mc Call, que sugiere seis características para medir la calidad de productos de software y las desglosa en sub-características. Las continuas actualizaciones de la ISO/IEC 9126 agregan, por encima de las características, las fases del ciclo de vida del producto en el cual se mide la calidad: Proceso de Desarrollo, Calidad Interna, Calidad Externa y Calidad de Ejecución (Figueroa, 2006).

[13] ISO/IEC 9126: es un estándar internacional para la evaluación del Software.

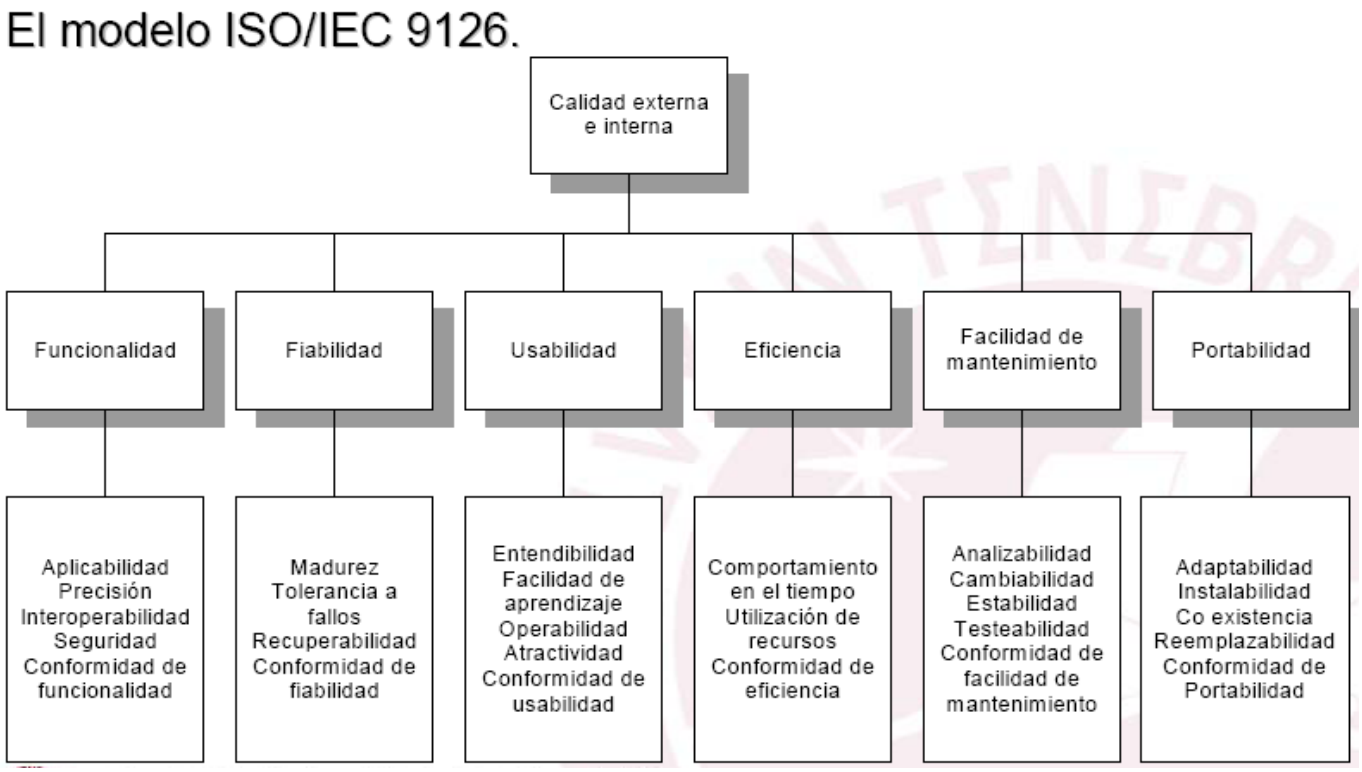


Figura 1. Modelo ISO/IEC 9126

A continuación se especifican las características de la norma:

Fiabilidad: aquí se agrupan un conjunto de atributos que se refieren a la capacidad del software de mantener su nivel de ejecución bajo condiciones normales en un periodo de tiempo establecido. Las sub-características que el estándar sugiere son (Figueroa, 2006):

· Nivel de Madurez. Permite medir la frecuencia de falla por errores en el software.

· Tolerancia a fallas. Se refiere a la habilidad de mantener un nivel específico de funcionamiento en caso de fallas del software o de cometer infracciones de su interfaz específica.

· Recuperación. Se refiere a la capacidad de restablecer el nivel de operación y recobrar los datos que hayan sido afectados directamente por una falla, así como el tiempo y el esfuerzo necesarios para lograrlo.

Funcionalidad: en este grupo se agrupan una serie de atributos que permiten calificar si un producto de software maneja en forma adecuada el conjunto de

funciones que satisfagan las necesidades para las cuales fue diseñado. Para este propósito se establecen los siguientes atributos (Figueroa, 2006):

· Adecuación. Se enfoca a evaluar si el software cuenta con un conjunto de funciones apropiadas para efectuar las tareas que fueron especificadas en su definición.

· Exactitud. Este atributo permite evaluar si el software presenta resultados o efectos acordes a las necesidades para las cuales fue creado.

· Interoperabilidad. Permite evaluar la habilidad del software de interactuar con otros sistemas previamente especificados.

· Conformidad. Evalúa si el software se acerca a estándares, convenciones o regulaciones en leyes y prescripciones similares.

· Seguridad. Se refiere a la habilidad de prevenir el acceso no autorizado, ya sea accidental o premeditado, a los programas y datos.

Usabilidad: Consiste en un conjunto de atributos que permiten evaluar el esfuerzo necesario que deberá invertir el usuario para utilizar el sistema (Figueroa, 2006).

· Comprensibilidad. Se refiere al esfuerzo requerido por los usuarios para reconocer la estructura lógica del sistema y los conceptos relativos a la aplicación del software.

· Facilidad de Aprender. Establece atributos del software relativos al esfuerzo que los usuarios deben hacer para aprender a usar la aplicación.

· Operabilidad. Agrupa los conceptos que evalúan la operación y el control del sistema.

Eficiencia: esta característica permite evaluar la relación entre el nivel de funcionamiento del software y la cantidad de recursos usados. Los aspectos a evaluar son (Figueroa, 2006):

· Comportamiento con respecto al tiempo. Atributos del software relativos a los tiempos de respuesta y de procesamiento de los datos.

· Comportamiento con respecto a recursos. Atributos del software relativos a la cantidad de recursos usados y la duración de su uso en la realización de sus funciones.

Mantenibilidad: Se refiere a los atributos que permiten medir el esfuerzo necesario para realizar modificaciones al software, ya sea por la corrección de errores o por el incremento de funcionalidad. En este caso, se tienen los siguientes factores (Figueroa, 2006):

· Capacidad de análisis. Relativo al esfuerzo necesario para diagnosticar las deficiencias o causas de fallas, o para identificar las partes que deberán ser modificadas.

· Capacidad de modificación. Mide el esfuerzo necesario para modificar aspectos del software, remover fallas o adaptar el software para que funcione en un ambiente diferente.

· Estabilidad. Permite evaluar los riesgos de efectos inesperados debidos a las modificaciones realizadas al software.

· Facilidad de Prueba. Se refiere al esfuerzo necesario para validar el software una vez que fue modificado.

Portabilidad: en este caso, se refiere a la habilidad del software de ser transferido de un ambiente a otro, y considera los siguientes aspectos (Figueroa, 2006):

· Adaptabilidad. Evalúa la oportunidad para adaptar el software a diferentes ambientes sin necesidad de aplicarle modificaciones.

· Facilidad de Instalación. Es el esfuerzo necesario para instalar el software en un ambiente determinado.

· Conformidad. Permite evaluar si el software se adhiere a estándares o convenciones relativas

· Capacidad de reemplazo. Se refiere a la oportunidad y el esfuerzo usado en sustituir el software por otro producto con funciones similares.

1.3.2 SPICE

En 1992 ISO[14] creó el grupo de trabajo WG10 a quien le fue dada la misión de desarrollar el nuevo estándar internacional. Éste grupo realizó diversas acciones, una de ellas fue llevar a cabo una primera fase del desarrollo del nuevo estándar, a través de un proyecto internacional llamado ISO/SPICE[15] (Software Process Improvement and Capability dEtermination), con el objetivo de acelerar el desarrollo del estándar (digital, 2009).

ISO/IEC TR 15504 provee un marco de trabajo para la valoración del proceso de software y parte de los mínimos requerimientos para la elaboración de una valoración para asegurar consistencia y repetitividad de las mediciones. Los requerimientos ayudan a asegurar que las salidas de la valoración son auto consistentes, y provee evidencia para fortalecer las mediciones y para verificar conformidad con los requerimientos (digital, 2009).

El modelo ISO/SPICE se considera tridimensional

1-Dimensión es funcional (procesos).

2-Capacidades (niveles).

3-Adecuación o efectividad (calificaciones).

El modelo SPICE está basado en procesos que se agrupan en cinco categorías diferentes: Cliente-Proveedor, Ingeniería, Soporte, Administración y Organización.

Procesos primarios

- CUS: Cliente - Proveedor
- ENG: Ingeniería

Procesos de soporte

- SUP: Soporte

14 ISO: International Organization for Standardization.

15 ISO/SPICE: Software Process Improvement and Capability dEtermination. Estándar para medir la capacidad de madurez de las organizaciones desarrolladoras de software.

Procesos organizacionales

- MAN: Gestión
- ORG: Organización

La dimensión de Capacidad está organizada ordinalmente en niveles de capacidad, permitiendo que la capacidad de cada proceso se mida de forma independiente, y que se defina un camino para la mejora individual de cada proceso (digital, 2009).

Nivel 0: No realizada, no hay productos de trabajo identificables.

Nivel 1: realizada informalmente, planificación y seguimiento dependientes del conocimiento individual. Productos de trabajo identificables.

Nivel 2: planificada, verificada de acuerdo a los procedimientos especificados.

Nivel 3: bien definida, procesos bien definidos y documentados

Nivel 4: controlada cuantitativamente, medidas detalladas de realización, predicción, etc. Productos de trabajo evaluados cuantitativamente.

Nivel 5: mejorada continuamente, objetivos cuantitativos de eficiencia basados en los objetivos de negocio.

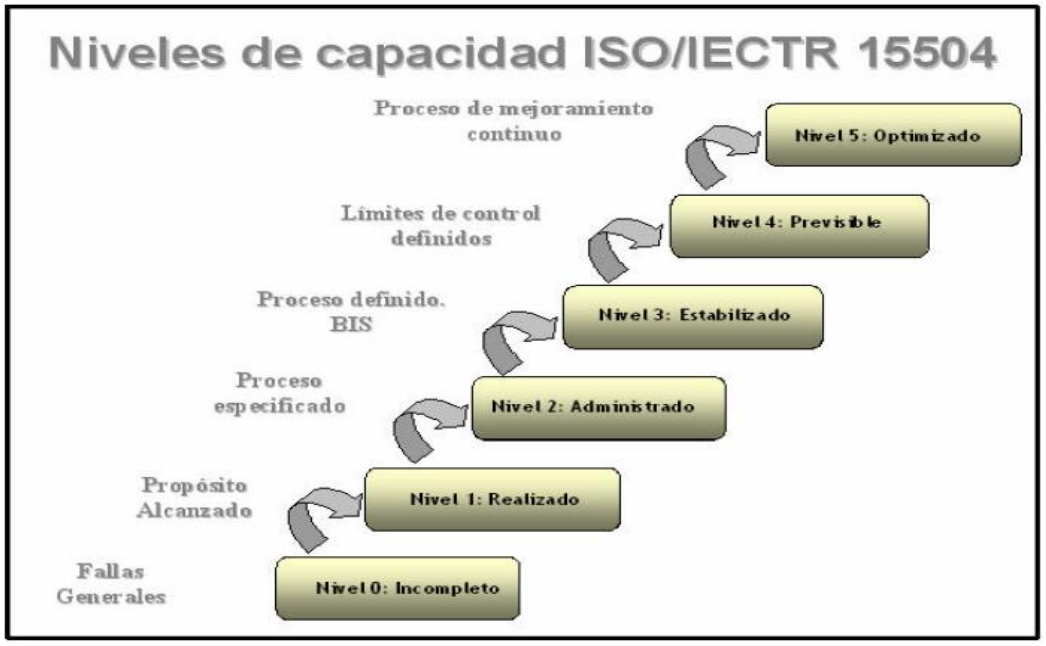


Figura 2. Modelo SPICE

La valoración del proceso es una actividad que es realizada durante una iniciativa de mejoramiento del proceso, como está descrita en ISO/IEC TR 15504-7, o como parte de un procedimiento de la determinación de capacidad, como esta descrito en ISO/IEC TR 15504-8. En ambos casos, la entrada formal del proceso de valoración define el propósito, el ámbito (proceso que es valorado), las restricciones y las responsabilidades de la valoración (digital, 2009).

Generalmente una valoración es llevada a cabo por un equipo, el cual puede utilizar o no herramientas de soporte. Dicha valoración es supervisada por un asesor competente que cuenta con la habilidad y destreza necesaria, tal como define ISO/IEC TR 15504-6 (digital, 2009).

1.3.3 CMMI y CMM

CMM significa “Capability Maturity Model[16]”, en español algo así como “Modelo de Capacidad de Madurez del Software”, se trata de un modelo de evaluación de los procesos de una organización. Fue desarrollado inicialmente para los procesos relativos al software por la Universidad Carnegie -Mellon para el SEI[17] (Software Engineering Institute).

El modelo entiende por organización inmadura aquella que lleva adelante sus proyectos sin una definición previa de los procesos a seguir. Estos proyectos habitualmente sobrepasan sus presupuestos y tiempos de terminación debido a que son iniciados con estimaciones poco realistas, sin una planificación adecuada, y son llevados adelante sin ninguna gestión. En general estos proyectos no terminan o terminan con una disminución importante en la calidad que se espera del producto (Pantaleo, y otros, 2009).

Por organizaciones maduras el modelo entiende a aquellas que desarrollan sus proyectos en forma planeada. El logro de los objetivos del proyecto es asignado al cumplimiento de las reglas preestablecidas. Los presupuestos asignados y el tiempo previsto son los necesarios ya que se parte de estimaciones metódicas y basadas en

16 CMM: Capability Maturity Model.

17 SEI: Software Engineering Institute

datos de proyectos previos, con roles y responsabilidades bien definidos (Pantaleo, y otros, 2009).

Este modelo define que deben existir algunas áreas o procesos claves en la organización que deberán realizar alguna función específica. A estas áreas se les denomina Áreas Claves de Procesos (KPA - Key Process Area). Además este modelo define para cada una de estas áreas un grupo de buenas prácticas, dependiendo de qué tanto puedan ajustarse estas áreas. Algunas de estas áreas son (Pantaleo, y otros, 2009):

- Compromiso para realizar (Co).
- Capacidad para realizar (Ab).
- Actividades realizadas (Ac).
- Medición y análisis (Me).
- Verificación de la implementación (Ve).

El CMM se centra en los tres principales aspectos que influyen en una organización:

a) Las personas: Se trata por disciplinas como el desarrollo organizativo, gestión de los RRHH y la Gestión de la Calidad Total (TQM).

b) La tecnología: La tecnología cambia a su propio ritmo a lo largo del tiempo, se puede adquirir.

c) El proceso: Pero, ¿cómo se gestiona el proceso y cómo se mejora?, ¿se puede comprar? La gestión del proceso se puede aprender e institucionalizar, aquí es donde entra a jugar su papel el CMM.

El modelo CMM está conformado por 5 niveles de madurez:

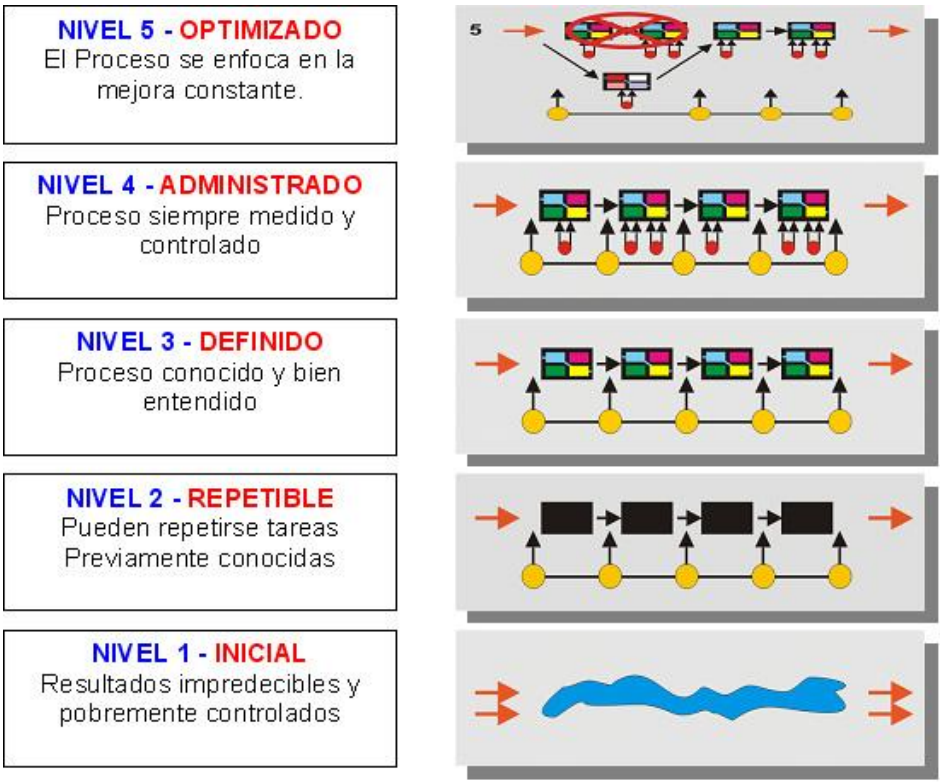


Figura 3. Modelo CMM (Bravo., 2008)

1 – Inicial: Es el primer nivel por lo que no es necesario hacer ningún esfuerzo para llegar a él, las organizaciones en este nivel no cuentan con un ambiente adecuado para el desarrollo de software. Aunque se utilicen técnicas correctas de ingeniería, los esfuerzos se ven cargados por falta de planificación. Los procesos varían según las personas, el éxito de los proyectos la mayoría de las veces se basa en el esfuerzo personal, aunque a menudo se producen fracasos y la mayoría de las veces retrasos y sobre costos. El resultado del producto final es impredecible y es poco controlado.

2 – Repetible: En este segundo nivel se encuentran las organizaciones en las que existe seguimiento y planificación de proyectos y está implementada la gestión de los mismos. Pero con todo y esto aún existe un gran riesgo de no cumplir las metas establecidas (Bravo, 2008).

3 – Definido: Existe un conjunto establecido de procesos estándar globales bien definidos (estableciendo sus objetivos) dentro de la organización. Tienen un sistema de gestión de los proyectos. Una diferencia crítica entre los niveles 2 y 3 de madurez

es el alcance de los estándares, descripciones de los procesos y procedimientos. En el nivel 2 pueden variar entre las distintas instancias de los procesos (entre diferentes proyectos); a nivel 3 son globales dentro de la organización e igual en todas las instancias de cada proceso (Bravo, 2008).

4 – Gestionado: Se identifica porque las organizaciones disponen de un conjunto de métricas importantes de calidad y productividad, que se usan de modo sistemático para la toma de decisiones y la gestión de riesgos. El software que resulta consta de una alta calidad (Bravo, 2008).

5 – Optimizado: La empresa completa está volcada en la mejora continua de los procesos. Se hace uso intensivo de las métricas y se gestiona el proceso de innovación (Bravo., 2008).

El modelo CMM y el modelo CMMI - Capability Maturity Model Integration se diferencian básicamente en que el primero se enfoca principalmente a las organizaciones o áreas de Tecnologías de información, en cambio, el modelo CMMI como su nombre lo indica es un modelo integrado y mejorado que se puede aplicar a un número mayor de organizaciones de diferentes sectores.

De Manera similar al CMM, el CMMI también maneja niveles:

Nivel de Capacidad 0 – Incompleto: área de proceso sin objetivos.

Nivel de Capacidad 1 – Ejecutada: objetivos específicos del área de proceso son satisfechos.

Nivel de Capacidad 2 – Administrada: área de proceso institucionalizada a partir de una política organizacional de uso de los procesos.

Nivel de Capacidad 3 – Definida: área de proceso institucionalizada a partir de un proceso definido.

Nivel de Capacidad 4 – Cuantitativamente Administrada: área de proceso institucionalizada a partir de una política organizacional de la administración cuantitativa de procesos.

Nivel de Capacidad 5 – Optimizada: área de proceso institucionalizada a partir de la optimización de sus procesos.

1.4 Metodologías.

La metodología utilizada en un desarrollo de software brinda la guía para poder conocer todo el camino a recorrer desde antes de empezar la implementación, con lo cual se asegura la calidad del producto final, así como también el cumplimiento en la entrega del mismo en un tiempo establecido. Es por ello, que a continuación se hace referencia a algunas de estas metodologías con el fin de conocer un poco más la utilización de las mismas.

1.4.1 RUP

Las siglas RUP[18] en ingles significa Rational Unified Process (Proceso Unificado de Desarrollo) es un producto del proceso de ingeniería de software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización del desarrollo. Su meta es asegurar la producción del software de alta calidad que resuelve las necesidades de los usuarios dentro de un presupuesto y tiempo establecidos (CHACÓN, marzo 2006) (Letelier, 2007).

El proceso puede ser descrito en dos dimensiones o ejes:

Eje horizontal: Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos. Se puede observar en la Figura 4 que RUP consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Como se mencionó anteriormente cada fase se subdivide a la vez en iteraciones (Kruchten, 2000).

Eje vertical: Representa los aspectos estáticos del proceso. Describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

18 Rational Unified Process es un proceso de desarrollo de software.

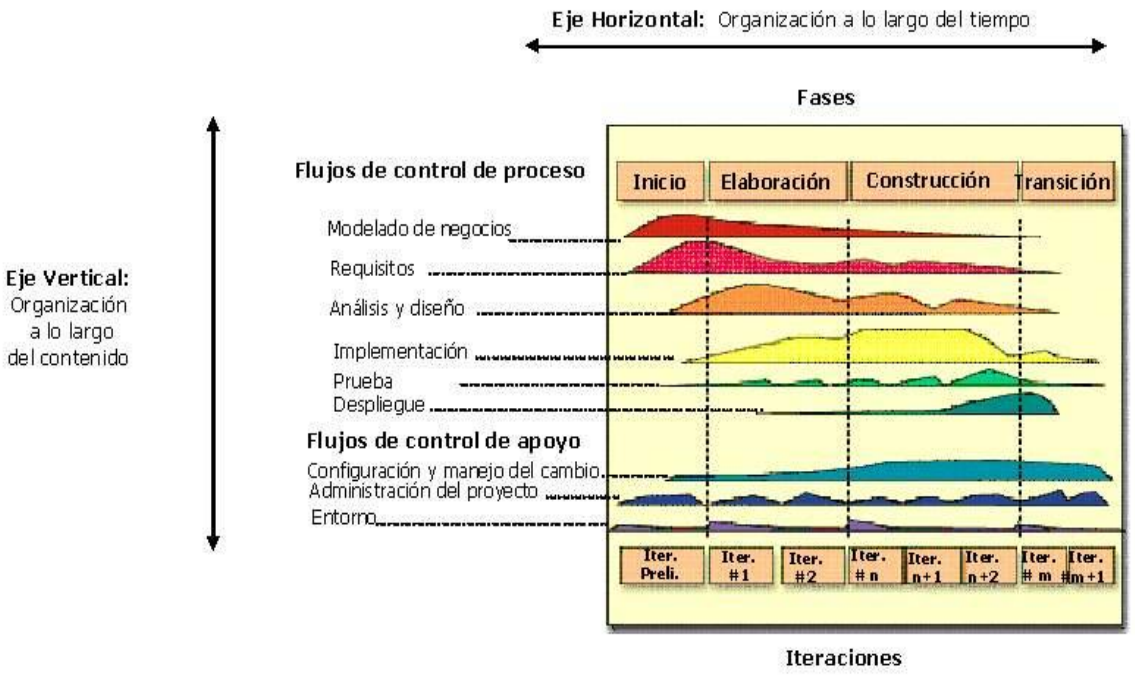


Figura 4. Fases e iteraciones de la Metodología RUP

Se recomienda que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta luego en un entregable al cliente. Trayendo como beneficio la retroalimentación en cada entregable o en cada iteración.

Flujos de trabajo

Con la enumeración de roles, actividades y artefactos no se define un proceso, se necesita contar con una secuencia de actividades realizadas por los diferentes roles, así como la relación entre los mismos. Un flujo de trabajo es una relación de actividades que produce resultados observables. A continuación se dará una explicación de cada flujo de trabajo (Letelier, 2007).

Modelado del negocio

Con este flujo de trabajo se pretende llegar a un mejor entendimiento de la organización donde se va a implantar el producto.

Los objetivos del modelado de negocio son:

- Entender la estructura y la dinámica de la organización para la cual el sistema va a ser desarrollado (organización objetivo).

- Entender el problema actual en la organización objetivo e identificar potenciales mejoras.
- Asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de la organización objetivo.
- Derivar los requisitos del sistema necesarios para apoyar a la organización objetivo.

Para lograr estos objetivos, el modelo de negocio describe cómo desarrollar una visión de la nueva organización, basado en esta visión se definen procesos, roles y responsabilidades de la organización por medio de un modelo de Casos de Uso del negocio y un Modelo de Objetos del Negocio. Complementario a estos modelos, se desarrollan otras especificaciones tales como un Glosario.

Requisitos

Este es uno de los flujos de trabajo más importantes, porque en él se establece qué tiene que hacer exactamente el sistema que se construya. En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen.

Los objetivos del flujo Requisitos son:

- Establecer y mantener un acuerdo entre clientes y otros *stakeholders*[19] sobre lo que el sistema podría hacer.
- Proveer a los desarrolladores un mejor entendimiento de los requisitos del sistema.
- Definir el ámbito del sistema.
- Proveer una base para la planeación de los contenidos técnicos de las iteraciones.
- Proveer una base para estimar costos y tiempo de desarrollo del sistema.
- Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.

19 Cualquier grupo o individuo que pueda afectar o ser afectado por el logro de objetivos de la organización.

Los requisitos se dividen en dos grupos. Los requisitos funcionales representan la funcionalidad del sistema. Se modelan mediante diagramas de Casos de Uso. Los requisitos no funcionales representan aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica. Por ejemplo requisitos de facilidad de uso, fiabilidad, eficiencia, portabilidad, etc.

Para capturar los requisitos es preciso entrevistar a todos los interesados en el proyecto, no sólo a los usuarios finales, y anotar todas sus peticiones. A partir de ellas hay que descubrir lo que necesitan y expresarlo en forma de requisitos.

En este flujo de trabajo, y como parte de los requisitos de facilidad de uso, se diseña la interfaz gráfica de usuario. Para ello habitualmente se construyen prototipos de la interfaz gráfica de usuario que se contrastan con el usuario final.

Análisis y Diseño

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema.

Los objetivos del análisis y diseño son:

- Transformar los requisitos al diseño del futuro sistema.
- Desarrollar una arquitectura para el sistema.
- Adaptar el diseño para que sea consistente con el entorno de implementación, diseñando para el rendimiento.

El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos.

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los Casos de Uso con las interacciones de las clases de análisis. Durante la fase de elaboración se va refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento

para diseñar componentes. Además si el sistema usará una base de datos, habrá que diseñarla también, obteniendo un modelo de datos.

El resultado final más importante de este flujo de trabajo será el modelo de diseño. Consiste en colaboraciones de clases, que pueden ser agregadas en paquetes y subsistemas.

Otro producto importante de este flujo es la documentación de la arquitectura de software, que captura varias vistas arquitectónicas del sistema.

Implementación

En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. Además se deben hacer las pruebas de unidad: cada implementador es responsable de probar las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable (Letelier, 2007) (Kruchten, 2000).

En cada iteración habrá que hacer lo siguiente:

- Planificar qué subsistemas deben ser implementados y en qué orden deben ser integrados, formando el Plan de Integración.
- Cada implementador decide en qué orden implementa los elementos del subsistema.
- Si encuentra errores de diseño, los notifica.
- Se prueban los subsistemas individualmente.
- Se integra el sistema siguiendo el plan.

La estructura de todos los elementos implementados forma el modelo de implementación. La integración debe ser incremental, es decir, en cada momento sólo se añade un elemento. De este modo es más fácil localizar fallos y los componentes se prueban más a fondo. En fases tempranas del proceso se pueden implementar prototipos para reducir el riesgo. Su utilidad puede ir desde ver si el sistema es viable desde el principio, probar tecnologías o diseñar la interfaz de usuario. Los prototipos pueden ser exploratorios (desechables) o evolutivos. Estos últimos llegan a transformarse en el sistema final.

Pruebas

Este flujo de trabajo es el encargado de evaluar la calidad del producto que se está desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

Esta disciplina brinda soporte a las otras disciplinas. Sus objetivos son:

- Encontrar y documentar defectos en la calidad del software.
- Generalmente asesora sobre la calidad del software percibida.
- Provee la validación de los supuestos realizados en el diseño y especificación de requisitos por medio de demostraciones concretas.
- Verificar las funciones del producto de software según lo diseñado.
- Verificar que los requisitos tengan su apropiada implementación.

Las actividades de este flujo comienzan pronto en el proyecto con el plan de prueba (el cual contiene información sobre los objetivos generales y específicos de las prueba en el proyecto, así como las estrategias y recursos con que se dotará a esta tarea), o incluso antes con alguna evaluación durante la fase de inicio, y continuará durante todo el proyecto.

El desarrollo del flujo de trabajo consistirá en planificar que es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida sirva para ir refinando el producto a desarrollar.

Despliegue

El objetivo de este flujo de trabajo es producir con éxito distribuciones del producto y distribuirlo a los usuarios (Kruchten, 2000) (Letelier, 2007).

Las actividades implicadas incluyen:

- Probar el producto en su entorno de ejecución final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el software.

- Proveer asistencia y ayuda a los usuarios.
- Formar a los usuarios y al cuerpo de ventas.
- Migrar el software existente o convertir bases de datos.

Este flujo de trabajo se desarrolla con mayor intensidad en la fase de transición, ya que el propósito del flujo es asegurar una aceptación y adaptación sin complicaciones del software por parte de los usuarios. Su ejecución inicia en fases anteriores, para preparar el camino, sobre todo con actividades de planificación, en la elaboración del manual de usuario y tutoriales.

Gestión del proyecto

La Gestión del proyecto es el arte de lograr un balance al gestionar objetivos, riesgos y restricciones para desarrollar un producto que sea acorde a los requisitos de los clientes y los usuarios.

Los objetivos de este flujo de trabajo son:

- Proveer un marco de trabajo para la gestión de proyectos de software intensivos.
- Proveer guías prácticas, realizar planeación, contratar personal, ejecutar y monitorear el proyecto.
- Proveer un marco de trabajo para gestionar riesgos.

La planeación de un proyecto posee dos niveles de abstracción: un plan para las fases y un plan para cada iteración.

Configuración y control de cambios

La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido.

Entorno

La finalidad de este flujo de trabajo es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas

que se van a necesitar en cada momento, así como definir la instancia concreta del proceso que se va a seguir.

En concreto las responsabilidades de este flujo de trabajo incluyen:

- Selección y adquisición de herramientas.
- Establecer y configurar las herramientas para que se ajusten a la organización.
- Configuración del proceso.
- Mejora del proceso.
- Servicios técnicos.

El principal artefacto que se usa en este flujo de trabajo es el caso de desarrollo que especifica para el proyecto actual en concreto, cómo se aplicará el proceso, qué productos se van a utilizar y cómo van a ser utilizados. Además se tendrán que definir las guías para los distintos aspectos del proceso, como pueden ser el modelado del negocio y los Casos de Uso, para la interfaz de usuario, el diseño, la programación, el manual de usuario.

Fases

Inicio

Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y casos de uso, y se diseñan los casos de uso más esenciales (aproximadamente el 20% del modelo completo). Se desarrolla, un plan de negocio para determinar qué recursos deben ser asignados al proyecto (Kruchten, 2000).

Los objetivos de esta fase son:

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre.

Los resultados de la fase de inicio deben ser:

- Un documento de visión: Una visión general de los requerimientos del proyecto, características clave y restricciones principales.
- Modelo inicial de casos de uso (10-20% completado).
- Un glosario inicial: Terminología clave del dominio.
- El caso de negocio.
- Lista de riesgos y plan de contingencia.
- Plan del proyecto, mostrando fases e iteraciones.
- Modelo de negocio, si es necesario
- Prototipos exploratorios para probar conceptos o la arquitectura candidata.

Al terminar la fase de inicio se deben comprobar los criterios de evaluación para continuar:

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda.
- Entendimiento de los requisitos, como evidencia de la fidelidad de los casos de uso principales.
- Las estimaciones de tiempo, coste y riesgo son creíbles.
- Comprensión total de cualquier prototipo de la arquitectura desarrollado.
- Los gastos hasta el momento se asemejan a los planeados.

Si el proyecto no pasa estos criterios hay que plantearse abandonarlo o repensarlo profundamente.

Elaboración

El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos (Kruchten, 2000) (Letelier, 2007).

En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los casos de uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

Los objetivos de esta fase son:

- Definir, validar y crear los cimientos de la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede.
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Al terminar deben obtenerse los siguientes resultados:

- Un modelo de casos de uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados.
- Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un caso de uso específico.
- Descripción de la arquitectura de software.
- Un prototipo ejecutable de la arquitectura.
- Lista de riesgos y caso de negocio revisados.
- Plan de desarrollo para el proyecto.
- Un caso de desarrollo actualizado que especifica el proceso a seguir.
- Un manual de usuario preliminar (opcional).

En esta fase se debe tratar de abarcar todo el proyecto con la profundidad mínima. Sólo se profundiza en los puntos críticos de la arquitectura o riesgos importantes. En la fase de elaboración se actualizan todos los productos de la fase de inicio.

Los criterios de evaluación de esta fase son los siguientes:

- La visión del producto es estable.
- La arquitectura es estable.
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles.
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual.
- Los gastos hasta ahora son aceptables, comparados con los previstos.

Si no se superan los criterios de evaluación quizá sea necesario abandonar el proyecto o replanteárselo considerablemente.

Construcción

La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto (Letelier, 2007) (Kruchten, 2000).

Los objetivos concretos incluyen:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea práctico.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea práctico.

Los resultados de la fase de construcción deben ser:

- Modelos Completos (casos de uso, análisis, diseño, despliegue e implementación)
- Arquitectura íntegra (mantenida y mínimamente actualizada)
- Riesgos Presentados Mitigados
- Plan del Proyecto para la fase de Transición.
- Manual Inicial de Usuario (con suficiente detalle)
- Prototipo Operacional – beta
- Caso del Negocio Actualizado

Los criterios de evaluación de esta fase son los siguientes:

- El producto es estable y maduro como para ser entregado a la comunidad de usuario para ser probado.
- Todos los usuarios expertos están listos para la transición en la comunidad de usuarios.
- Son aceptables los gastos actuales versus los gastos planeados.

Transición

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto (Letelier, 2007) (Kruchten, 2000).

Cosas que puede incluir esta fase:

- Prueba de la versión Beta para validar el nuevo sistema frente a las expectativas de los usuarios
- Funcionamiento paralelo con los sistemas legados que están siendo sustituidos por el proyecto.
- Conversión de las bases de datos operacionales.
- Entrenamiento de los usuarios y técnicos de mantenimiento.
- Traspaso del producto a los equipos de mercado, distribución y venta.

Los principales objetivos de esta fase son:

- Conseguir que el usuario se valga por si mismo.
- Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Los resultados de la fase de transición son:

- Prototipo Operacional
- Documentos Legales
- Caso del Negocio Completo
- Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la Arquitectura completa y corregida
- Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

Los criterios de evaluación de esta fase son los siguientes:

- El usuario se encuentra satisfecho.
- Son aceptables los gastos actuales versus los gastos planificados.

Artefactos

Un producto o artefacto es un trozo de información que es producido, modificado o usado durante el proceso de desarrollo de software. Los productos son los resultados tangibles del proyecto, las cosas que va creando y usando hasta obtener el producto final (Kruchten, 2000) (Letelier, 2007).

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

1.4.2 MSF

Microsoft Solution Framework (MSF) es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas (Sanchez, 2006).

Figura 5. Metodología MSF

Esta metodología tiene las siguientes características:

Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.

Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.

Tecnología Agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

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

Modelo de Arquitectura del Proyecto: Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones (Sanchez, 2006).

Modelo de Equipo: Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.

Modelo de Proceso: Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo.

Modelo de Gestión del Riesgo: Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.

Modelo de Diseño del Proceso: Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.

Modelo de Aplicación: Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar

aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores (Sanchez, 2006).

En conclusión se puede decir que la Metodología MSF se adapta a proyectos de cualquier dimensión y de cualquier tecnología.

1.4.3 XP

Extreme Programing (XP) es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, corto equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto (Sanchez, 2006).

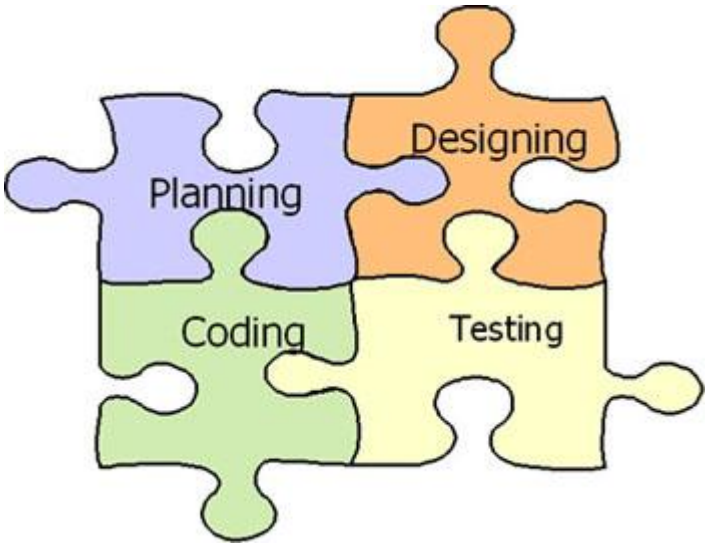


Figura 6. Metodología Extreme Programing

Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudiesen ocurrir. Es como si se adelantaran a obtener los posibles errores (Sanchez, 2006).

Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa (Sanchez, 2006) (Solís., 2003).

XP añade funcionalidad con retroalimentación continua. El manejo del cambio se convierte en parte sustantiva del proceso. El costo del cambio no depende de la fase o etapa. No introduce funcionalidades antes que sean necesarias y el cliente o el usuario se convierten en miembro del equipo (Solís., 2003).

Los clientes tienen derecho a: Decidir qué se implementa. Saber el estado real y el progreso del proyecto. Añadir, cambiar o quitar requerimientos en cualquier momento. Obtener lo máximo de cada semana de trabajo para saber así en qué estado del proyecto se encuentra. Obtener un sistema funcionando cada 3 o 4 meses para poder ver las fallas del mismo.

Por otro lado los desarrolladores tienen derecho a: Decidir cómo se implementan los procesos para una mayor eficiencia en el desarrollo del software. Crear el sistema con la mejor calidad posible para así satisfacer los requerimientos del cliente. Pedir al cliente en cualquier momento aclaraciones de los requerimientos para no caer en errores posteriores. Estimar el esfuerzo para implementar el sistema y cambiar los requerimientos en base a nuevos descubrimientos (Solís., 2003).

Lo fundamental en este tipo de metodología es la comunicación, entre los usuarios y los desarrolladores para evitar inconsistencias en el período de desarrollo del software. Así como la simplicidad, al desarrollar y codificar los módulos del sistema para una mejor organización y entendimiento del mismo. También es fundamental la retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales (Solís., 2003).

1.5 Verificación y Validación.

Una de las tareas más importantes y difíciles a las que se debe enfrentar un desarrollador de software es la validación y verificación. Se debe trabajar lo más

próximo posible a los usuarios finales durante el periodo de desarrollo y validación para reducir (o eliminar, si es posible) el escepticismo de los mismos, respecto a los resultados obtenibles mediante el proceso de desarrollo del software (Ujaen, 2008).

La verificación no es más que la construcción correcta de un modelo. Se puede definir verificación como el proceso de determinar si la lógica operacional del modelo (software) se corresponde con la lógica del diseño. En términos más simples, consiste en determinar si existen errores en el programa.

- Verificación: ¿Estamos construyendo el producto correctamente? (Rodríguez, 2006)

La validación es el proceso de comparar la salida del modelo con el comportamiento del fenómeno. En otras palabras comparar la ejecución del modelo con la realidad (física). Schlesinger lo define como "sustanciación de que un modelo para computadora, con su dominio de aplicación, posee un rango satisfactorio de precisión consistente con la aplicación para la que se desea el modelo" (Isi, 2006).

- Validación: ¿Estamos construyendo el producto correcto? (Rodríguez, 2006)

Cuando se valida un modelo se establece que el modelo es una representación creíble del sistema real, cuando se verifica un modelo se determina si la lógica del modelo ha sido correctamente implementada. Dado que los objetivos de la verificación y de la validación son diferentes también lo son las técnicas para realizarlos.

Verificación dinámica:

Las pruebas del software se ocupan de la ejercitación y la observación del comportamiento del producto (verificación dinámica). El sistema se ejecuta con datos de pruebas y se observa su comportamiento operativo (Administración, 2007).

Verificación estática:

Las inspecciones de software se ocupan del análisis de representaciones estáticas del sistema para describir problemas. Pueden ser complementadas por documentos basados en herramientas y análisis del código (Administración, 2007).

La verificación y validación (V&V) es el proceso de todo un ciclo vital. Debe aplicarse en cada etapa del software y tiene dos objetivos fundamentales:

- El descubrimiento de defectos en el sistema.
- La evaluación de si el sistema es útil y utilizable en una situación operacional o no.

Metas de la V&V

La verificación y la validación deberían establecer la confianza de que el software es adecuado al propósito. Esto no significa que esté completamente libre de defectos, sino que debe ser lo suficientemente bueno para su uso previsto y el tipo de uso determinará el grado de confianza que se necesita (Administración, 2007).

Un ejemplo de herramienta en la verificación y validación de software es Destiny una nueva herramienta que provee un análisis aún más grande y más poderoso facilitando un mayor acoplamiento entre los programas de software y sus especificaciones. También se pueden mencionar los probadores automáticos de teoremas, junto con la interpretación humana, que han sido vistos como herramientas poderosas en la verificación y validación de software computacional (Dykstra, 2008).

1.6 Gestión de la Configuración.

La Gestión de Configuración del Software es una disciplina encargada del control de la evolución de los productos de software, es el proceso de identificar y definir los elementos en el sistema, controlando el cambio de estos elementos a lo largo de su ciclo de vida, registrando y reportando el estado de los elementos y las solicitudes de cambio, y verificando que los elementos estén completos y que sean los correctos (Pressman, 2005).

Configuración por si sola es el conjunto de características funcionales y físicas de una versión específica de hardware y elementos de software que combinados de acuerdo a procedimientos de construcción específicos cumplen un propósito particular.

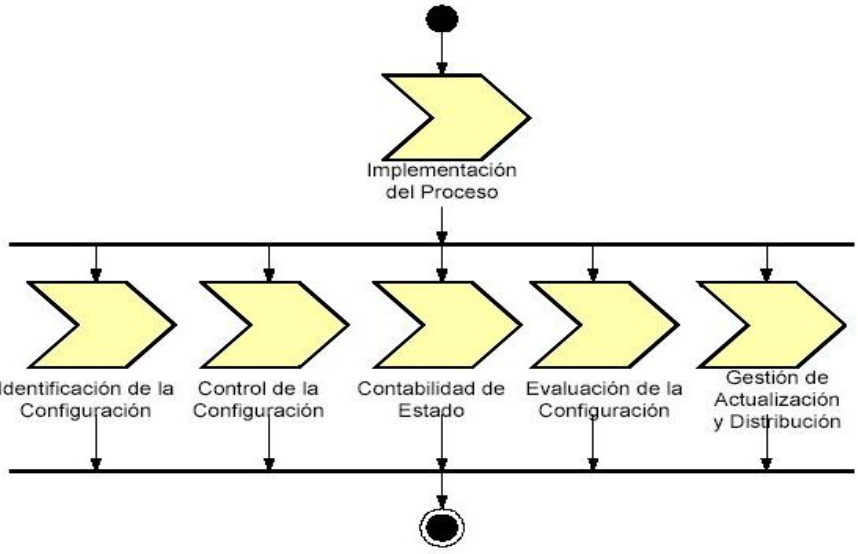


Figura 7. Diagrama de actividades OMG-SPEM de la Gestión de Configuración en ISO/IEC 12207.

El estándar IEEE Std. 1074-1995 ([IEEE 1074]) para el Desarrollo de Procesos del Ciclo de Vida del Software, establece el Proceso de Gestión de Configuración del Software como uno de los Procesos Integrales. Estos son los procesos necesarios para completar exitosamente las actividades del proyecto, y son utilizados para asegurar la finalización y calidad de las funciones del proyecto. Este proceso consiste de las siguientes actividades (Especialista, 2007):

1. Planificar la Gestión de Configuración.
2. Desarrollar la Identificación de la Configuración.
3. Realizar el Control de la Configuración.
4. Realizar la Contabilidad de Estado.

La gestión de la configuración del software es una actividad de garantía de calidad del software que se aplica en todas las fases del proceso de ingeniería del software.

Una línea base es un concepto de gestión de configuración del software que ayuda a controlar los cambios sin impedir seriamente los cambios justificados. La IEEE[20] define una línea base como: una especificación o producto que se ha revisado

[20] Instituto de Ingenieros Electricistas y Electrónicos.

formalmente y sobre los que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios (dptoInformática, 2006) (Pressman, 2005).

Figura 8. Línea base

Un ECS (elemento de configuración de software) es la información creada como parte del proceso de ingeniería. Es un documento, un conjunto completo de casos de prueba o un componente de un programa dado (dptoInformática, 2006).

1.7 Rol SQA.

Actualmente, la responsabilidad del aseguramiento de calidad del software corresponde a ingenieros del software, gestores de proyectos, clientes, comerciales y grupos de SQA. Los grupos de SQA son los encargados de velar por los intereses del cliente y asegurar que exista y se mantenga la calidad del software.

El rol para SQA es brindar a la administración la seguridad de que procesos oficialmente establecidos están siendo implementados y asegura que (Corporativo, 2008):

1.-Una metodología de desarrollo apropiada esté establecida.

2.-Que los proyectos utilicen estándares y procedimientos en su trabajo.

3.-Que la documentación sea creada para mantenimiento y mejoramiento.

4.-La administración de configuración de software esté adecuada para controlar cambios.

5.-Se realicen pruebas y que se aprueben.

6.-Cualquier deficiencia o desviación sea identificada y llevada con atención a la administración.

Las personas responsables del proyecto de software (desarrollo y cliente) son las únicas que pueden ser responsables por la calidad. El rol de SQA es monitorear la manera en que estos grupos ejecutan sus responsabilidades. Por lo tanto existen los siguientes peligros latentes (enrike, 2007):

- Es un error asumir que el personal de SQA puede por sí solo hacer algo por la calidad del proyecto.
- La existencia de una función de SQA no asegura que se siguen los estándares y los procedimientos.
- Sólo si la gerencia demuestra periódicamente su soporte a SQA, siguiendo sus recomendaciones, SQA podrá ser efectiva.
- A menos que la gerencia de línea requiera que SQA trate de resolver sus no-conformidades con la gerencia del proyecto antes de elevarlas, SQA y desarrollo no trabajarán efectivamente.

Todo lo que puede hacer SQA es alertar a la gerencia sobre las desviaciones a los estándares y procedimientos establecidos. La gerencia debe entonces insistir acerca de que los problemas de calidad se solucionen antes de que el producto sea liberado para su uso, sino SQA se transforma en un ejercicio burocrático y costoso (enrike, 2007).

1.8 Herramientas.

Algunas de las herramientas usadas en el SQA son las inspecciones, revisiones y las pruebas.

Inspeccionar: es revisar un programa con el objetivo de encontrar defectos en él. Las inspecciones actúan de manera estática. No se ejecuta el código y puede realizarse antes que la prueba (Ruiz de Mendarozqueta, y otros, 2008).

Walkthrough (recorrida) o revisiones con usuarios: Estos son los encargados de recorrer el producto para extraer requerimientos o para validar el producto. Realizan la traducción de los modelos a utilizar. Se encargan de los diagramas de instancias y de los gráficos simplificados así como de la simulación de los procesos (Ruiz de Mendarozqueta, y otros, 2008).

Revisión de Caras: Es la encargada de citar a los participantes (juntos o por separado). También es la parte encargada de pedir la opinión o lo actuado. Buscar información, ver las similitudes y diferencias que existen en el producto y como un aspecto fundamental son los encargados de acordar en todo momento (Ruiz de Mendarozqueta, y otros, 2008).

Pruebas: No son más que una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas, los resultados se observan, registran y se realiza una evolución de algún aspecto. Estas pruebas siempre deben realizarse contra un resultado esperado y para que sean exitosas deben encontrar errores (Ruiz de Mendarozqueta, y otros, 2008).

1.9 Conclusiones Parciales.

El análisis del proceso de aseguramiento de la calidad de software demuestra la necesidad de implantar una guía SQA para la facultad de informática de la Universidad de Cienfuegos. Debido a que con esto se lograría que los clientes salieran satisfechos y que los sistemas al concluir con su ciclo de vida terminaran con la menor cantidad de errores posibles.

El estudio de las diferentes metodologías de desarrollo de software y de los diferentes modelos como CMM/CMMI, SPICE e ISO/IEC9126 así como la V&V y la Gestión de Configuración permitieron conocer cuáles son las mejores técnicas para aplicar el aseguramiento de la calidad de software y saber en qué momento utilizar cada una de estas. Tomando así a RUP para el desarrollo de la guía propuesta por

ser una metodología organizada y abarcar la mayor cantidad de flujos de trabajo desde el modelado del negocio y hasta el despliegue. Además de ser la metodología estudiada y la más utilizada en la facultad de informática de la UCF.

También el conocimiento de algunas herramientas de SQA como las pruebas, las inspecciones y las revisiones permitió profundizar aun más en el tema quedando así centradas las bases para comenzar el desarrollo de la guía del SQA.

CAPÍTULO 2: Guía de Aseguramiento de la Calidad de Software para la Facultad de Informática (SQAFINF).

2.1 Introducción.

Debido a que la facultad de informática de la Universidad de Cienfuegos no es una empresa productora de software y sus desarrolladores no cuentan con la experiencia suficiente en el campo del aseguramiento de la calidad, en este capítulo se propondrá una guía para que los mismos, puedan llevar a cabo el aseguramiento de la calidad de cada producto software desarrollado desde sus inicios y hasta su puesta en marcha. Tratando de mejorar con dicha guía la calidad de cada uno de los sistemas que sean implementados en la facultad de informática de la UCF y además incrementar los conocimientos y preparar a todos los estudiantes que se encuentren vinculados a la producción de software.

2.2 Guía.

2.2.1 Objetivo.

Elaborar una Guía para facilitar el aseguramiento de la calidad en el proceso de desarrollo de software en la facultad de informática de la Universidad de Cienfuegos (UCF) Carlos Rafael Rodríguez.

2.2.2 Propósito.

Con la confección de esta guía se logrará que los desarrolladores de software de la facultad de informática de la Universidad de Cienfuegos lleven el aseguramiento de la calidad a lo largo del ciclo de vida del software. Así como también facilitar una mayor organización y rapidez dentro del proceso de desarrollo de software.

2.2.3 Alcance.

Con la elaboración de esta guía se espera obtener excelentes resultados debido a que se vinculan actividades de SQA, gestión de la configuración y Verificación y Validación (V&V), proponiendo que se ejecuten de forma paralela al ciclo de vida de desarrollo de cualquier software, mejorando así la calidad de los productos de

software desarrollados por la facultad de informática de la UCF. Teniendo como meta fundamental verificar que todo software y documentación liberados cumpla con los requerimientos técnicos especificados por cada cliente y los determinados por el equipo de desarrollo.

2.3 Explicación.

Para tener un mejor conocimiento de la estrategia que se propondrá se muestra una tabla especificando los flujos de trabajo definidos por la metodología RUP, así como cada una de las actividades definidas por SQA, Gestión de la Configuración y V&V, que permitan llevar un buen control del aseguramiento de la calidad de los software desarrollados.

Se ubica al recurso humano encargado de llevar el aseguramiento de la calidad, en qué flujo de trabajo se encuentra y cada una de las actividades que deben ser llevadas a cabo en el mismo, dándole la posibilidad de orientarse y saber cuáles son los parámetros que debe medir y chequear.

Posteriormente se hace una descripción detallada de cada una de las tareas que deben ser desarrolladas en cada actividad, explicando con exactitud lo que se debe hacer. Así como las posibles entradas y salidas que se puedan generar.

Tabla 1. Actividades y tareas de V&V, Gestión de la Configuración y SQA.

RUP	V&V	Gestión de la Configuración	SQA
Negocio	**V&V de la conceptualización** -Plan de tareas V&V -Evaluar Documentación	-Plan de Proyecto -Plan de Gestión de la Configuración	-Documento Visión -Pronóstico -Identificación de Riesgos -Plan de Proyecto -Plan de Calidad -Plan SQA
Requisitos	**V&V de Requerimientos** -Análisis de la Interfaz -Revisión e Inspección de los requerimientos de software -Plan V&V de generación de prueba y verificación del sistema	-Documento de definición de requerimientos -Documento de análisis del sistema -Documento de diseño del sistema	-Requisitos generales del proyecto -Evaluar los planes -Evaluar los requerimientos -Evaluar las herramientas de software
Análisis y Diseño	**V&V de Análisis y Diseño** - Evaluación de Diseño de Software - Análisis de la Interfaz	-Especificación de diseño -Documento de diseño de alto nivel	-Evaluar el diseño de software -Evaluar documento de

	- Plan V&V de Generación de prueba y de verificación de Componente - Plan V&V de Generación de prueba y de verificación de Integración - Plan V&V de Generación de prueba y de verificación de Diseño - Revisión e Inspección de la arquitectura	-Documento de diseño de bajo nivel -Prototipos -Descripciones de diseño de las interfaces	análisis -Realizar revisiones y seguimiento de las desviaciones encontradas -Verificar avances del proyecto
Implementación	**V&V de Implementación** -Análisis de la interfaz -Evaluación de código fuente y documentación -V&V de la gestión de procedimientos de prueba y verificación -V&V de prueba y ejecución	-Conformar línea base del proyecto y documento asociado -Código fuente del programa -Código objeto y ejecutable -Documento de vercionamiento -Documento de diseño de bases de datos	-Control de código -Reportes de errores y actividades correctivas -Evaluar desarrollo del software -Evaluar las desviaciones -Revisiones e intervenciones -Verificar avances del

	-V&V de verificación de componentes		proyecto
Prueba	**V&V de Pruebas** -V&V de la ejecución y verificación de las pruebas de integración -V&V de pruebas de ejecución y verificación de aceptación -V&V de prueba de ejecución y verificación del sistema -V&V de la documentación	-Realizar auditorías de configuración física -Realizar auditorías de configuración funcional -Plan y procedimientos de prueba -Casos de prueba y resultados registrados -Planes de prueba de unidad -Datos de prueba -Especificaciones de prueba de unidad	-Actividades de testeo -Actualizar plantillas de métricas de calidad -Realizar auditorías de procesos -Evaluar procesos de Administración de la Configuración de Software (SCM) -Evaluar proceso de prueba de unidad -Pruebas de integración -Revisiones del proyecto y auditorías -Revisiones técnicas -Verificar avances del proyecto

			-Inspecciones de software
Despliegue	**V&V de Instalación y Chequeo** -V&V de mantenimiento * V&V de generación de informe final * Auditoría de la configuración de instalación * Evaluación de las migraciones * Análisis de criticidad * Revisiones internas	-Datos de prueba -Datos de proyecto -Manuales de usuario -Documentos de mantenimiento *Informes de problemas de software *Peticiones de mantenimiento *Órdenes de cambio de Ingeniería	-Control de medios -Colección de registros, mantenimiento y retención -Entrenamiento -Administración de riesgos -Control de proveedores -Evaluar las instalaciones -Plan de despliegue

2.3.1 Negocio

Actividad V&V de la conceptualización: Esta representa la delimitación de una solución de aplicaciones específicas para resolver el problema del usuario. Durante la actividad de Conceptualización, la arquitectura del sistema está seleccionada, y los requisitos del sistema son asignados a hardware, software y componentes de interfaz de usuario. El propósito de esta actividad es el de validar y verificar la asignación de los requisitos del sistema, la solución escogida, asegurar que no existan falsas suposiciones y velar porque sean incorporados en la solución (IEEE P1012/D12, 2004).

Tareas que deben ser desarrolladas en esta actividad.

- ✓ Plan de tareas V&V.

  Entradas:

  Informe de tareas.

  Evaluación de la documentación.

  Salidas:

  Modelos y planes de desarrollo.

  Definir o Elegir Modelos de ciclo de vida.

  Plan de prueba.

- ✓ Evaluar Documentación:

  Entradas:

  Plan de desarrollo de proveedores y anexos.

  Necesidades de los usuarios.

  Las necesidades de adquisición.

  Salidas:

  Informe Tareas.

  Evaluación de la Documentación del Negocio.

Plan de Proyecto: Este plan será usado para evaluar y administrar el proyecto. Las suposiciones principales que puedan afectar el plan deberán ser documentadas aquí. Este deberá ser actualizado durante el ciclo de vida del proyecto. En resumen el plan de proyecto no es más que la fotografía de las

acciones que se deben tomar para alcanzar un objetivo determinado (Software, 2005).

Plan de Gestión de la Configuración: Este documento describe todas las actividades de Gestión de Configuración y Cambios que serán realizadas durante todo el ciclo de vida del proyecto. Muestra planificaciones detalladas de las actividades, responsabilidades asignadas, recursos necesarios que incluyen personal, herramientas y equipamiento. Este plan contiene información que puede ser cubierta a una mayor o menor magnitud por otros planes (José, 2006).

Documento Visión: Este refleja los valores y propósitos principales de un proyecto. Es un documento primordial tanto como punto de partida como para ir aclarando y pudiendo tomar las decisiones correctas en el camino. También es clave para saber quiénes están de acuerdo y quiénes no, con el propósito del proyecto. Se centra fundamentalmente en la funcionalidad requerida por los participantes en el proyecto y los usuarios finales (Arca, 2007).

Pronóstico: esta actividad se encarga de brindar los pronósticos del proyecto en cuanto a tiempo, costo y distribución del mismo.

Identificación de Riesgos: es el proceso por el cual se determina qué riesgos pueden afectar al proyecto y se documentan sus características. Este proceso es un proceso interactivo, ya que se descubrirán nuevos riesgos a medida que se avance con el ciclo de vida del proyecto. En este proceso se elabora el Registro de Riesgos, el cual contiene en la etapa de planificación por lo menos los siguientes cuatro datos (Losada, 2005):

- ✓ Nombre y descripción del riesgo.
- ✓ Descripción de las respuestas posibles al riesgo.
- ✓ Causa primaria o raíz (“Root Cause”) del riesgo: condición o evento que harán que el riesgo se concrete.
- ✓ Categoría de riesgo: tecnológico, técnico, organizacional, externo.

Amenaza de no alcanzar alguna de las metas principales del proyecto. Estos riesgos aumentan con la incertidumbre y disminuyen con la aplicación de métodos.

Plan de Calidad: Este documento especifica los objetivos de calidad, selecciona estrategias para asegurar que las metas se han logrado, y detalla un plan de acción para llevar a cabo las estrategias.

Plan SQA: no es más que un conjunto de evaluaciones, auditorías y revisiones a realizar, estándares aplicables al proyecto, procedimientos para información y seguimiento de errores, documentos producidos por el grupo SQA, realimentación de información (feedback) al equipo que permiten que se lleve correctamente la calidad de un proyecto (software, 2009).

2.3.2 Requisitos

Actividad V&V de Requerimientos: responde a los requerimientos de software de análisis de los requisitos funcionales y de rendimiento, las interfaces externas al software, los requisitos de cualificación, la seguridad, ingeniería de factores humanos, las definiciones de datos, documentación del usuario para el software, la instalación y aceptación, las operaciones de usuario así como la ejecución y mantenimiento de usuario. Su objetivo se centra en garantizar la exactitud, integridad, capacidad de prueba, y la coherencia de los requisitos del sistema de software (IEEE P1012/D12, 2004).

Tareas que deben ser desarrolladas en esta actividad.

- ✓ Análisis de la Interfaz.

  Entradas:

  Documentación del negocio.

  Especificación de requisitos del software. (SRS)

  Especificación de requisitos de la interfaz. (IRS)

  Salidas:

  Tarea Informe(s) – Análisis de la Interfaz.

  Informe(s) anomalía.

- ✓ Revisiones e Inspecciones de los Requerimientos de Software.

  Entradas:

  La documentación del sistema.

  Especificación de requisitos del software.

Especificación de requisitos de la interfaz.

Salidas:

Informe de Evaluación de requisitos de software.

- ✓ Plan V&V de Generación de Pruebas y de Verificación del Sistema.

Entradas:

La Documentación del negocio (los requisitos del Sistema).

Especificación de requisitos del software.

Especificación de requisitos de la interfaz.

La Documentación del usuario.

El Plan de Prueba de sistema.

Salidas:

Informe de V&V del Plan de Prueba del Sistema.

Documento de requerimientos del software: este es la declaración oficial de qué es lo que requieren los desarrolladores del sistema. Incluye tanto los requerimientos del usuario para el sistema como una especificación detallada de los requerimientos del sistema. En algunos casos, los dos tipos de requerimientos se integran en una única descripción. En otros, los del usuario se definen en una introducción de la especificación de los del sistema. Si existe un gran número de requerimientos, los detalles de los requerimientos del sistema se pueden presentar como documentos separados. El documento de requerimientos tiene un conjunto diverso de usuarios que va desde los administradores principales de la organización, quienes pagan por el sistema, hasta los ingenieros responsables del software (Lauro, 2008).

Documento de análisis del sistema: guarda la descripción detallada de todos los componentes y pasos que se utilizan para realizar un sistema (Pressman, 2005).

Documento de diseño del sistema: en este se presenta una visión global y resumida de la arquitectura del sistema y de los objetivos generales del diseño. Se describen las influencias con los requisitos funcionales y no funcionales del sistema y las decisiones y prioridades establecidas como por ejemplo eficiencia vs portabilidad. El objetivo del diseño se basa en la enumeración de las pautas de diseño del sistema. Algunos corresponderán con los requisitos no funcionales establecidos en la fase de

requisitos y otros con los requisitos generales de calidad del diseño. Se justificarán las decisiones tomadas en la elección entre objetivos deseables pero incompatibles, así como las prioridades establecidas en el diseño e implementación del sistema (InfoMedia, 2010).

Requisitos generales del proyecto: principales requerimientos a tener en cuenta en todo el proyecto. Estos son especificados minuciosamente teniendo en cuenta siempre las necesidades del cliente.

Evaluar los planes: el equipo de SQA deberá evaluar todos los planes para el proyecto (plan de desarrollo de software, plan de administración de la configuración del software, plan de administración de riesgos, plan de calidad, plan SQA), ayudando a identificar los estándares y los lineamientos para los mismos (software, 2009).

Evaluar los requerimientos: el análisis de requisitos establece un único entendimiento del problema entre el cliente y el equipo de desarrollo. Se realiza un documento con el cliente sobre los requisitos para el software el cual es establecido y mantenido por el personal encargado (software, 2009) (Anexo 1).

Tareas para los requerimientos:

- ✓ Verificar que los participantes correctos estén involucrados en el análisis de los requisitos para identificar todas las necesidades del usuario.
- ✓ Revisar todos los requerimientos para determinar si su implementación es factible.
- ✓ Verificar que los contratos son documentados, comunicados y revisados.
- ✓ Verificar que los requisitos que puedan presentar algún tipo de error sean analizados por el equipo de requerimientos.
- ✓ Verificar que los requisitos estén documentados.
- ✓ Darle seguimiento a los cambios que puedan tener los requerimientos.
- ✓ Verificar que las personas involucradas en el equipo de requisitos estén entrenadas para el trabajo.
- ✓ Verificar que los procesos establecidos para definir y documentar requisitos son seguidos y documentados.

Evaluar las herramientas de software: esta tarea consiste en verificar si en realidad las herramientas son útiles para el desarrollo del proyecto y que además sean soportadas por los sistemas de cómputo actuales en los que se desarrolla el proyecto (software, 2009) (Anexo 2).

2.3.3 Análisis y diseño

Actividad V&V de Análisis y Diseño: los requisitos de software se transforman en una arquitectura y un diseño detallado para cada componente de software. El diseño incluye bases de datos y de interfaces de sistema. Esta actividad aborda la arquitectura de software de diseño detallado y su objetivo consiste en demostrar que el diseño es correcto, preciso y que no se introducen las características no deseadas en la transformación de los requisitos de software (IEEE P1012/D12, 2004).

Tareas que deben ser desarrolladas en esta actividad.

- ✓ Evaluación de Diseño de Software.

  Entradas:

  SRS, IRS.

  SDD (Documento de Diseño de software).

  IDD (Documento de Diseño de la Interfaz).

  Diseñe las Normas (normas, prácticas, y convenciones).

  Salidas:

  El Informe de la tarea (s) –la Evaluación de Diseño de Software.

  El Informe de la anomalía (s).

- ✓ Análisis de la Interfaz.

  Entradas:

  La Documentación de concepto (los requisitos del Sistema).

  SRS, IRS, SDD, IDD.

  Salidas:

  Informe de la tarea (s) –el Análisis de la Interfaz

  Informe de la anomalía (s)

- ✓ Plan V&V de Generación de prueba y de verificación de Componente.

  Entradas:

SDD, IDD.

Documentación del usuario.

Planes de diseño.

Planes de prueba.

Salidas:

Componente V&V.

Diseño de prueba (s).

Informe de anomalía (s).

- ✓ Plan V&V de Generación de prueba y de verificación de Integración.

Entradas:

SRS, IRS, SDD, IDD.

Plan de prueba de integración.

Salidas:

La integración del Plan de Prueba V&V.

Informe de anomalía (s)

- ✓ Plan V&V de Generación de prueba y de verificación de Diseño.

Entradas:

SDD, IDD.

La Documentación del usuario.

Los Planes de diseño.

Los Planes de prueba.

Salidas:

Componente V&V.

Diseño de prueba (s).

Informe de anomalía (s).

La integración de V&V de Diseño(s) de Prueba.

El sistema de V&V de Prueba Diseño (s).

- ✓ Revisión e Inspección de la arquitectura.

Entradas:

SDD. IDD.

Informe del Análisis de riesgo.

Análisis de seguridad.

Tarea V&V de los Resultados.

Salidas:

Informe de la tarea (s) –el Análisis de Riesgo.

Informe de la anomalía (s).

Informe de la tarea (s) –el Análisis de Seguridad.

Especificación de diseño: selección y justificación del diseño a utilizar.

Documento de diseño de alto nivel: es el encargado de llevar la parte del proyecto que define de forma explícita pero general el comportamiento del mismo (García, 2002).

Una lista de las propiedades a verificar del Diseño de Alto Nivel es la siguiente:

- ✓ Todos los Casos de Uso se identificaron y se priorizaron, identificándose los más relevantes.
- ✓ Las Interfaces funcionales con otros sistemas de software se definieron completamente.
- ✓ El diseño se ha realizado como un todo, definiéndose claramente la localización de los componentes del software (capas, módulos, etc.), flujos de información y funcional, requerimientos de almacenamiento y diseño de la base de datos.
- ✓ El diseño es verificable (por ejemplo, es posible construir un prototipo).
- ✓ Las interfaces máquina – humano son adecuadas y consistentes con el diseño.

Documento de diseño de bajo nivel: es aquella parte del diseño que se centra en los aspectos más técnicos del proyecto.

Descripciones de diseño de las interfaces: interfaz es el conjunto de trabajos y pasos que seguirá el usuario, durante todo el tiempo que se relacione con el programa, detallando lo que verá y escuchará en cada momento, y las acciones que realizará, así como las respuestas que el sistema le brindará. Los documentos de diseño de la interfaz deben incluir

- ✓ Objetos en las pantallas.
- ✓ Interacciones posibles sobre cada ventana o pantalla.

Prototipos: consiste en la creación de una versión más detallada y realista de la interfaz. Es importante que se realice un demo del prototipo, ya sea con el cliente o director del proyecto para demostrar que se han mitigado todos los riesgos tecnológicos y que exista un resultado visible del mismo. El prototipo sirve para ir estableciendo la coordinación y comunicación entre el grupo de especialistas técnicos y asegurarse que se estudiaron realmente las tecnologías a aplicar en el proyecto. Además es vital que se tomen medidas acerca de la productividad del grupo, poder ajustar las estimaciones y tener elementos para decidir a quién asignar tareas específicas. También se debe intentar tomar registro acerca de la proporción de errores en el prototipo para poder estimar la densidad de defectos a buscar en las próximas fases.

Evaluar el diseño de software: se encarga de revisar el diseño del software de manera sistemática previendo siempre que cumpla con las especificaciones del cliente (Anexo 3).

Tareas de SQA en el proceso de diseño:

- ✓ Verificar que los procesos de diseño de software sigan los estándares determinados.
- ✓ Verificar que todos los elementos que no cumplan con la calidad requerida sean procesados de acuerdo a los estándares y procedimientos establecidos.
- ✓ Verificar que la matriz de rastreo de los requerimientos al diseño esté lista.
- ✓ Verificar que todos los requerimientos estén presentes en el diseño.

Evaluar documento de análisis de sistemas: este documento relaciona los siguientes aspectos

- ✓ Identifica las necesidades (objetivos, funciones, rendimiento, información)
- ✓ Estudio de viabilidad (riesgos, restricciones, impedimentos legales)
- ✓ Análisis económico: (costos vs beneficios)
- ✓ Análisis técnico: (mejoras técnicas)

- ✓ Asignación de funciones (descomponer el sistema en módulos y asignar las funciones a algunos de sus componentes)

Realizar revisiones y seguimiento de las desviaciones encontradas: se basa en evitar que las desviaciones sean acumuladas para así poder corregir las mismas a medida que van siendo detectadas.

Verificar avances del proyecto: SQA verificará periódicamente el estado del proyecto, el progreso, los problemas y los riesgos del mismo, brindando una valoración independiente acerca del proyecto. Además, SQA le proveerá la siguiente información a la administración (Anexo 4):

Cumplimiento: identificará el nivel de cumplimiento actual del proyecto.

Problemas: identificará los problemas potenciales o actuales que pueden afectar en el desarrollo del proyecto.

Riesgos: identificará los riesgos basado en el estado de avance del proyecto.

Entonces los resultados se le informarán al administrador del proyecto y al equipo de desarrollo.

2.3.4 Implementación

Actividad V&V de Implementación: transforma el diseño del sistema en código, estructuras de base de datos y equipos relacionados con las representaciones ejecutables. El objetivo de V&V de implementación es para verificar y validar que estas transformaciones son correctas, exactas y completas (IEEE P1012/D12, 2004).

Tareas que deben ser desarrolladas en esta actividad.

- ✓ Evaluación del Código Fuente y documentación: Evalúa los componentes de código fuente (el código fuente y documentación de código fuente) para la corrección, consistencia, integridad, exactitud, legibilidad y mantenimiento.

  Entradas:

  El Código fuente.

  SDD. IDD.

Las Normas codificado (normas, prácticas, restricciones del proyecto, convenciones). La Documentación del usuario.

Salidas:

El Informe de la tarea (s) –el Código Fuente y Código Fuente de la Evaluación de Documentación.

El Informe de la anomalía (s).

- ✓ Análisis de la Interfaz.

  Entradas:

  La Documentación de concepto (los requisitos del Sistema).

  SDD.IDD.

  El Código fuente.

  La Documentación del usuario.

  Salidas:

  Informe de la tarea (s) –el Análisis de la Interface.

  Informe de anomalía (s)

- ✓ V&V Generación de casos de prueba y de verificación.

  Entradas:

  SRS, IRS, SDD, IDD.

  La Documentación del usuario.

  El Diseño de prueba.

  Los Casos de prueba.

  Salidas:

  Componente de V&V Casos de Prueba.

  Informe de anomalía (s).

- ✓ V&V Generación de Procedimientos de Pruebas y Verificación.

  Entradas:

  SRS, IRS, SDD, IDD.

  La Documentación del usuario.

  Los Casos de prueba.

  Pruebe los Procedimientos.

  Salidas:

Componente de V&V Procedimientos de Prueba.

Informe de anomalía (s).

- ✓ V&V Prueba de Ejecución y Verificación de Componentes.

  Entradas:

  Código fuente.

  Código ejecutable.

  SDD.

  IDD.

  Los Planes de Prueba de componente.

  Los Procedimientos de Prueba de componente.

  Los Resultados de Prueba de componente.

  Salidas:

  Informe de la tarea (s) – los Resultados de la Prueba.

  Informe de anomalía (s).

Conformar línea base del proyecto y documento asociado: una revisión aprobada de un documento o fichero fuente, a partir del cual se pueden realizar cambios subsiguientes y su documentación pertinente.

Código fuente del programa: es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa por tanto es donde está descrito su funcionamiento.

Código objeto: es el código que resulta de la compilación del código fuente.

Código ejecutable: para obtener este es necesario enlazar todos los archivos de código objeto con un programa llamado linker (enlazador).

Documento de vercionamiento: almacena y controla las distintas versiones del código fuente o del proyecto en general.

Documento de diseño de bases de datos: es el encargado de almacenar todo lo relacionado con el diseño de la base de datos.

Control de código: esta actividad se subdivide en dos.

Control estático del código: es la validación del código contra un conjunto de reglas, mejores prácticas y estándares predefinidos.

Control dinámico del código: el control se focaliza en el uso de los recursos que hace la aplicación y la cobertura del código que hacen las pruebas unitarias.

Reportes de errores y actividades correctivas: el proceso de acciones correctivas se describe de manera siguiente:

1-Identificar el problema.

2-Reportar el problema a las autoridades pertinentes.

3-Analizar el problema y dar una solución.

4-Realizar la acción correctiva.

El grupo SQA se encargará de:

Revisar el proceso de acción correctiva continuamente para estar seguro de que se está aplicando con los estándares establecidos.

Realizar análisis periódicos a todos los problemas reportados para identificar tendencias que puedan generar problemas genéricos en las aéreas. Este análisis debe tener el estudio de las causas, la magnitud del impacto, la frecuencia en la que ocurre y las medidas de prevención.

Evaluar desarrollo del software: consiste en las revisiones y verificaciones del software durante todo el desarrollo del mismo.

Evaluar las desviaciones: En caso de que haya desviaciones en el proyecto el equipo de SQA deberá de apoyar o dar soporte a la administración de proyectos.

Revisiones e intervenciones: Se realizan periódicamente para verificar que todos los procedimientos del proyecto están siendo ejecutados según los estándares, normas y especificaciones del cliente.

Verificar avances del proyecto: fue descrito en el flujo de análisis y diseño.

2.3.5 Prueba

Actividad V&V de Pruebas: incluye a las pruebas de integración de software, las pruebas de validación de software, pruebas de integración de sistemas, y las pruebas de calificación del sistema. El objetivo V&V de Prueba es garantizar que los requisitos de software y los requisitos del sistema asignados al software son validados para la ejecución de la integración, sistema, y pruebas de aceptación (IEEE P1012/D12, 2004).

Tareas que deben ser desarrolladas en esta actividad.

- ✓ V&V Generación de Procedimientos de Pruebas y Verificación de la Aceptación.

  Entradas:

  SDD.

  IDD.

  El Código fuente.

  La Documentación del usuario.

  El Plan de Prueba de aceptación.

  Los Procedimientos de Prueba de aceptación.

  Salidas:

  La aceptación de V&V Procedimientos de Prueba.

  Informe de anomalía (s).

- ✓ V&V Ejecución y Verificación de las Pruebas de Integración

  Entradas:

  El Código fuente.

  El Código ejecutable.

  El Plan de Prueba de integración.

  Los Procedimientos de Prueba de integración.

  Los Resultados de Prueba de integración.

  Salidas:

  Informe de la tarea (s) –los Resultados de la Prueba.

  Informe de anomalía (s).

- ✓ V&V Prueba de Ejecución y Verificación de Sistema.

  Entradas:

  Código fuente.

  Código ejecutable.

  Plan de Prueba de sistema.

  Los Procedimientos de Prueba de sistema.

  Los Resultados de Prueba de sistema.

  Salidas:

  El Informe de la tarea (s) –los Resultados de la Prueba

  El Informe de la anomalía (s)

- ✓ V&V Prueba de Ejecución y Verificación de Aceptación'.

  Entradas:

  Código de la fuente.

  Código ejecutable.

  La Documentación del usuario.

  Plan de Prueba de aceptación.

  Los Procedimientos de Prueba de aceptación.

  Los Resultados de Prueba de aceptación.

  Salidas:

  Informe de la tarea (s) –los Resultados de Prueba.

  Informe de anomalía (s).

- ✓ V&V de la Documentación.

  Entradas:

  Documentación de usuario.

  Documentación del Diseño de Software.

  Documentación del Diseño de la Interfaz.

  Documentación del Sistema.

  Documentación del Negocio.

  Salidas:

  Plan de la Documentación.

  Documentos preparados.

Documentos producidos.

Documentos Modificados.

Realizar auditorías de configuración física: es una inspección formal al producto físico para verificar que esté conforme con lo que indican los documentos de configuración (República, 2003) (Anexo 5).

Realizar auditorías de configuración funcional: Es una inspección formal para verificar que un ítem de configuración haya alcanzado los resultados y las características funcionales especificadas en los documentos de configuración (República, 2003) (Anexo 5).

Casos de prueba y resultados registrados: son un conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

Planes de pruebas: un plan de pruebas está constituido por un conjunto de pruebas las cuales deben (República, 2003):

- ✓ Dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad,)
- ✓ Dejar claro cómo se mide el resultado
- ✓ Especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta)
- ✓ Definir cuál es el resultado que se espera (identificación, tolerancia,) ¿Cómo se decide que el resultado es acorde con lo esperado?

Datos de prueba: son los datos que se generan al culminar cualquier proceso de prueba.

Especificaciones de prueba de unidad: no son más que las principales pautas para realizar la prueba de unidad.

Realizar auditorías de procesos: identifica, documenta y analiza las desviaciones de procesos, controla las correcciones, e informa periódicamente los resultados al gestor del proyecto (República, 2003).

Evaluar procesos de SCM (software, 2009) (Anexo 6).

- ✓ Verificar que las configuraciones de identificación de documentos, código, y datos de computadora, han sido establecidas de acuerdo a los estándares de título y nombres.
- ✓ Verificar que las líneas bases han sido establecidas por medio de los estándares y procedimientos definidos.
- ✓ Verificar que las personas que van a participar en las auditorías conozcan el sistema y tengan dominio de la administración de la configuración.
- ✓ Verificar que los procesos de administración de la configuración se sigan al pie de la letra.

Evaluar proceso de prueba de unidad: es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado (software, 2009).

- ✓ Verificar que los procesos de codificación y pruebas de unidas están siendo implementados de acuerdo a los estándares y procesos establecidos.
- ✓ Verificar que los elementos encontrados en las revisiones del código sean procesados mediante los estándares y procedimientos establecidos.
- ✓ Verificar que las pruebas de unidad se sigan al pie de la letra.

Pruebas de integración: son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez (Pressman, 2005).

- ✓ Verificar que las actividades de prueba de software y el ambiente donde se van a llevar a cabo las mismas ya esté identificado.
- ✓ El equipo de SQA llevará a cabo las pruebas.
- ✓ Los lineamientos para las pruebas ya están definidos
- ✓ Ejecutar las pruebas de unidad.
- ✓ Ejecutar las pruebas de integración.
- ✓ Analizar el resultado de las pruebas.

Revisiones del proyecto y auditorías: El equipo de SQA será el encargado de realizar las revisiones y auditorías durante todo el proceso de desarrollo del proyecto para garantizar así que los errores sean mínimos y se puedan corregir en el menor tiempo posible (Pressman, 2005).

Revisiones técnicas formales (RTF): el equipo de SQA es el encargado de llevar a cabo todas las revisiones técnicas y debe tener conocimiento acerca del producto que se evaluará. Estas revisiones permiten purificar las actividades desarrolladas en el proceso de ingeniería de software, detectar errores y defectos (Pressman, 2005).

Estas RTF se utilizan para:

- ✓ Descubrir errores en la función, la lógica o la implementación.
- ✓ Verificar que el software bajo revisión alcanza sus requisitos.
- ✓ Asegurar que el software ha sido realizado con ciertos estándares.
- ✓ Descubrir errores al principio para que no se propaguen a etapas posteriores.

Algunos elementos a los que se les puede aplicar la revisión técnica son los siguientes:

Especificación de requisitos de software: el objetivo es determinar que los requisitos están listos, para que se pueda pasar al diseño arquitectónico (Pressman, 2005).

Arquitectura de software: el objetivo es determinar que los requisitos se encuentran contemplados en el diseño de alto nivel.

Diseño de software: el objetivo es determinar que todos los requisitos estén presentes en el diseño y que el mismo es consistente (Pressman, 2005).

Desarrollo de software: el objetivo es determinar que todos los productos de software desarrollados cumplen con los estándares establecidos.

Inspecciones de software: Implican que las personas examinen la representación de la fuente con el propósito de descubrir anomalías y defectos.

- ✓ Las inspecciones no requieren la ejecución de un sistema por lo que debe utilizarse antes de la implementación (Pressman, 2005).

- ✓ Pueden estar aplicados a cualquier representación del sistema (requerimientos, diseño, configuración, datos, pruebas de datos).
- ✓ Se ha demostrado que es una técnica efectiva para descubrir errores del programa.

Verificar avances del proyecto: fue descrito en el flujo de análisis y diseño.

2.3.6 Despliegue

V&V de Instalación y Chequeo: el producto de software está instalado y probado en el entorno de ambiente. El objetivo de V&V instalación y chequeo es verificar y validar la exactitud de la instalación del software en el entorno de ambiente (IEEE P1012/D12, 2004).

Tareas que deben ser desarrolladas en esta actividad.

- ✓ Auditoría de Instalación de Configuración.

  Entradas:

  El Paquete de la instalación (por ejemplo, Código Fuente, Código Ejecutable, la Documentación del Usuario, SDD, IDD, SRS, IRS, la Documentación de Concepto, los Procedimientos de la Instalación, y Datos de Dirección de Configuración).

  Salidas:

  Informe de tarea (s) – la Auditoría de Configuración de Instalación.

  Informe de anomalía (s.)

- ✓ Chequeo de instalación.

  Entradas:

  La Documentación del usuario.

  El Paquete de instalación.

  Salidas:

  Informe de tarea (s) – Chequeo de la Instalación.

  Informe de anomalía (s).

- ✓ V&V Informe Final de Generación: Resume las actividades de V&V, tareas y resultados, incluso el estado y disposición de anomalías. Proporciona una valoración de la calidad del software global y las recomendaciones.

Entradas:

V&V Actividad Resumen Informe (s).

Salidas:

Informe de tarea (s) –V&V Último Informe.

- ✓ V&V Análisis de Riesgos: Revisa y pone al día el análisis de riesgo que usa los informes de las tareas anteriores. También te permite recomendar acciones para eliminar y reducir los riesgos.

  Entradas:

  Paquete de la instalación.

  Desarrollo del proveedor Plan y Horarios.

  Análisis de seguridad.

  Salidas:

  Informe de tarea (s) –el Análisis de Riesgo.

  Informe de anomalía (s).

Datos de proyecto: incluye toda la documentación referente al proyecto.

Manuales de usuario: son los documentos que deben ser entregados al usuario para un mejor dominio y entendimiento del software.

Documentos de mantenimiento: documentación referente al mantenimiento del software

- ✓ Informes de problemas de software.
- ✓ Peticiones de mantenimiento.
- ✓ Órdenes de cambio de Ingeniería: un cambio de ingeniería es la adición, eliminación o modificación de un producto durante su diseño, desarrollo o fabricación. Aparece durante el progreso del programa, la mayoría de veces causado por un entorno de desarrollo o un avance en la tecnología del producto. Un cambio puede traer un serio impacto sobre el alcance, el coste y la programación del proyecto.

Control de medios: se basa en llevar el control de todos los medios que serán utilizados en el proceso de desarrollo del software.

Colección de registros, mantenimiento y retención: almacenamiento de todos los registros de mantenimiento y retención del software para un mejor control por parte del usuario.

Entrenamiento: es el adiestramiento que se le debe dar al usuario luego de ser desplegado el software para que tengan dominio y conocimiento sobre el mismo.

Administración de riesgos: significa planear, organizar, dirigir y ejecutar tanto procesos como actividades que conlleven a asegurar que los software de la facultad de informática estén protegidos apropiadamente contra los riesgos que podrían afectarlos. Resumiendo el SQA debe asegurar en el transcurso de todo el proyecto que se realiza un seguimiento de los riesgos dentro de cada área y que estos son discutidos en las reuniones quincenales. En cada iteración se vuelven a evaluar y se le da un orden a los riesgos de acuerdo al impacto y a la probabilidad de ocurrencia actual.

Control de proveedores: la participación de los proveedores en el diseño, análisis y proceso de desarrollo del software contribuye en gran medida a la calidad del mismo. Debido a que estos van vinculándose estrechamente con el producto que se obtendrá al final y pueden contribuir con la detección de posibles errores que posea el producto.

Evaluar las instalaciones: SQA debe de evaluar las instalaciones actuales y las futuras, estos deben de proveer el equipo necesario y el espacio necesario para el desarrollo y soporte (software, 2009) (Anexo 7).

Plan de despliegue: establece cómo se llevará a cabo la instalación del producto propuesto en su ámbito de producción final.

2.4 Estrategia Propuesta.

Para lograr una base sólida del objetivo propuesto se hará un vínculo entre las actividades de SQA, la gestión de configuración y las actividades de V&V, donde se seleccionaron de ellas las más asequibles y de mejor aplicación en la facultad de informática y que permitan llevar a cabo un proceso bien determinado de aseguramiento de la calidad del software.

Aplicando dichas actividades paralelamente a cada uno de los flujos de trabajo principales definidos por RUP se facilita un mejor desempeño por parte del desarrollador debido a que los mismos ya se encuentran relacionados con la metodología propuesta porque es la estudiada en clase, determinando así que la estrategia a seguir es desarrollar las actividades identificadas a continuación de forma paralela al ciclo de vida del software que se esté desarrollando.

Negocio.

- Plan de tareas V&V.
- Evaluar Documentación.
- Plan de Proyecto.
- Plan de Gestión de la Configuración.
- Documento Visión.
- Pronóstico.
- Identificación de Riesgos.
- Plan de Calidad.
- Plan SQA.

Requisitos.

- Análisis de la Interfaz.
- Revisión e Inspección de los requerimientos de software.
- Plan V&V de generación de prueba y verificación del sistema.
- Documento de definición de requerimientos.
- Documento de análisis del sistema.
- Documento de diseño del sistema.
- Requisitos generales del proyecto.
- Evaluar los planes.
- Evaluar los requerimientos.
- Evaluar las herramientas de software.

Análisis y Diseño.

- Evaluación de Diseño de Software.
- Análisis de la Interfaz.

- Plan V&V de Generación de prueba y de verificación de Componente.
- Plan V&V de Generación de prueba y de verificación de Integración.
- Plan V&V de Generación de prueba y de verificación de Diseño.
- Revisión e Inspección de la arquitectura.
- Especificación de diseño.
- Documento de diseño de alto nivel.
- Documento de diseño de bajo nivel.
- Prototipos.
- Descripciones de diseño de las interfaces.
- Evaluar el diseño de software.
- Evaluar documento de análisis.
- Realizar revisiones y seguimiento de las desviaciones encontradas.
- Verificar avances del proyecto.

Implementación.

- Análisis de la interfaz.
- Evaluación de código fuente y documentación.
- V&V de la gestión de procedimientos de prueba y verificación.
- V&V de prueba y ejecución.
- V&V de verificación de componentes.
- Conformar línea base del proyecto y documento asociado.
- Código fuente del programa.
- Código objeto y ejecutable.
- Documento de vercionamiento.
- Documento de diseño de bases de datos.
- Control de código.
- Reportes de errores y actividades correctivas.
- Evaluar desarrollo del software.
- Evaluar las desviaciones.
- Revisiones e intervenciones.
- Verificar avances del proyecto.

Prueba.

- V&V de la ejecución y verificación de las pruebas de integración.
- V&V de pruebas de ejecución y verificación de aceptación.
- V&V de prueba de ejecución y verificación del sistema.
- V&V de la documentación.
- Realizar auditorías de configuración física.
- Realizar auditorías de configuración funcional.
- Plan y procedimientos de prueba.
- Casos de prueba y resultados registrados.
- Planes de prueba de unidad.
- Datos de prueba.
- Especificaciones de prueba de unidad.
- Actividades de testeo.
- Actualizar plantillas de métricas de calidad.
- Realizar auditorías de procesos.
- Evaluar procesos de SCM.
- Evaluar proceso de prueba de unidad.
- Pruebas de integración.
- Revisiones del proyecto y auditorías.
- Revisiones técnicas.
- Verificar avances del proyecto.
- Inspecciones de software.

Despliegue.

- V&V de mantenimiento.
- V&V de generación de informe final.
- Auditoría de la configuración de instalación.
- Evaluación de las migraciones.
- Análisis de criticidad.
- Revisiones internas.
- Datos de prueba.

- Datos de proyecto.
- Manuales de usuario.
- Documentos de mantenimiento.
- Informes de problemas de software.
- Peticiones de mantenimiento.
- Órdenes de cambio de Ingeniería.
- Control de medios.
- Colección de registros, mantenimiento y retención.
- Entrenamiento.
- Administración de riesgos.
- Control de proveedores.
- Evaluar las instalaciones.
- Plan de despliegue.

2.5 Modelo de actividades SQA:

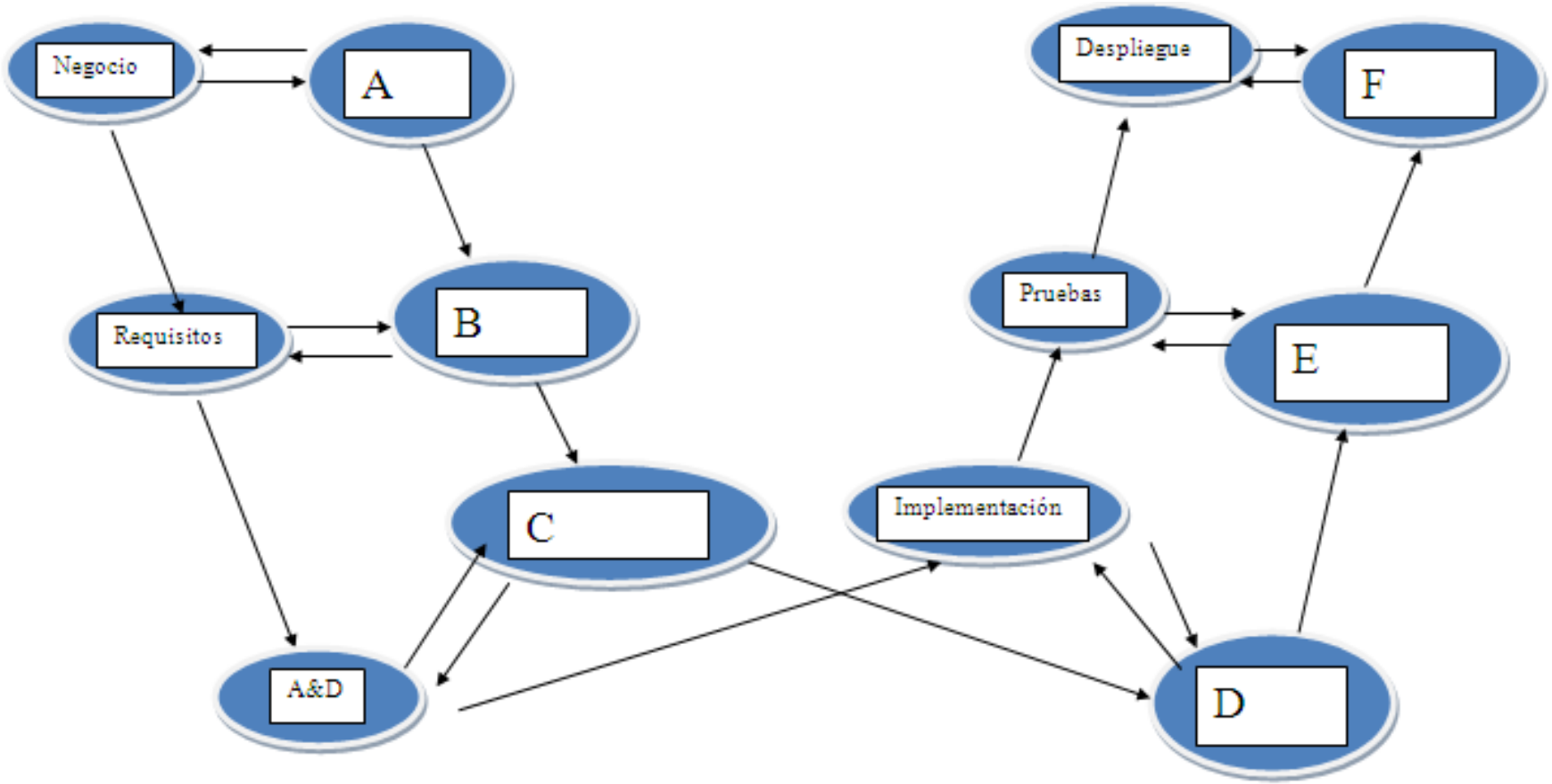


Figura 9. Modelo de actividades SQA.

En el modelo anteriormente mostrado se representa la estrategia (definida en el modelo con letras) que deben seguir los desarrolladores de software de la facultad de informática de la Universidad de Cienfuegos.

Quedan definidas a continuación cada una de las actividades para el aseguramiento de la calidad de forma paralela al ciclo de vida del software.

A:

- Plan de Proyecto
- Documento Visión.
- Pronóstico.
- Identificación de Riesgos.
- Plan de Proyecto.
- Plan de Calidad.
- Plan SQA.

B:

- Requisitos generales
- Evaluar los planes.
- Evaluar los requerimientos.
- Evaluar las herramientas de software.

C:

- Especificación Análisis y Diseño
- Evaluar el diseño de software.
- Evaluar documento de análisis.
- Realizar revisiones y seguimiento de las desviaciones encontradas.
- Verificar avances del proyecto.

D:

- Control de Código
- Reportes de errores y actividades correctivas.
- Evaluar desarrollo del software.
- Evaluar las desviaciones.
- Revisiones e intervenciones.

- Verificar avances del proyecto.

E:

- Actividad de testeo
- Actualizar plantillas de métricas de calidad.
- Realizar auditorías de procesos.
- Evaluar procesos de SCM.
- Evaluar proceso de prueba de unidad.
- Pruebas de integración.
- Revisiones del proyecto y auditorías.
- Revisiones técnicas.
- Verificar avances del proyecto.
- Inspecciones de software.

F:

- Plan de Despliegue
- Control de medios.
- Colección de registros, mantenimiento y retención.
- Entrenamiento.
- Administración de riesgos.
- Control de proveedores.
- Evaluar las instalaciones.

2.6 Recomendaciones a la Guía:

- Las actividades a revisar deben ser controladas desde sus inicios.
- Avisar al responsable de cada artefacto cuándo debe comenzar a construirse y qué cosas van a ser evaluadas del mismo.
- Si en las revisiones se detectan desviaciones que implican fuertes cambios que a criterio del responsable de SQA impactan en el desarrollo del proyecto, el informe acerca de estas debe dirigirse al Administrador y Arquitecto para que ese impacto se analice y se tenga en cuenta en los planes futuros del proyecto.

- Asegurar que el responsable del proyecto esté al tanto de todos los movimientos que ocurren en el mismo sino el rol SQA pasa a ser una parte insignificante dentro del proceso de desarrollo.
- Tener presente en todo momento los estándares de calidad que muy a menudo suelen ser olvidados.

2.7 Conclusiones Parciales.

En este capítulo se profundizan las actividades de SQA, V&V y gestión de la configuración, así como la creación de un modelo de actividades SQA. Trayendo consigo la creación de la guía SQAFINF que proporciona un mayor entendimiento y una mejor práctica del aseguramiento de la calidad de software dentro del proceso de desarrollo de la facultad de informática de la UCF, garantizando con ella que los productos o software sean entregados con mayor calidad y lo más cercanos posible a las especificaciones de los clientes.

CAPÍTULO 3: Proceso de evaluación del contenido de la guía SQAFINF.

3.1 Introducción.

En este capítulo se analizaran diferentes técnicas de validación para luego seleccionar una de ellas y utilizarla en el análisis de los resultados, obtenidos de las encuestas realizadas a especialistas de aseguramiento de la calidad de software, utilizando el programa estadístico SPSS v.15.0 para el procesamiento de estos datos. Además se utilizará la prueba de hipótesis para la comparación de los resultados, apreciando así si existe acuerdo o no entre los especialistas, respecto a la guía SQAFINF propuesta.

3.2 Técnicas de validación.

En la actualidad existen diversas técnicas de validación. Una de ellas es el método de Delphi muy utilizado para cuando no se tienen datos o estos son muy pocos acerca de la guía de la cual se está tratando. En este método, se selecciona un grupo de expertos los cuales deben llegar a un consenso en las respuestas que den acerca de una serie de preguntas que se les plantean. El método Delphi excluye las discusiones cara a cara entre los miembros del grupo (Ujaen, 2008).

Otra de las técnicas es el Test de Turing donde a un experto, o grupo de expertos, se le presenta resúmenes o informes de resultados de la guía, a los que se les ha dado el mismo formato. Estos informes se reparten aleatoriamente a los ingenieros de software y especialistas en el tema, para ver si son capaces de discernir cuáles son los reales de la guía y cuáles la imitación. Si los expertos no son capaces de distinguir entre ambos, se puede concluir que no hay evidencias para considerar inadecuada la guía. Si descubren diferencias las respuestas sobre lo que encuentran inconsistente se puede utilizar para realizar mejoras en la guía (Ujaen, 2008).

Para llevar a cabo el desarrollo de la evaluación de la guía se tomó como aspecto fundamental, el enfoque teórico citado de Pérez Gómez, donde se definen dos grandes paradigmas y dentro de ellos se ubican los modelos de evaluación

experimental y cualitativo (Gutiérrez, 2007). Esto lleva a decir que la evaluación de la guía para facilitar el aseguramiento de la calidad de software en la facultad de informática de la UCF debe abordarse desde su propio diseño y desde su práctica profesional para así comprobar el grado de veracidad y realización que contiene su información. Este análisis conduce a realizar diagnósticos del contenido y propone ideas más actualizadas en los diferentes temas que contiene la guía en lo que a conocimientos y habilidades se refiere (Davila Cabo de Villa, 2007).

Todo el contenido de la misma contiene conocimientos y habilidades pero no con la misma intensidad, por lo que estos aspectos se analizan desde ángulos diferentes dado el desarrollo y las funciones que realizan.

En la documentación analizada se encuentra como tendencia para decidir el valor del contenido que existen criterios, pautas, puntos de vista que permiten la evaluación del contenido de procesos, libros, programas, ciencias, entre otros. En la bibliografía estudiada se dice que para que la muestra sea significativa se deben tomar como mínimo 11 expertos (Sampieri, 2003) por lo que para la elaboración de esta se tuvieron en cuenta 12. Para evaluar la guía se siguieron los criterios de diferentes autores que permiten medir el contenido de la misma. La guía de Aseguramiento de la Calidad de Software para la Facultad de Informática será evaluada según los criterios planteados por (Sampieri, 2003) con algunas adecuaciones de los mismos (Davila Cabo de Villa, 2007).

3.3 Criterios para evaluar la teoría de la guía.

1.- Capacidad de descripción, explicación y predicción.

2.- Consistencia lógica.

3.- Perspectiva.

4.- Fructificación (heurística).

5.- Parsimonia.

1) Capacidad de descripción, explicación y predicción:

Describir implica varias cuestiones: definir las condiciones en que se presenta y las distintas maneras en que puede manifestarse (Soto, 2008).

Explicar, significa incrementar el entendimiento de las causas del fenómeno, y también se refiere “a la prueba empírica” de las proposiciones de las teorías (Soto, 2008).

Predicción, está asociado con el segundo término de explicación, que depende de la evidencia empírica de las proposiciones de la teoría (Soto, 2008).

Cuanta más evidencia empírica apoye a la teoría, mejor podrá describir, explicar y predecir el fenómeno o fenómenos estudiados por ella.

2) Consistencia lógica:

Una teoría debe ser lógicamente consistente. Es decir, las proposiciones que la integran deberán estar interrelacionadas (no pueden contener proposiciones sobre fenómenos que no están relacionados entre si), ser mutuamente excluyentes (no puede haber repetición o duplicación), ni caer en contradicciones internas o incoherencias (Soto, 2008).

3) Perspectiva:

Se refiere al nivel de generalidad. Una teoría posee más perspectiva cuanta mayor cantidad de fenómenos explique y mayor número de aplicaciones admita (Soto, 2008).

4) Fructificación (heurística):

Es la capacidad que tiene una teoría de generar nuevas interrogantes y descubrimientos. Las teorías que originan, en mayor medida, la búsqueda de nuevos conocimientos son las que permiten que una ciencia avance (Soto, 2008).

5) Parsimonia:

Una teoría parsimoniosa es una teoría simple, sencilla. Este no es un requisito, sino una cualidad deseable. Indudablemente las teorías que pueden explicar uno o varios

fenómenos en unas cuantas proposiciones sin omitir ningún aspecto son más útiles que las que necesitan un gran número de proposiciones para ello (Soto, 2008).

3.4 Análisis de las herramientas utilizadas para aplicar los criterios de evaluación.

3.4.1 Procesamiento estadístico.

Para consultar a los especialistas se utilizó como método el sistema de expertos no estructurados llamados nominales. A los cuales se les pidió evaluar el contenido de la guía para facilitar el aseguramiento de la calidad en la facultad de informática según los criterios escogidos. Se evaluó con una escala de importancia de cinco alternativas de respuestas donde se mide la importancia del criterio desde en ‘total desacuerdo’ y hasta ‘total acuerdo’. Donde la alternativa tres es ‘no aplica’ la cual significa que alguna de las preguntas que se realizaron no se ajustan a la guía y se explica el por qué.

Para el procesamiento de la información se utilizó el paquete estadístico SPSS V.15.0 comenzando con un análisis detallado de la información. Las encuestas fueron diseñadas cumpliendo con que contaran con preguntas sencillas y claras, no muy extensas, legibles y que motivaran al encuestado (BECANA, 2007). En el (Anexo 8) se muestra un ejemplo de la encuesta realizada.

La aplicación de la encuesta de forma personal y anónima garantizó que la cantidad de encuestas entregadas fueran recogidas y respondidas satisfactoriamente. Para determinar la fiabilidad de estas encuestas se utilizó el coeficiente Alpha de Cronbach que se encuentra en el SPSS. Esta fiabilidad se refiere al grado en que las puntuaciones obtenidas en las diferentes preguntas del cuestionario se encuentren interrelacionadas. Si una encuesta o cuestionario es fiable debe mostrar resultados estables cuando es aplicado por diferentes personas y en diferentes circunstancias.

El coeficiente Alpha oscila entre 0 y 1. Es uno de los más utilizados para medir la fiabilidad de una escala. Se dice que mientras más próximo a 1 la fiabilidad es

mayor. En el análisis realizado el valor de Alpha es de 0.852 por lo que permite ver que la puntuación entre las variables está interrelacionada. (Anexo 9)

Para continuar con el análisis se realizó la prueba no paramétrica W de Kendall con el objetivo de demostrar estadísticamente que existe acuerdo entre los evaluados. Esta prueba cuenta con la hipótesis H_0 que plantea que no hay acuerdo, contra la hipótesis H_1 que sí plantea que hay acuerdo entre los expertos. Para decidir cuál hipótesis aceptar se debe comparar el nivel de significación con la significación asintótica del estadígrafo, si esta última es menor que la significación entonces se acepta H_1 (Anexo 3).

3.4.2 Análisis de los resultados de las encuestas realizadas a los expertos.

Para el procesamiento de los datos con el SPSS, de acuerdo a la encuesta realizada a los expertos se definieron las siguientes variables (Anexo 10):

X1-Redacción.

X2-Lenguaje adecuado al nivel de enseñanza.

X3-Capacidad de descripción, explicación y predicción.

X4-Consistencia Lógica.

X5-Motivación.

X6-Confiabilidad Psicopedagógica.

X7-Expectativas.

Estas variables fueron codificadas asignándole etiquetas correspondientes a los valores que tienen las diferentes alternativas de respuesta en cada pregunta. Se encuestaron 12 especialistas 10 en Desoft y 2 en la facultad de informática todos con experiencia en el tema de calidad de software.

- Criterio 1 capacidad de descripción, explicación y predicción:

El 75% de los encuestados otorgó una puntuación de 5, mientras que el 25% restante otorgó una puntuación de 4 por lo que se considera como satisfactorio este criterio dentro de la guía SQAFINF, según la opinión de los especialistas.

Criterio#1	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Capacidad de descripción, explicación y predicción	9	75	3	25					

- Criterio 2 consistencia lógica:

El 66,7% de los especialistas otorgó una puntuación de 5, mientras que el 33,3% restante otorgó una puntuación de 4 por lo que se considera como satisfactorio este criterio dentro de la guía SQAFINF según la opinión de los especialistas.

Criterio#2	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Consistencia lógica	8	66,7	4	33,3					

- Criterio 3 perspectiva:

El 83,3% de los especialistas respecto a las expectativas de la guía otorgó una puntuación de 5, mientras que el 16,7% restante otorgó una puntuación de 4 por lo que se considera como satisfactorio, siendo este criterio dentro de la guía SQAFINF según la opinión de los especialistas el mejor evaluado.

Criterio#3	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Expectativa	10	83,3	2	16,7					

- Criterio 4 Heurística:

El 8,3% de los especialistas, respecto a la motivación, otorgó una puntuación de 5, el 66,7% otorgó una puntuación de 4 y el 25% restante otorgó una puntuación de 3, por lo que se considera como menos satisfactorio, siendo este criterio dentro de la guía SQAFINF, según la opinión de los especialistas, el peor evaluado.

El 58,3% de los especialistas, respecto al lenguaje adecuado, otorgó una puntuación de 5, mientras que el 41,7% restante otorgó una puntuación de 4, por lo que se considera como satisfactorio este criterio dentro de la guía SQAFINF, según la opinión de los especialistas.

El 33,3% de los especialistas, respecto a la confiabilidad psicopedagógica, otorgó una puntuación de 5, mientras que el 66,7% restante otorgó una puntuación de 4, por lo que se considera como satisfactorio este criterio dentro de la guía SQAFINF, según la opinión de los especialistas.

Criterio#4	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Motivación	1	8,3	8	66,7	3	25			
Lenguaje adecuado	7	58,3	5	41,7					
Confiabilidad psicopedagógica	4	33,3	8	66,7					

- Criterio 5 Parsimonia:

El 75% de los especialistas, respecto a la redacción, otorgó una puntuación de 5, mientras que el 25% restante otorgó una puntuación de 4, por lo que se considera como satisfactorio este criterio dentro de la guía SQAFINF, según la opinión de los especialistas.

Criterio#5	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Redacción	9	75	3	25					

De acuerdo a la prueba de concordancia W de Kendall que se realizó tomando en cuenta la hipótesis H_0 (No hay acuerdo entre los especialistas) y H_1 (Hay acuerdo entre los especialistas) para así decidir cuál de las hipótesis aceptar, se adoptó un nivel de significación de 0.05 mayor que la significación asintótica calculada en el SPSS de 0.01, por lo que se puede concluir que la hipótesis a aceptar es H_1 y sí existe un acuerdo entre las opiniones de los expertos (Anexo 11).

Los rangos obtenidos en la prueba de Kendall permiten organizar los criterios evaluados de acuerdo a su importancia en una escala descendiente de la siguiente forma:

- Expectativas (4,92).
- Redacción (4,67).
- Capacidad de descripción, explicación y predicción (4,63).
- Consistencia lógica (4,33).
- Lenguaje adecuado (4,04).
- Confiabilidad Psicopedagógica (3,25).
- Motivación (2,17).

Luego de haber concluido con el análisis de los resultados de la guía se puede observar que la evaluación de los expertos osciló entre 4 (de acuerdo) y 5 (Total acuerdo), exceptuando en la motivación que se obtuvieron algunos resultados de 3 (No Aplica) por lo que en general se puede decir que la opinión de los especialistas respecto a la guía fue coincidente en la mayoría de los aspectos, lo que permite ver la importancia que puede tener la misma. Teniendo en cuenta que en el aspecto de motivación se pueden hacer mejoras para futuras modificaciones en la guía.

3.5 Conclusiones Parciales

En este capítulo luego de haber realizado todo el análisis estadístico de las encuestas realizadas a especialistas de calidad de software para evaluar la guía, se llegó a la conclusión de que la misma está lista para ser utilizada por los desarrolladores de software de la facultad de informática en la Universidad de Cienfuegos o por cualquier otro que desee usarla. Debido a que los resultados estadísticos obtenidos muestran que las opiniones de los expertos coinciden en la mayoría de los aspectos con puntuaciones entre 4 (de acuerdo) y 5 (total acuerdo).

Conclusiones Generales

Después de haber realizado esta investigación se arribó a las siguientes conclusiones:

1- El estudio de estándares, metodologías, normas, modelos y herramientas de aseguramiento de la calidad fue de gran ayuda para la elaboración de la guía cumpliendo con los objetivos y tareas propuestos.

2- La Utilización de la metodología RUP jugó un papel fundamental ya que a parte de ser la más utilizada en la facultad de informática sirvió de ayuda para llevar paralelamente a sus flujos de trabajo las actividades de V&V, gestión de la configuración y aseguramiento de la calidad.

3- La guía permite que los desarrolladores de la facultad de informática tengan un mayor conocimiento respecto al aseguramiento de la calidad de software, proporcionando que los productos software realizados en ella salgan con una mayor calidad y sean entregados en el tiempo establecido.

4- La creación del modelo de actividades de SQA permite que los desarrolladores de software vean claramente como estas actividades se desarollan paralelamente a los flujos de trabajo de RUP.

Recomendaciones

1- Aplicar la guía SQAFINF en el proceso de desarrollo de los productos informáticos que se están elaborando en los grupos científicos estudiantiles de la facultad de informática.

2- Realizar un análisis estadístico de los resultados obtenidos al aplicar la guía en los grupos científicos estudiantiles.

3- Crear un grupo en la facultad de informática de la UCF que se encargue de llevar el SQA.

Referencias bibliográficas

(Administración. 2007.) *VerificacionyValidacion.* [En línea] 2007. capítulo 22 de V&V,clases de académicos.. http://www.eici.ucm.cl/Academicos/.../cap22verificacionyvalidacion.ppt -.

(Arca. 2007.) Coactivate. *Documento Visión.* [En línea] 11 de julio de 2007. http://www.coactivate.org/projects/.../documento-de-visiA-n.

(BECANA, RUT BOSQUE. 2007). *Orientaciones para realizar Evaluaciones.* 2007. se puede encontrar en http://www.profes.net.

(Bravo., Alexander Oré. 2008.) CalidadySoftware. [En línea] 13 de de Abril de 2008. http://www.calidadysoftware.com/otros/introduccion_c....

(Carlos, Universidad Rey Juan. 2007.) Kybele. *Calidad delsoftware.* [En línea] 2007. http://www.kybele.escet.urjc.es/Documentos/.../Calidad%20del%20software.ppt.

(CHACÓN, JULIO CÉSAR RUEDA. marzo 2006.) *APLICACIÓN DE LA METODOLOGÍA RUP PARA EL DESARROLLO RÁPIDO DE APLICACIONES BASADO EN EL ESTÁNDAR J2EE.* Guatemala : UNIVERSIDAD DE SAN CARLOS DE GUATEMALA, marzo 2006. págs. 1-2, TESIS PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS Y SISTEMAS.

(Corporativo. 2008.) MiTecnologico. *Definicion y Proposito Sqa.* [En línea] 2008. http://www.MiTecnologico.com.

(Davila Cabo de Villa, Evangelina. 2007.) *Evaluación del contenido del manual para Enfermeros Auxiliares de Anestecia.* Centro de Estudios de Didáctica de la Educación Superior, Universidad de Cienfuegos. 2007. Tesis de Maestría. se puede encontrar en http://www.biblioteca.ucf.edu.cu.

(Desoft. 2008.) Desoft Exelencia en software. [En línea] 2008. http://www.desoft.com.cu.

(digital, biblioteca. 2009.) bdigital. *MODELO ISO/SPICE.* [En línea] 2009. http://www.bdigital.eafit.edu.co/bdigital/PROYECTO/P005.../capitulo1.pdf.

(dptoInformática. 2006.) mundodescargas. *configuracion de software.* [En línea] 2006. http:/http://www.mundodescargas.com/apuntes-trabajos/informatica/decargar_configuracion-de-software.pdf.

(Dykstra, Josiah. 2008.) destiny. [En línea] 2008. http://www.acm.org/crossroads/espanol/...3/destiny.html.

(enrike. 2007.) Plan de calidad. [En línea] 2007. www.enrike.com.mx/downs/PlanSQA_ejemp.pdf.

(Especialista, software. 2007.) hista internacional. *Gestión de laConfiguración del Software.* [En línea] 2007. http://www.histaintl.com/soluciones/configuracion/configuracion.php.

(Figueroa, María Antonieta Abud. 2006.)evistaupiicsa. *Calidad en la Industria del Software. La Norma ISO-9126.* [En línea] 2006. http://www.revistaupiicsa.20m.com/Emilia/RevEneAbr04/Antonieta1.pdf.

(García, Ibon Gotxi. 2002.) *Diseño de Alto Nivel.* Bilbao : s.n., 2002. Ponencia. Tomado de la Escuela superior de Ingenieros de Bilbao.

(Gutiérrez, Lidia. 2007.) *PARADIGMAS CUANTITATIVO Y CUALITATIVO.* 2007. Ponencia. se puede encontrar en http://biblioteca.idict.villaclara.cu/UserFiles/File/METODOLOGIA%20DE%20INVESTIGACION/PARADIGMAS%20CUANTITATIVO%20Y%20CUALITATIVO.doc.

(Gutiérrez. 2006.) *PARADIGMAS CUANTITATIVO Y CUALITATIVO.* 2006.

(IEEE P1012/D12. 2004.) *Draft Standard for Software Verification and Validation.* New York, : s.n., 2004. pág. 100. IEEE 1012.

(InfoMedia. 2010.) *Documento de Diseño del Sistema.* Ingeniería de Software . 2010.

(Isi. 2006.)*Verificación y Validación.* 2006.

(José, María. 2006.) Isi. *plan de gestión de laconfiguración.* [En línea] 2006. http://www.lsi.ugr.es/.../gestion/...plantilla/plan_gestion_configuracion_dp.doc.

(Kruchten, P. 2000.) *The Rational Unified Process:An Introduction.* [ed.] Addison Wesley. 2000.

(Kynetia. 2007.) Kynetia. *Calidad.* [En línea] 2007. http://www.kynetia.es/calidad/metodologia.html.

(Lauro, Soto. 2008.) mitecnologico. *Especificaciones De Requerimientos.* [En línea] 2008. creado en México Encenada. http://www.mitecnologico.com/.../EspecificacionesDeRequerimientos.

(Letelier. 2007.) *Introducción a RUP.* Valencia : Universidad Politécnica de Valencia, 2007.

(Llerena, Gloria María Guerrero. 2008.) *EXPERIENCIAS EN LA IMPLANTACIÓN DE UN SISTEMA DE GESTIÓN DE LA CALIDAD PARA ELPROCESO DE PRODUCCIÓN DE SOFTWARE.* 2008. Ponencia. encontrada en http://www.citmatel.cu.

(Losada, Roberto. 2005.) udistrital. *IDENTIFICACION Y ADMINISTRACION RIESGOS.* [En línea] 2005. http://www.udistrital.edu.co/.../IdentificacionAdministracionRiesgos.pdf.

(Lovelle, Juan Manuel Cueva. 2005.)*Calidad del Software.* s.l. : Univerisidad de Oviedo, 2005. págs. 2-4. se puede encontrar en http://www.uniovi.es.

(Universidad. 2008.) biblioteca virtual. *Problemas y costos del Aseguramiento de la Calidad en el Software.* [En línea] 2008. http://www.eumed.net/libros/2008a/351.

(Pantaleo, Guillermo, Forradellas, Patricia y Rogers, Juan. 2009.) *El modelo CMM/CMMI - Cómo garantizar el éxito del proceso de mejoras en las organizaciones, superando los conflictos y tensiones generados por su implementación.* 2009.

(Pressman, Roger S. 2005.) *Ingeniería de Software.Un enfoque Práctico.* Quinta Edición. 2005.

(Qualitrain. 2008.) Qualitrain. *Aseguramiento de la calidad de software.* [En línea] 2008. http://www.qualitrain.com.

(República, Universidad de la. 2003.) *Guía SQA.* Ingeniería de Software, facultad de Ingeniería, Instituto de Computación, México. 2003. se puede encontrar http:// www.fing.edu.uy/inco/cursos/ingsoft/tgs/2003/.../guia-SQA.doc.

(Rodríguez, Ana Isabel. 2006.)Proceso de verificación y validación independiente-Tecnologías aplicadas. [En línea] 2006. http://www.mityc.es/dgdsi/es-ES/Servicios/.../s05AnaIsabelRguez.pdf.

(Ruiz de Mendarozqueta, Álvaro y Fau, Carlos Alberto. 2008.) *Gestión de Calidad.* 2008. se puede encontrar en http://www.noqualityinside.com/.../QA2%20Herramientas%20para%20SQA.pdf.

(Sampieri, Hernández. 2003.) *Metodología de laInvestigación.* 2003.

(Sanchez, María A. Mendoza. 2006.) Informatizate. [En línea] 2006. http://www.informatizate.net/.../metodologias_de_desarrollo_de_software_07062004.html.

(software, Catedra de Ingeniería de. 2009.) *SQA Plan.* 2009. se puede encontrar en http://www.vast.uccs.edu/~tboult/SE/DOCS/SQA_Plan_Template.doc.

(Software, Dpto Ingeniería de. 2005.)dc.uba. *Preparación del Plan de Proyecto.* [En línea] 5 de septiembre de 2005. http://www.dc.uba.ar/.../PlanProyecto(incWBS)20050905_CONNOTAS.pdf.

(Solís., Manuel Calero. 2003.) willynet. *Una explicación de la programación extrema (XP).* [En línea] 2003. V Encuentro de usuarios xBase 2003 MADRID.. http://www.willydev.net/InsiteCreation/v1.0/descargas/.../explicaxp.pdf.

(Soto, Lauro. 2008.)mitecnologico. *Implicaciones Prácticas Investigación.* [En línea] 2008. (Citado el: 20 de mayo de 2010.) aqui se pueden encontrar los 5 criterios de evaluación. http://www.mitecnologico.com/Main/ImplicacionesPracticasInvestigacion.

(Ujaen. 2008.) Verificación y Validación. [En línea] 2008. http://www.di.ujaen.es/asignaturas/computacionestadistica/pdfs/tema6.pdf.

Bibliografía

Arca. 2007. Coactivate. *Documento Visión.* [En línea] 11 de julio de 2007. http://www.coactivate.org/projects/.../documento-de-visiA-n.

austral. [En línea] http://www.austral.edu.ar/.../pdf/PresentacionIdeaFactoryAustralFinal.ppt.

BECANA, RUT BOSQUE. 2007. *Orientaciones para realizar Evaluaciones.* 2007. se puede encontrar en http://www.profes.net.

bibliotecass. [En línea] http://biblioteca.usac.edu.gt/tesis/08/08_7691.pdf.

Bravo., Alexander Oré. 2008. CalidadySoftware. [En línea] 13 de de Abril de 2008. http://www.calidadysoftware.com/otros/introduccion_c....

calidadysoftware. [En línea] http://www.calidadysoftware.com/otros/introduccion_c...

Carlos, Universidad Rey Juan. 2007. Kybele. *Calidad delsoftware.* [En línea] 2007. http://www.kybele.escet.urjc.es/Documentos/.../Calidad%20del%20software.ppt.

CHACÓN, JULIO CÉSAR RUEDA. marzo 2006. *APLICACIÓN DE LA METODOLOGÍA RUP PARA EL DESARROLLO RÁPIDO DE APLICACIONES BASADO EN EL ESTÁNDAR J2EE.* Guatemala : UNIVERSIDAD DE SAN CARLOS DE GUATEMALA, marzo 2006. págs. 1-2, TESIS PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS Y SISTEMAS.

cinterfor. [En línea] http://cinterfor.org.uy/public/spanish/region/ampro/cinterfor/publ/papel/18/pdf/cap1.pdf.

conicyt. [En línea] http://www.ri.conicyt.cl/575/article-19460.html.

Corporativo. 2008. MiTecnologico. *Definicion y Proposito Sqa.* [En línea] 2008. http://www.MiTecnologico.com.

Davila Cabo de Villa, Evangelina. 2007. *Evaluación del contenido del manual para Enfermeros Auxiliares de Anestecia.* Centro de Estudios de Didáctica de la Educación Superior, Universidad de Cienfuegos. 2007. Tesis de Maestría. se puede encontrar en http://www.biblioteca.ucf.edu.cu.

Desoft. 2008. Desoft Exelencia en software. [En línea] 2008. http://www.desoft.com.cu.

digital, biblioteca. 2009. bdigital. *MODELO ISO/SPICE.* [En línea] 2009. http://www.bdigital.eafit.edu.co/bdigital/PROYECTO/P005.../capitulo1.pdf.

dptoInformática. 2006. mundodescargas. *configuracion de software.* [En línea] 2006. http:/http://www.mundodescargas.com/apuntes-trabajos/informatica/decargar_configuracion-de-software.pdf.

Dykstra, Josiah. 2008. destiny. [En línea] 2008. http://www.acm.org/crossroads/espanol/...3/destiny.html.

eafit. [En línea] http://www.eafit.edu.co/NR/rdonlyres/00132306-9035-46D9-ACBA-947E70EA8250/0/Boletin42AseguramientodelaCalidadAuditoria.pdf.

eici. [En línea] http://www.eici.ucm.cl/.../metricas%20tecnicas%20del%20software.ppt.

enrike. 2007. Plan de calidad. [En línea] 2007. www.enrike.com.mx/downs/PlanSQA_ejemp.pdf.

Especialista, software. 2007. hista internacional. *Gestión de laConfiguración del Software.* [En línea] 2007. http://www.histaintl.com/soluciones/configuracion/configuracion.php.

fabricadesoftware. [En línea] http://www.fabricadesoftware.cl/download.php?id=110&sid....

Figueroa, María Antonieta Abud. 2006. revistaupiicsa. *Calidad en la Industria del Software. La Norma ISO-9126.* [En línea] 2006. http://www.revistaupiicsa.20m.com/Emilia/RevEneAbr04/Antonieta1.pdf.

García, Ibon Gotxi. 2002. *Diseño de Alto Nivel.* Bilbao : s.n., 2002. Ponencia. Tomado de la Escuela superior de Ingenieros de Bilbao.

Gutiérrez, Lidia. 2007. *PARADIGMAS CUANTITATIVO Y CUALITATIVO.* 2007. Ponencia. se puede encontrar en http://biblioteca.idict.villaclara.cu/UserFiles/File/METODOLOGIA%20DE%20INVESTIGACION/PARADIGMAS%20CUANTITATIVO%20Y%20CUALITATIVO.doc.

Gutiérrez. 2006. *PARADIGMAS CUANTITATIVO Y CUALITATIVO.* 2006.

InfoMedia. 2010. *Documento de Diseño del Sistema.* Ingeniería de Software . 2010.

ingenierosoftware. [En línea] http://www.ingenierosoftware.com/.../cmm-cmmi.php.

Isi. 2006. *Verificación y Validación.* 2006.

José, María. 2006. Isi. *plan de gestión de laconfiguración.* [En línea] 2006. http://www.lsi.ugr.es/.../gestion/...plantilla/plan_gestion_configuracion_dp.doc.

Kruchten, P. 2000. *The Rational Unified Process:An Introduction.* [ed.] Addison Wesley. 2000.

Kynetia. 2007. Kynetia. *Calidad.* [En línea] 2007. http://www.kynetia.es/calidad/metodologia.html.

Lauro, Soto. 2008. mitecnologico. *Especificaciones De Requerimientos.* [En línea] 2008. creado en México Encenada. http://www.mitecnologico.com/.../EspecificacionesDeRequerimientos.

Letelier. 2007. *Introducción a RUP.* Valencia : Universidad Politécnica de Valencia, 2007.

Llerena, Gloria María Guerrero. 2008. *EXPERIENCIAS EN LA IMPLANTACIÓN DE UN SISTEMA DE GESTIÓN DE LA CALIDAD PARA ELPROCESO DE PRODUCCIÓN DE SOFTWARE.* 2008. Ponencia. encontrada en http://www.citmatel.cu.

Losada, Roberto. 2005. udistrital. *IDENTIFICACION Y ADMINISTRACION RIESGOS.* [En línea] 2005. http://www.udistrital.edu.co/.../IdentificacionAdministracionRiesgos.pdf.

Lovelle, Juan Manuel Cueva. 2005. *Calidad del Software.* s.l. : Univerisidad de Oviedo, 2005. págs. 2-4. se puede encontrar en http://www.uniovi.es.

Málaga, Universidad. 2008. biblioteca virtual. *Problemas y costos del Aseguramiento de la Calidad en el Software.* [En línea] 2008. http://www.eumed.net/libros/2008a/351.

Mgar. [En línea] http://mgar.net/soc/isointro.htm.

monografias. [En línea] http://www.monografias.com/...gestion-configuracion.../implementacion-herramienta-gestion-configuracion-subversion.shtml.

mygnet. [En línea] http://www.mygnet.net/articulos/software/1082/.

navegapolis. [En línea] http://www.navegapolis.net/content/view/371/55/.

opticalres. [En línea] http://www.opticalres.com/.../ChoosingOpticalDesignSoftware_ESP.pdf.

Pantaleo, Guillermo, Forradellas, Patricia y Rogers, Juan. 2009. *El modelo CMM/CMMI - Cómo garantizar el éxito del proceso de mejoras en las organizaciones, superando los conflictos y tensiones generados por su implementación.* 2009.

Pressman, Roger S. 2005. *Ingeniería de Software.Un enfoque Práctico.* Quinta Edición. 2005.

Qualitrain. 2008. Qualitrain. *Aseguramiento de la calidad de software.* [En línea] 2008. http://www.qualitrain.com.

regax. [En línea] http://regax.ei.uvigo.es/reuniones/gwai.pdf.

República, Universidad de la. 2003. *Guía SQA.* Ingeniería de Software, facultad de Ingeniería, Instituto de Computación, México. 2003. se puede encontrar http:// www.fing.edu.uy/inco/cursos/ingsoft/tgs/2003/.../guia-SQA.doc.

revistaupiicsa. [En línea] http://www.revistaupiicsa.20m.com/Emilia/RevEneAbr04/Antonieta1.pdf.

Rodríguez, Ana Isabel. 2006. Proceso de verificación y validación independiente-Tecnologías aplicadas. [En línea] 2006. http://www.mityc.es/dgdsi/es-ES/Servicios/.../s05AnaIsabelRguez.pdf.

Ruiz de Mendarozqueta, Álvaro y Fau, Carlos Alberto. 2008. *Gestión de Calidad.* 2008. se puede encontrar en http://www.noqualityinside.com/.../QA2%20Herramientas%20para%20SQA.pdf.

Sampieri, Hernández. 2003. *Metodología de laInvestigación.* 2003.

Sanchez, María A. Mendoza. 2006. Informatizate. [En línea] 2006. http://www.informatizate.net/.../metodologias_de_desarrollo_de_software_07062004.html.

slideshare. [En línea] http://www.slideshare.net/.../gestin-del-cambio-del-software.

software, Catedra de Ingeniería de. 2009. *SQA Plan.* 2009. se puede encontrar en http://www.vast.uccs.edu/~tboult/SE/DOCS/SQA_Plan_Template.doc.

Software, Dpto Ingeniería de. 2005. dc.uba. *Preparación del Plan de Proyecto.* [En línea] 5 de septiembre de 2005. http://www.dc.uba.ar/.../PlanProyecto(incWBS)20050905_CONNOTAS.pdf.

Solís., Manuel Calero. 2003. willynet. *Una explicación de la programación extrema (XP).* [En línea] 2003. V Encuentro de usuarios xBase 2003 MADRID.. http://www.willydev.net/InsiteCreation/v1.0/descargas/.../explicaxp.pdf.

Soto, Lauro. 2008. mitecnologico. *Implicaciones Prácticas Investigación.* [En línea] 2008. [Citado el: 20 de mayo de 2010.] aqui se pueden encontrar los 5 criterios de evaluación. http://www.mitecnologico.com/Main/ImplicacionesPracticasInvestigacion.

tenstep. [En línea] http://www.tenstep.es/.../0.0.1.1_Modelo_Madurez_MMC.htm.

todoexpertos. [En línea] http://www.todoexpertos.com/categorias/tecnologia-e-internet/software-y-aplicaciones/respuestas/2225912/gestion-de-la-configuracion.

udec. [En línea] http://www.inf.udec.cl/~revista/ediciones/edicion9/febles.pdf.

Ujaen. 2008. Verificación y Validación. [En línea] 2008. http://www.di.ujaen.es/asignaturas/computacionestadistica/pdfs/tema6.pdf.

Anexos

Anexo 1 – Reporte del análisis del proceso de auditoría a los requisitos

reporte del análisis del proceso de auditoría a los requisitos
Proyecto: **Fecha:** **Preparado por:**
Procedimientos: **Part 1.** Requerimientos de software ____los requerimientos de software están documentados e incluyen la matriz de rastreo. ____el SRS esta bajo al administración de la configuración ____los planes de desarrollo de software y otros productos son modificados si se modifica el SRS para que estos sean consistentes. ___se han usado mediciones para determinar el estado de la administración de requisitos.

Anexo 2 – Evaluación de herramientas de software.

EVALUACION DE HERRAMIENTA DE SOFTWARE

SQA: _____ FECHA: _____

Herramienta de software evaluada:

Métodos o criterio usados en la evaluación:

Resultados de la evaluación:

Acciones correctivas recomendadas

Acciones correctivas tomadas

Anexo 3 – Checklist del proceso de auditorías al diseño de software

Checklist del proceso de auditorías al diseño de software
Proyecto: **Fecha:** **Preparado por:**
Procedimientos: **Parte 1. diseño** ____los documentos de diseño y la matriz de rastreo estén listos. ____las caminatas evalúan si todos los requerimientos están plasmadas en el diseño, a si como el de tratar de encontrar errores ____el diseño esta actualizado con respecto a los últimos cambios en los requisitos. ____los cambios en el diseño seguidas, evaluadas. ____el diseño de software es consistente con la metodología de diseño propuesta en el Proceso de Desarrollo de Software (SDP).

Anexo 4 – Verificar avances del proyecto

FASES DE DESARROLLO	Productos de software	Auditorías y revisiones
Requisitos de software	(1) ERS	(1) revisión de especificación de software (2)auditorías (3) revisión de administrador de proyecto (4) revisiones a par
Diseño de software	Diseño de software	(1)auditorías (2) revisión de administrador de proyecto (3) revisiones a par
Desarrollo de software	Productos de software	(1) auditorías (2) revisión de administrador de proyecto (3) revisiones a par
pruebas	Documento de pruebas	(1) auditorías (2) revisión de administrador de proyecto (3) revisiones a par

Anexo 5 – Reporte de procesos de auditoría

Reporte de procesos de auditoria

NÚMERO DE REPORTE:_____

LÍDER DE LA AUDITORÍA:_____

FECHA DREPORTE:_____

EQUIPO DE AUDITORÍA:_____

_____

NOMBRE DEL PROYECTO:_____

FECHA DE LA AUDITORÍA:_____

PROCESOS/PROCEDIMIENTOS AUDITADOS:_____

CHECKLIST USADO(S): _____

RESULTADOS DELA AUDITORIA: (SELECCIONAR UNA)

_____ Procesos/Procedimientos Aceptado

_____ Procesos/Procedimientos Condicionalmente aceptados
Condiciones:

_____ Procesos/Procedimientos no aceptados
Condiciones:

_____

_____ELEMENTO (ELE):

ELE # TÍTULO	ASIGNADO A :	FECHA DE ASIGNACIÓN:	FECHA DE FIN
:---			

_____

_____

_____

ACCCIÓN DE CORRECCIÓN:

_____

ESTADO: APPROVADO CANCELADO APLAZADO

ADMINISTRADOR DE PROYECTO: FECHA:

Anexo 6 – Proceso de auditoría a la administración de la configuración

Checklist para el proceso de auditoría a la administración de la configuración.
Proyecto: **Fecha:** **Preparado por:**
Procedimientos: **Parte 1.** SCM Plan ____las secciones de este documento están organizados de acuerdo al estándar. ____existe un grupo designado para implementar este plan. ____el documento del plan SCM es usado como base para todas las actividades de administración de la configuración ____los cambios a la línea base son manejadas de acuerdo al plan de SCM ____se ha establecido un repositorio para guardar la línea base. ____el acceso a los elementos de la línea base alojados en el repositorio son de acuerdo a los procedimientos establecidos en el plan SCM. ____los productos de software alojados en el repositorio deben de estar identificados como elementos de configuración en el plan SCM. ____los cambios a la línea base son controlados de acuerdo a los establecidos en el plan SCM

Checklist para el proceso de auditoría a la administración de la configuración.
Proyecto: **Fecha:** **Preparado por:**
Parte 2. Identificación de la configuración ____la línea base puede ser identificada ____si, describe la metodología usada para identificar la línea base ____cada elemento de la configuración pueden ser identificadas a si como sus componentes ____si, describe los métodos para identificarlos ____un método es usado para identificar en nombre, versión, cambio de estado, y cualquier otro detalle de identificación de cada elemento liberado. ____si, describe el método usado.

Checklist para el proceso de auditoría a la administración de la configuración.
Proyecto: **Fecha:** **Preparado por:**
Parte 3. Propuesta de cambios ____existe un formato para una propuesta de cambio. ____el proceso para los cambios a realizar en la línea base están establecidos en el plan SCM. ____en el plan existen procesos para aprobar los cambios.

Anexo 7 – Proceso de auditoría a la administración de la configuración

Evaluación de las instalaciones

SQA:_____ FECHA DE LA EVALUACIÓN:_____

Instalación evaluada (equipo, espacio):

Métodos o criterio usados en la evaluación:

Resultados de la evaluación:

Acciones correctivas recomendadas:

Acciones correctivas tomadas:

Anexo 8 – Encuesta realizada a los especialistas.

Usted ha sido seleccionado como especialista para ofrecer sus criterios valorativos acerca del contenido mostrado en la Estrategia de la Guía para facilitar el aseguramiento de la calidad en el proceso de desarrollo en la Facultad de informática de la UCF. De antemano le agradecemos por su cooperación.

Instrucciones: Para llenar este cuestionario de evaluación es importante que siga los siguientes pasos:

Evalúe los criterios que se resaltan utilizando para ello las variables que se adjuntan a cada uno.

Marque con una (X) en la escala de evaluación que se adjunta a cada variable utilizando la siguiente escala:

1___ Total desacuerdo 2 ___ NA 3___ 4___ Total acuerdo 5___.

Cuando el especialista no tiene elementos suficientes para emitir un criterio de valor sobre el ítem se considera NA. No aplica. Que sería en el valor 3.

Cuando lo considere pertinente escriba sus criterios en la celda correspondiente a las Observaciones.

Lenguaje y redacción.
REDACCIÓN 1___ 2___ 3___ 4___ 5___
a) La expresión de las ideas es clara y precisa.
b) Las estructuras gramaticales se utilizan correctamente.
Observaciones:
LENGUAJE ADECUADO 1___ 2___ 3___ 4___ 5___
a) El lenguaje utilizado requiere de un conocimiento elemental sobre el aseguramiento de la calidad.
b) El lenguaje es fácilmente comprensible.
Observaciones:
Contenido.
CAPACIDAD DE DESCRIPCIÓN, EXPLICACIÓN Y PREDICCIÓN 1___ 2___ 3___ 4___ 5___
a) Se observa coherencia de los objetivos con el contenido.
b) Favorece el desarrollo de habilidades en el desempeño.
c) Ética adecuada.
d) Evidente utilidad práctica.

e) La profundidad y generalidad del contenido se adecua a los desarrolladores, grupos SQA e Ingenieros de software.
f) Permite elevar los conocimientos de los usuarios e Ingeniero de software sobre las actividades de aseguramiento de la calidad.
Observaciones:
CONSISTENCIA LOGICA 1___ 2___ 3___ 4___ 5___
a) Adecuada interrelación y coherencia entre los aspectos tratados.
b) Las actividades están estructuradas con inicio desarrollo y cierre.
c) Correcta relación entre la teoría y la práctica.
d) Las ilustraciones son oportunas y se corresponden con el contenido.
Observaciones:
Aspectos pedagógicos.
MOTIVACIÓN 1___ 2___ 3___ 4___ 5___
a) Logra motivar por su originalidad.
b) La interactividad es apropiada para el usuario.
c) El contenido estimula su utilización.
Observaciones:
CONFIABILIDAD PSICOPEDAGÓGICA 1___ 2___ 3___ 4___ 5___
a) La estrategia es eficaz instructivamente.
b) La estrategia se adapta a las exigencias comunicativas del usuario.
Observaciones:
Perspectiva
EXPECTATIVAS 1___ 2___ 3___ 4___ 5___
a) La estrategia satisface las expectativas del usuario.
b) Constituye una fuente de consulta necesaria para los desarrolladores e Ingenieros de software.
Observaciones:

Anexo 9 – Scale: ALL VARIABLES.

Case Processing Summary			
		N	**%**
Cases	**Valid**	12	100.0
	Excluded(a)	0	.0
	Total	12	100.0
a Listwise deletion based on all variables in the procedure.			

Reliability Statistics	
Cronbach's Alpha	**N of Items**
.852	7

Anexo 10 –Resultados.

Statistics								
		Redacción	Lenguaje adecuado al nivel de enseñanza	Capacidad de descripción, explicación y predicción	Consistencia Lógica	Motivación	Confiabilidad Psicopedagógica	Expectativas
N	Valid	12	12	12	12	12	12	12
	Missing	0	0	0	0	0	0	0
Median		5.00	5.00	5.00	5.00	4.25	5.00	5.00
Range		1	1	1	1	2	1	1

Redacción					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	En acuerdo	3	25.0	25.0	25.0
	Total acuerdo	9	75.0	75.0	100.0
	Total	12	100.0	100.0	

Lenguaje adecuado al nivel de enseñanza					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	De acuerdo	5	41.7	41.7	41.7
	Total acuerdo	7	58.3	58.3	100.0
	Total	12	100.0	100.0	

Capacidad de descripción, explicación y predicción					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	De acuerdo	3	25.0	25.0	25.0
	Total acuerdo	9	75.0	75.0	100.0
	Total	12	100.0	100.0	

Consistencia Lógica					
		Frequency	Percent	Valid Percent	Cumulative Percent
valid	De acuerdo	4	33.3	33.3	33.3
	Total acuerdo	8	66.7	66.7	100.0
	Total	12	100.0	100.0	

Motivación					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NA	3	25.0	25.0	25.0
	De acuerdo	8	66.7	66.7	66.7
	Total acuerdo	1	8.3	8.3	100.0
	Total	12	100.0	100.0	

Confiabilidad Psicopedagógica					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	De acuerdo	8	66.7	66.7	66.7
	Total acuerdo	4	33.3	33.3	100.0
	Total	12	100.0	100.0	

Expectativas					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	De acuerdo	2	16.7	16.7	16.7
	Total acuerdo	10	83.3	83.3	100.0
	Total	12	100.0	100.0	

Anexo 11 –Prueba de Kendall.

Ranks	
	Mean Rank
Redacción	4.67
Lenguaje adecuado al nivel de enseñanza	4.04
Capacidad de descripción, explicación y predicción	4.63
Consistencia Lógica	4.33
Motivación	2.17
Confiabilidad Psicopedagógica	3.25
Expectativas	4.92

Test Statistics	
N	12
Kendall's W(a)	.567
Chi-Square	4.829
df	6
Asymp. Sig.	.001
a Kendall's Coefficient of Concordance	