La necesidad de los procesos de calidad en toda la vida de los diferentes ciclos de desarrollo de software

Manuel Alejandro Monroy Vargas & John Alexander González.
Universidad Nacional Abierta y a Distancia.
Escuela de ciencias administrativas, contables, económicas y de negocios
ECACEN.
Especialización en Gestión de Proyectos
Septiembre 2018.

Dedicatoria

Dedicado al esfuerzo y al apoyo incondicional de mi mamita *Lila Patricia Vargas Gómez* que desde el cielo me dio fuerzas para poder llevar a cabo esta Monografía y a que sin sus concejos y apoyo no hubiera podido llegar a ser lo que soy y sé que desde allá este triunfo es más tuyo que mío.

Manuel Alejandro Monroy Vargas

Esta monografía está dedicada a mi madre e hijo, quienes me han apoyado día a día, quienes me han enseñado el sentido de la unión y la motivación. Ellos han sido el pilar de mi vida y por quienes me he esforzado laboral y académicamente para salir adelante. A mi compañero de monografía, Manuel Monroy, quien con todos sus valiosos aportes y dedicación hizo posible la finalización de esta fase académica.

John Alexander González

Agradecimientos

Gracias a nuestra tutora Olivia Mendoza ya que a su constante entrega, dedicación y apoyo no hubiéramos podido conseguir este resultado tan esperado.

Introducción

El propósito de esta monografía será el de analizar la baja importancia a los procesos de calidad y QA, dentro de las empresas que realizan diferentes desarrollos de software. Lo anterior, pese a que en la actualidad existen demasiados procesos y técnicas en los diferentes ciclos de desarrollo de software para garantizar un mínimo estándar de calidad con el fin de evitar reproceso y perdidas de dinero en los desarrollos de software, se ha identificado que no se da la importancia necesaria, lo que conlleva a gastos muchos más altos por errores que se inyectan a los desarrollos en las diferentes etapas de los ciclos de vida.

Por tanto, lo que se busca con esta monografía ver las metodologías existentes para garantizar la calidad del desarrollo viendo sus pro y contras y así evaluar la importancia de los mismo en los diferentes proyectos de desarrollo de software y poder determinar en qué etapas es más necesario estos procesos de QA y Calidad.

Resumen

El costo de los problemas de software o errores generados la baja calidad en los desarrollos, sumados a los pobres procesos de calidad en todas las fases del ciclo de vida de los sistemas hacen parte de un problema global en las industrias que optan por desarrollar software para mejorar sus procesos internos, comerciales y de relación con los clientes.

La dificultad de mejorar y garantizar la calidad en el desarrollo de software es evidente, la falta de aplicación de procesos y metodologías de gestión incurren en que se deje de lado la calidad y se busque solamente la generación de un producto, donde un error puede generar altos costos a las empresas por conceptos de recuperación de plataformas e información, multas, y transacciones no exitosas.

El fundamento de necesidad de asegurar la calidad en el desarrollo de software se basa en la dificultad de ejecutar tareas de evaluación de software desde la visión del gerente de proyectos en las diferentes fases del ciclo de vida y del bajo presupuesto que se aplica especialmente a las actividades de testeo y pruebas de calidad exhaustivas.

Mediante la aplicación de métodos de testeo y de aseguramiento de la calidad que se mencionan en este documento, se podrán mejorar los resultados del desarrollo de software

El propósito de esta monografía será el de analizar la baja importancia a los procesos de calidad y QA, dentro de las empresas que realizan diferentes desarrollos de software. Lo anterior, pese a que en la actualidad existen demasiados procesos y técnicas en los diferentes ciclos de desarrollo de software para garantizar un mínimo estándar de calidad con el fin de evitar reproceso y perdidas de dinero en los desarrollos de software, se ha identificado que no se da la importancia necesaria, lo que conlleva a gastos muchos más altos por errores que se inyectan a los desarrollos en las diferentes etapas de los ciclos de vida del software.

Abstract

The cost of software problems or errors generated by the low quality of the developments, coupled with the poor quality processes in all phases of the systems life cycle are part of a global problem in the industries that choose to develop software for improve their internal processes, commercial and relationship with customers.

The difficulty of improving and guaranteeing quality in software development is evident, the lack of application of management processes and methodologies incurs that quality is left aside and only the generation of a product is sought, where an error can generate high costs to companies for concepts of recovery of platforms and information, fines, and unsuccessful transactions.

The foundation of the need to ensure quality in software development is based on the difficulty of executing software evaluation tasks from the project manager's vision in the different phases of the life cycle and the low budget that applies especially to the testing activities and exhaustive quality tests.

Through the application of testing and quality assurance methods mentioned in this document, the results of software development can be improved

The purpose of this monograph will be to analyze the low importance of quality processes and QA, within companies that perform different software developments. The foregoing, although there are currently too many processes and techniques in the different software development cycles to guarantee a minimum quality standard in order to avoid reprocessing and loss of money in software developments, it has been identified that the necessary importance is given, which leads to much higher expenses for errors that are injected into the developments in the different stages of the software's life cycles.

Tabla de Contenidos

Dedicatoria 3

Agradecimientos 4

Introducción 5

Resumen 6

Abstract 7

Tabla de Contenidos 8

Lista de tablas 10

Lista de figuras 11

Capítulo 1 12

- Planteamiento del problema 12
- Justificación 13

Capítulo 2 14

- Objetivos 14
  - Objetivo General 14
  - Objetivos Específicos 14

Capítulo 3 15

- Marco Referencial 15
  - Marco Teórico 15
  - Marco Conceptual 17
  - Acerca de las pruebas y calidad de Software 20
  - Principales métodos de pruebas de Software 21
  - El Aseguramiento de la calidad del software 26

Defectos de Software 29

Modelos de gestión de calidad del Software..... 31

Nuevos modelos de administración de Aseguramiento de software 33

El mejoramiento de la calidad de software en ambientes productivos reales..... 35

Capítulo 4..... 44

Opiniones Personales..... 44

Lista de referencias 46

Lista de tablas

Tabla 1 - Calculo de Factores de ponderación y factor de ajuste del valor 28

Tabla 2 - Etapas de evaluación del proceso 39

Lista de figuras

Ilustración 1 - Integración de procesos Aseguramiento Calidad 18

Ilustración 2 - Tendencia costo de errores 25

Ilustración 3- Tomada de: https://paulhammant.com/2012/11/01/testability-and-cost-of-change/ 25

Ilustración 4 - Clasificación de Errores de software 31

Ilustración 5- Modelo Rayleigh - Tomada de: https://flylib.com/books/en/1.428.1/basic_assumptions.html 32

Ilustración 6- Niveles de CMM 34

Ilustración 7 - Organigrama Empresa 35

Ilustración 8 - Diagrama Sistema BSM 36

Ilustración 9 - Funcionamiento Controlador 38

Capítulo 1

Planteamiento del problema

Dentro de nuestro corto trasegar de vida profesional hemos tomado un rumbo de acuerdo con las tendencias mundiales de crecimiento tecnológico, lo que nos ha llevado a encaminarnos al maravilloso mundo de TI, el cual está lleno de cambios y de innovaciones diarias. En este no solo hemos encontrado nuestra formación como profesionales sino una serie de inquietudes que día a día nos lleva a indagarnos si los procesos ya definidos como metodologías comprobadas, son lo suficientemente beneficiosas para los ciclos de vida de los desarrollos que estamos construyendo con nuestro apoyo a las diferentes empresas donde brindamos nuestros servicios.

Por esta razón vemos que los ciclos de vida de desarrollo en vez de ser cada día más rápidos y eficientes en algunas empresas son demasiado complejos y paquidérmicos lo cual se traduce en costos elevados para las mismas. Por consiguiente, hemos evaluado que gran parte de los problemas que se tienen son derivados a que no se aplican correctamente los procesos de calidad o en su defecto se aplican metodologías que no son acordes con el deber ser de sus empresas. Así mismo nos hemos podido cuestionar en ciertas metodologías de calidad que a nuestro punto de vista no son realmente beneficiosas por tanto queremos realizar las comparaciones para determinar si esto depende del tipo de empresa en donde se aplican las metodologías o necesariamente es la metodología la que no es enriquecedora para el ciclo de vida de software.

Justificación

El mundo actual depende del software, todos los negocios dependen del software, los dispositivos móviles lo usan, empresas grandes y pequeñas utilizan el software para soportar sus procesos internos y de negocios. Data esta realidad la calidad del software realmente importa. Teniendo en cuenta la importancia y el alcance global un software de baja calidad nos es aceptable.

La calidad de cualquier producto es la que ofrece un valor a dicho producto y el software no es la excepción. Los procesos de desarrollo de software requieren aplicación de procesos de calidad por varios actores entre ellos: equipos de desarrollo, clientes, patrocinadores, usuarios finales, entre otros.

La calidad del software depende de varios aspectos, sobre los cuales es necesario aplicar procesos definidos en las diferentes fases del cliente de vida del software. Entre los aspectos sobre los cuales se debe aplicar procesos específicos de calidad se encuentran: calidad en aspectos funcionales, calidad en aspectos estructurales, calidad en procesos, calidad en documentación, calidad en procesos de mejora y mantenimiento. Cada uno de ellos requiere un nivel detallado de análisis y documentación.

Mediante esta propuesta se tendrán unas bases sobre las necesidades de los procesos de calidad en toda la vida de los diferentes ciclos del desarrollo de software, sirviendo como base de consulta para aplicar de manera consolidada las metodologías de gestión de calidad en búsqueda de la obtención de productos de alto grado de calidad, que satisfaga las necesidades de los clientes, usuarios y patrocinadores de este tipo de proyectos.

Capítulo 2

Objetivos

Objetivo General

Demostrar la necesidad de los procesos de calidad en todas las etapas del ciclo de vida de los desarrollos de Software

Objetivos Específicos.

- Validar la eficiencia de las metodologías de pruebas en los ciclos de desarrollo
- Conocer la necesidad de calidad en los Desarrollos de Software
- Comprobar la reducción del ciclo de vida de desarrollo de software apalancados en los procesos de calidad.

Capítulo 3

Marco Referencial

Marco Teórico

Antes de comenzar definamos la palabra "calidad", ya que algunas veces esta no se toma en contexto formal y se da para confusiones especialmente en el marco del negocio de servicios relacionados con pruebas de software.

Para ello nos basamos en el artículo de Joel E. Ross columnista del Total Quality Management quien nos define el significado de Calidad amparado en la vida de desarrollo de cualquier software y donde podemos concluir que existen varias formas de ver el punto de vista en término de calidad de un producto y que se tiene que tener cuidado con las expectativas del Cliente y en cómo se utiliza el término en el diario vivir.

Es errado decir, que la calidad de un producto depende de las pruebas que se le hagan al mismo o que las pruebas "garantizan" porque el nivel de calidad de un producto depende de todos los procesos y personas involucrados en el desarrollo del producto, en el caso de nuestro tema central dependerá de los procesos y personas involucradas en el proceso de desarrollo de software.

Cómo se expuso la "calidad" depende del punto de vista que se defina y esta definición no es realizada por pruebas sino por las necesidades y expectativas del productor. (Ross, 2009)

Los tres aspectos de la calidad del software son la calidad funcional, la calidad estructural y la calidad del proceso. Las pruebas de software generalmente se enfocan en la calidad funcional. Todas las características del software se pueden probar, por lo que una gran parte de garantizar la calidad funcional se reduce a las pruebas.

El segundo aspecto de la calidad del software, la calidad estructural, significa que el código en sí está bien estructurado. A diferencia de la calidad funcional, la calidad estructural es difícil de probar. Entre los atributos de este tipo de calidad se tiene: calidad de código, calidad de mantenibilidad del código, calidad de entendimiento de código y documentación, calidad de seguridad del código.

El tercer aspecto, calidad del proceso, muy importante. La calidad del proceso de desarrollo afecta significativamente la percepción de los usuarios, los equipos de desarrollo y los patrocinadores, por lo que los tres grupos tienen interés en mejorar este aspecto de la calidad del software. Entre los atributos de este tipo de calidad se tiene: calidad en las entregas, calidad en el levantamiento de requerimientos, calidad en las reuniones de presupuesto, etc.

La conexión de estos tres aspectos nos lleva al éxito o al fracaso en proyectos de desarrollo de software, en el sentido que se haga una correcta conexión de estos tres aspectos se llega a un producto de calidad y a la satisfacción del usuario y del equipo de desarrollo.

Mediante la aplicación de procesos documentados, metodologías de desarrollo, utilización de herramientas de testeo, codificación y demás se puede llegar al objetivo requerido. Por esto la importancia de documentar y conocer los procesos de calidad en toda la vida de los diferentes ciclos del desarrollo de software. Desglosando cada uno de los aspectos de calidad, para verlos como un todo, identificado metodologías de desarrollo, enfocándose en la inclusión de requerimientos de los stakeholders, conociendo herramientas que ayuden a controlar y medir la calidad.

Teniendo en cuenta la importancia que tiene el software en el mundo actual no es descabellado exagerar en la aplicación de procesos y metodologías de aseguramiento de la calidad en el desarrollo de software que generen valor económico y al proceso de desarrollo como tal.

Marco Conceptual

Según la IEEE el software “es la suma total de los programas de ordenador, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo”.

En el pasado los cálculos a mano eran utilizados para controlar procesos sencillos y complejos, mantener el control de elementos críticos para las compañías y que en caso de falla podría traer delicadas consecuencias tanto físicas (problemas en plantas físicas, problemas ambientales, riegos de seguridad), como legales (demandas, millonarias multas, desfalcos, fraudes).

Por medio del presente documento nos ponemos nuestra atención en la verificación de la calidad en el ciclo de vida del desarrollo de software, relacionado la gestión de proyectos con los procesos de calidad, consultando publicaciones y tomando ejemplos de los costos acarreados por problemas de calidad en el desarrollo de software.

Para entender las necesidades de aplicar esfuerzos en la aplicación de procesos de calidad en el desarrollo de software podemos enfocarnos en varios tópicos relevantes:

- Necesidades de verificación del software
- Defectos en el desarrollo de software
- Reporte errores y acciones correctivas
- Integración de procesos de calidad en el desarrollo de software

Mediante la siguiente integración de procesos de manera general se puede contribuir en la integración del desarrollo de software con métodos de gestión de calidad, con el fin de asegurar y verificar el correcto desarrollo de Programas de computación y contribuyendo integralmente a la

disminución de errores y efectos negativos, que por causa de una mala calidad en el software se puedan presentar.

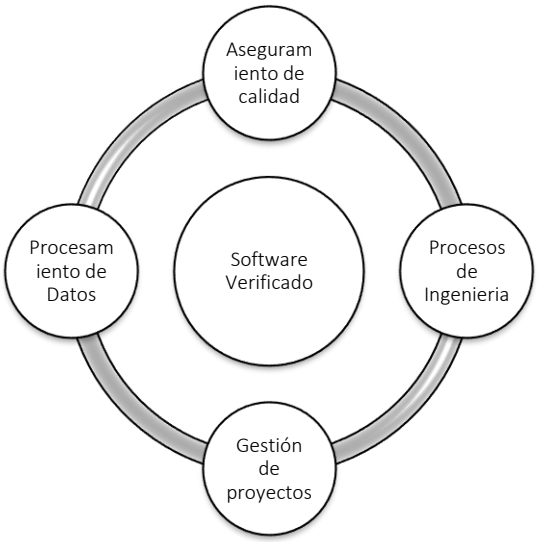

Ilustración 1 - Integración de procesos Aseguramiento Calidad

Es necesario plantearnos algunas preguntas con el fin de encontrar las verdaderas razones del porqué es necesario desarrollar software y con altos niveles de calidad, como:

- ¿La aplicación de métodos de gestión de proyectos para asegurar calidad de software incremente la seguridad y el valor de los activos de la compañía?
- ¿Los cálculos, procesos y resultados que genera el software son importante para la toma de decisiones de las compañías?
- ¿La calidad del software infiere directamente en la seguridad física, ambiental y capital tanto de los seres humanos que lo usan como de los activos de las compañías que lo implementan?

(Richter, 1982,) examina la productividad y la confiabilidad para el desempeño de los programas de computador con un gran nivel de detalle, donde aborda aspectos relacionados con la

fiabilidad, rendimiento y especialmente los pasos para garantizar el uso del software de calidad, no solamente para la seguridad sino para todas las aplicaciones.

Dentro de las revisiones que recomienda el autor se identifican las siguientes áreas críticas en el aseguramiento de la calidad del software:

- El software o desarrollos de dominio público son difícilmente asegurables
- El aseguramiento del software debe aplicar procesos de verificación y documentación.
- El aseguramiento de la calidad del software debe controlarse y comunicarse de la manera más adecuada posible, aplicando procesos de gestión de proyectos.

Métodos de investigación como los que plantea (Ferrance, 2000) han sido utilizados para el mejoramiento de los procesos de calidad en el desarrollo de software. (Ferrance, 2000) plantea que investigación para la "no es un proyecto de biblioteca donde aprendemos más sobre un tema que nos interesa. No es una solución de problemas en el sentido de intentar para descubrir lo que está mal, sino más bien una búsqueda de conocimiento sobre cómo mejorar".

El propósito de este tipo de investigación es aprender desde la experiencia y aplicar este aprendizaje para ejecutar cambios. "La tarea del investigador profesional es proporcionar liderazgo y dirección a otros participantes o partes interesadas en el proceso de investigación." (Stringer, 2007)

Basados en estos conceptos y en un listado de procesos que plantea (Brannick, 2005) acerca de los procesos de investigación para la acción, podemos listar una serie de pasos a seguir para la aplicación de esta metodología en la búsqueda del mejoramiento aplicado al desarrollo de software, así:

- Revisar las prácticas y procesos actuales
- Identificar aspectos a mejorar

- Generar un plan de acción
- Ejecutar plan de acción
- Evaluar resultados
- Replantear un ciclo adicional
- Continuar hasta terminar.

Este método de investigación puede aplicarse ampliamente a la práctica de la mejora en la calidad de software. Como desventaja se tiene que es un método más difícil de hacer que la investigación convencional ya que el investigador tiene que realizar dos funciones marcadas: realizar la investigación y realizar el control de cambios mediante el registro de los resultados.

Existen varios métodos de investigación-acción, donde tomando como referencia los que propone (Holter IM, 1993) se proponen tres tipos: enfoque colaborativo, enfoque de colaboración mutua y enfoque de mejora.

Teniendo en cuenta que el aseguramiento de calidad en el desarrollo de software requiere de mejoramiento de procesos, creación de un ambiente donde se apliquen las mejores prácticas para las pruebas se sugiere adoptar el enfoque colaborativo, el cual sugiere: probar una intervención particular basada en un marco teórico existente donde la naturaleza de la colaboración entre el investigador y el profesional es técnica y facilitadora. (Conference, 2004)

Acerca de las pruebas y calidad de Software

El propósito de las pruebas de software es detectar errores en las aplicaciones de software. El encargado de realizar las pruebas de software debe detectar los errores antes de la liberación del producto para uso final en el mayor porcentaje de usos posibles.

(Myers, 1983) Indica que el principio de las pruebas de software es: “es el proceso de ejecutar un programa con la intensión de buscar errores” este proceso se debe llevar a cabo por diferentes frentes: validar que el programa haga lo que tenga que hacer y ejecutar el programa para confirmar que no haga lo que no tiene que hacer.

Para que el probador pueda buscar esos errores, él debe planear un número de pruebas a ejecutar. Estas pruebas deben ser basadas en el conocimiento que se tenga del software, basados principalmente en la composición del software (estructura interna) y en segundo lugar en función del negocio (funcionalidad). Basado en estos dos factores, el probador debe escribir una serie de casos para pruebas, para tratar de detectar errores y de esta manera asegurar la función del software. (Myers, 1983) Sugiere que “Los datos de entrada no válidos e inesperados son más efectivos para detectar errores que los datos válidos y esperados”. El problema muchas veces para el probador es saber si los resultados de las pruebas son errores o realmente son los resultados esperados.

Principales métodos de pruebas de Software

Caja Negra – Prueba Funcional

Las pruebas funcionales o de caja negra son las pruebas que: " ignoran el mecanismo interno de un sistema o componente y se centran únicamente en los resultados generados en respuesta a las entradas seleccionadas y las condiciones de ejecución" (Gao, 2003) Las pruebas de caja negra se enfocan en la ejecución de casos de prueba sobre los requisitos funcionales del software para determinar si los resultados son aceptables.

Caja Blanca – Prueba Estructural

Hay dos beneficios de las pruebas estructurales; el primero es la creación de casos de prueba basados en la lógica de la aplicación. El segundo es la detección de qué tan exitosas son las pruebas al examinar cuántas rutas diferentes se ejecutaron a través de un programa.

"En las pruebas de ruta, una dificultad importante es que hay demasiadas rutas factibles en un programa. Las técnicas de prueba de ruta usan solo información estructural para derivar un subconjunto finito de esas rutas, y a menudo es muy difícil derivar un subconjunto efectivo de rutas " (Gao, 2003).

La complejidad de la lógica está determinada por la cantidad de nodos diferentes y el número de diferentes rutas posibles a través de la aplicación. El uso métricas permitiría al probador determinar qué parte del código se ha ejecutado. Los resultados del caso de prueba demuestran la probabilidad del éxito futuro de la aplicación.

Caja Gris

"Un enfoque más adecuado para los sistemas en tiempo real es el de las pruebas de hilos. Los sistemas están segmentados en hilos donde la prueba de software y la prueba estructural se entrelazan. Los módulos asociados con cada hilo se codifican y prueban en el orden que se define dentro del cronograma. La integración de las compilaciones eventualmente construirá todo el sistema" (Millo, 1987)

La viabilidad de las pruebas de hilos depende de un proceso de desarrollo secuencial. Las pruebas se llevan a cabo en cada compilación o subproceso sucesivos, cada subproceso si tiene éxito se integra luego en todo el sistema. El proceso de prueba se entrelaza con el proceso de desarrollo más de cerca que con otros enfoques y esto a su vez se enfoca a la gestión de proyectos en los procesos de mejora continua.

Teniendo en cuenta estos procesos de pruebas no podemos dejar de un lado los enfoques que el profesor de Harvard David Garvin, en su libro Managing Quality , resumió dichos enfoques para definir la calidad: trascendente, basado en productos, basado en el usuario, basado en la fabricación y basado en valores. Por tanto, analicemos cada uno de ellos:

1. Visión trascendental de la calidad: los que tienen una visión trascendental dirían: "No puedo definirlo, pero sé cuándo lo veo".

Los anunciantes son aficionados a promocionar productos en estos términos. "Donde comprar es un placer" (supermercado), "Nos encanta volar y se nota" (línea aérea), y "Significa ojos hermosos" (cosméticos) son un ejemplo.

2. Vista basada en productos: las definiciones basadas en productos son diferentes. La calidad se ve como características o atributos cuantificables y mensurables. Por ejemplo, se puede medir la durabilidad o confiabilidad (p. Ej., Tiempo medio entre falla, ajuste y acabado), y el ingeniero puede diseñar para ese punto de referencia. La calidad se determina objetivamente. Aunque este enfoque tiene muchos beneficios, también tiene sus limitaciones. Cuando la calidad se basa en el gusto o preferencia individual, el punto de referencia para la medición puede ser engañoso.

3. Vista basada en el usuario: las definiciones basadas en el usuario se basan en la idea de que la calidad es un asunto individual, y los productos que mejor satisfacen sus preferencias (es decir, la calidad percibida) son aquellos con la más alta calidad. Este es un enfoque racional, pero conduce a dos problemas. En primer lugar, las preferencias de los consumidores varían ampliamente, y es difícil agregar estas preferencias en productos con gran atractivo. Esto lleva a la elección entre una estrategia de nicho o un enfoque de agregación de mercado que trata de

identificar los atributos de los productos que satisfacen las necesidades del mayor número de consumidores.

4. Vista basada en la fabricación: las definiciones basadas en la fabricación se refieren principalmente a las prácticas de ingeniería y fabricación y utilizan la definición universal de "conformidad con los requisitos". Los requisitos o especificaciones son diseños establecidos y cualquier desviación implica una reducción de la calidad. El concepto se aplica tanto a los servicios como a los productos. La excelencia en la calidad no está necesariamente en el ojo del espectador sino en los estándares establecidos por la organización. (Javier J. Gutiérrez)

Es importante resaltar la idea principal de los autores en Javier J. Gutiérrez, María J. Escalona, Arturo H. Torres, Manuel Mejías y Jesús Torres en su trabajo “Hacia una propuesta de pruebas tempranas del sistema”, donde quisieron plasmar a través de un ejemplo práctico en la construcción de un editor de notas como es de importante realizar pruebas tempranas.

Para ello hemos analizado en un ciclo de vida normal de construcción de un sistema de software cualquiera, en donde se tienen las acostumbradas macro etapas:

- Requerimientos (definición de los detalles específicos de la construcción)
- Diseño (diseño de lo especificado de acuerdo a las necesidades)
- Desarrollo (Construcción de los diseños preestablecidos)
- Pruebas (Validación de los requerimientos dados en la necesidad)
- Producción (Resultado final de los solicitado)

En un proceso normal vemos que la pruebas están en 4 lugar dentro de las etapas definidas, lo que hace más costoso generar un reproceso por algún error encontrado una vez ya esté construido el gran parte el sistema. Lo que buscamos con el análisis de estos procesos es evitar los errores en etapas finales los que hacen mucho más costoso reparar cada error en una etapa mayor

como lo podemos apreciar en la gráfica donde la tendencia exponencial de reparación en términos de costos es mucho más económica en las etapas iniciales.

Ilustración 2 - Tendencia costo de errores

Desde el año de 1981 se ha venido identificando la necesidad de tener en cuenta los costos de los errores en las fases iniciales del proceso de desarrollo de software, como lo demuestra (Boehm, 1981) en su libro Software Engineering Economics:

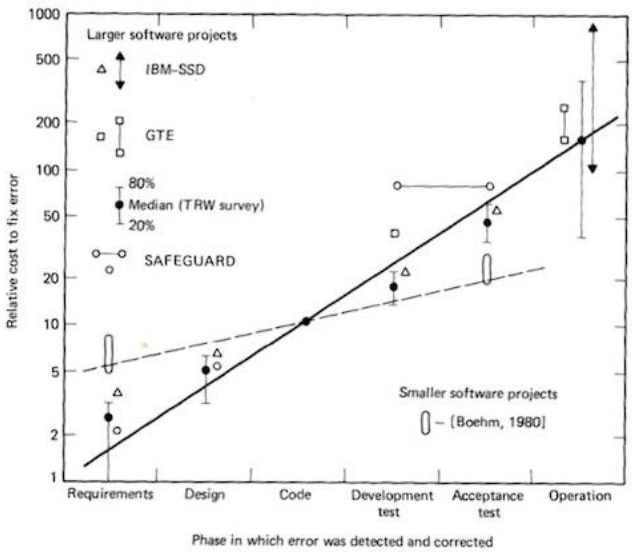

Ilustración 3- Tomada de: https://paulhammant.com/2012/11/01/testability-and-cost-of-change/

Por tanto, podemos evidenciar que podemos cambiar las perspectivas de las pruebas tradicionales una vez se realice la construcción del software, llevándolas a revisiones tempranas como en la etapa de Requerimientos por parte de los diferentes equipos de pruebas, en donde si se encuentra un error allí, lo único que retrasa es la entrega de un documento y no de un producto complemente ya construido.

Esto nunca va a garantizar la no percepción de errores, pero si garantizará la reducción de los mismos y sobre todo en las etapas finales del ciclo de vida lo que generará cosos excesivos y pérdida de tiempo.

Esto a nivel de todo el ciclo de pruebas lo que permite es una celeridad en el proceso lo que permitirá reducción del ciclo y así poder enfocar estos esfuerzos ahorrados en nuevos desarrollos que al final es lo que busca un área o un cliente final que es la razón por la cual existiría cualquier software.

El Aseguramiento de la calidad del software

Según la IEEE y la0 Universidad SEI Carnegie Mello, en su glosario de terminamos claves para CMM, el aseguramiento de calidad de software se describe como un conjunto de actividades diseñadas para evaluar el proceso por el cual funcionan, son desarrollados y mantenidos los productos de software. (Galin, 2004)

Para los propósitos d0e esta monografía y asociando los procesos del ciclo de vida del desarrollo de software, el aseguramiento de calidad de software – SQA – se considerará como un proceso para la medición de entregables y actividades durante cada etapa del ciclo de vida del desarrollo. El objetivo de SQA es cuantificar la calidad de los productos y las actividades que los originan y también guiar un esfuerzo de mejora de la calidad.

Por esta razón es ventajoso integrar los procesos de calidad en el proceso de desarrollo de software. Donde la SQA debería también tomar en consideración el mantenimiento de un producto, la solución técnica, presupuesto y alcance del producto (procesos de gestión de proyectos). Así mismo, teniendo en cuenta las fases de calidad de los proyectos: La garantía de calidad difiere del control de calidad. Donde el control de calidad es un conjunto de actividades diseñadas para evaluar la calidad de un producto manufacturado, mientras que la evaluación se lleva a cabo durante o después de la producción del producto. Sin embargo, el aseguramiento de la calidad **reduce el costo de garantizar la calidad** verificando las actividades realizadas durante el proceso de desarrollo y fabricación.

Medida de calidad del software

Basándonos en las publicaciones de Stephen H. Kan (Kan, 2003) La satisfacción con la calidad general del producto y sus dimensiones específicas es generalmente obtenida a través de varios métodos de encuestas a clientes. Por ejemplo, empresas de tecnología a nivel mundial como lo es IBM y HPe, dentro de los parámetros específicos de satisfacción del cliente usuario de software incluye:

- Parámetros de Capacidad, usabilidad, rendimiento, confiabilidad, instalabilidad, mantenibilidad, documentación, servicio y general (IBM)
- Funcionalidad, facilidad de uso, fiabilidad, rendimiento y servicio (HPe)

La calidad del software que se produce en cada proceso o modelo se describe en términos del número de defectos que se crean. Por lo general, la métrica más común para defectos es el número de defectos por mil líneas de código, o hay otra métrica ligeramente diferente para la tasa de defectos en términos de análisis de puntos de función (FPA) abreviado a (FP).

Tasa de defectos = Suma de defectos / KLOC

Tasa de defectos = Suma de defectos / FP

Una versión más reciente de FPA - Mark II se utiliza "para medir el tamaño funcional de cualquier aplicación de software que se puede describir en términos de transacciones lógicas, las cuales comprenden un componente de entrada, proceso y salida. Es un método para la cuantificación análisis y medición de aplicaciones de procesamiento de información. Cuantifica los requisitos de procesamiento de información especificados por el usuario para proporcionar una cifra que expresa un tamaño del producto de software resultante. Este tamaño es adecuado para fines de la medición y estimación del desempeño en relación con la actividad asociada con el producto de software" (Standar, 2004) El número de puntos de función se deriva multiplicando

el recuento de funciones (FC) por el factor de ajuste del valor (VAF). El FC se deriva de sumar el total del número de cada uno de los cinco factores de ponderación multiplicado por la cantidad de componentes. Por ejemplo:

Tabla 1 - Calculo de Factores de ponderación y factor de ajuste del valor

FP=FC*VAF	
$FC = \sum_{i=1..n} wi * xi$	$VAF = 0{,}65 + 0.01 * \sum ci$
Donde w es el factor de ponderación y x es la cantidad de componentes	c es el total de puntajes de características, y i = 1 a 14
Factores de ponderación: • Número de entradas internas • Número de entradas externas • Numero de archivos lógicos	El VAF es una evaluación sobre el impacto de 14 características generales del sistema en términos de su posible efecto en la aplicación. Es escalado en el rango de cero a cinco. Las 14 características generales son:

• Numero de interfaces externas • Número de consultas externas	• Comunicaciones de datos • Funciones distribuidas • Rendimiento • Configuraciones muy utilizadas • Tasa de transacción • Entrada de datos en línea • Eficiencia del usuario final • Actualización en línea • Procesamiento complejo • Reutilización • Facilidad de instalación • Facilidad operacional • Varios sitios • Facilitación del cambio

Defectos de Software

Es importante nombrar en esta monografía que con respecto al software, hay tres clasificaciones de defectos de software o errores, como se les conoce más comúnmente, como:

Error de Software

Se produce un error de software durante el desarrollo del software. Este error puede estar en la forma de un error gramatical, un error lógico en el que el resultado de una secuencia de las

ejecuciones no dará como resultado que se pretendía o una interpretación errónea del usuario en los requisitos reales. Un error puede o no ser detectado durante la codificación o prueba del programa antes de que se lance a un cliente.

Fallo de Software

Se produce un fallo de software como resultado de un error que permanece en el programa de ejecución. No obstante, no se detectan todas las fallas y el software puede continuar ejecutándose sin cualquier problema obvio. Hay casos en los que los fallos del software pasan desapercibidos por muchos años de existencia de un programa.

Falla de Software

Una falla de software es una falla que resulta en un problema detectable; por lo tanto, se refiere como un fracaso. Una falla causaría un mal funcionamiento de la aplicación de manera obvia, de tal manera que se requiere la atención del mantenimiento del sistema.

Los errores de software se pueden categorizar según las diferentes etapas en las que ocurrir en el ciclo de vida del desarrollo. Como incida (Galin, 2004) : "Los errores de software son la causa de un software de deficiente calidad, es importante investigar las causas de estos errores para evitarlos". Un error de software puede ser un:

- Error de código
- Error de procedimiento
- Error de documentación
- Error de datos de software

Teniendo en cuenta el objetivo de esta monografía, es importante conocer las causas de los errores de software en las diferentes etapas del ciclo de vida de desarrollo de software, así:

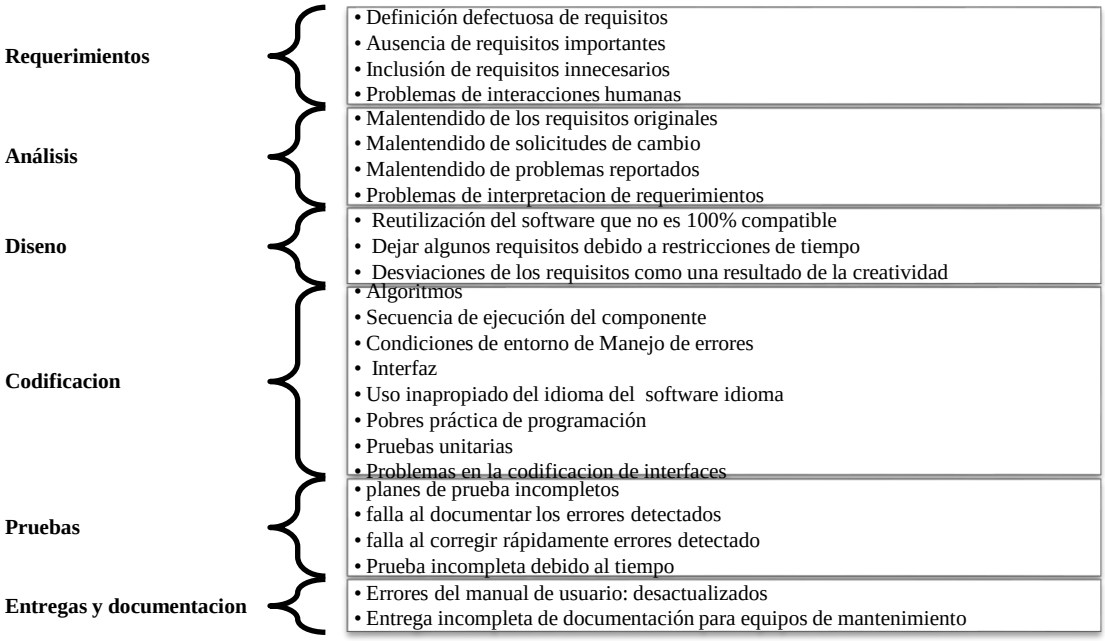

Ilustración 4 - Clasificación de Errores de software

Como se describe en el apartado asociado a la tendencia del costo de los errores según la fase del ciclo de vida del desarrollo de software (Ilustración 2 - Tendencia costo de errores), el número de defectos que entran en el proyecto aumenta con la continuación de las fases de desarrollo de software.

(Kan, 2003) describe que "La eficacia de eliminación de defectos de fase y las métricas relacionadas asociadas con los análisis de efectividad son útiles para la planificación de la calidad y la gestión de la calidad. Estas mediciones indican claramente en qué fase del proceso de desarrollo debemos centrarnos para mejorar ".

Modelos de gestión de calidad del Software

Es importante nombrar dentro de este documento los diferentes modelos de gestión de la calidad de software, donde los diferentes autores han aportado de manera importante a la mejora

de la calidad en los productos de software. Por tanto, dentro de lo modelos más significativos podemos encontrar:

Curva de Rayleigh para la calidad del desarrollo

El uso típico de los modelos de confiabilidad para la gestión de la calidad es predecir la fecha de finalización de las pruebas dado un nivel de detección de defectos. Si el nivel de defectos detectados es bajo, se requerirá un mayor esfuerzo de prueba. Los defectos para cada etapa del ciclo de vida de desarrollo se trazan y la curva resultante da una indicación de la fase de mayor inyección de defectos y eliminación de defectos.

"La relación entre los defectos formales de prueba de máquina y los defectos de campo, como se describe en el modelo (Raleigh) es congruente con el famoso principio contra intuitivo en pruebas de software de Myers (1979), que básicamente establece que cuantos más defectos se encontraron durante las pruebas formales más quedaron por encontrar más adelante. La razón es que, en la última etapa de las pruebas formales, la inyección de errores del proceso de desarrollo está básicamente determinada. Las altas tasas de defectos de prueba indican que la inyección de error es alta, si no se ejerce ningún esfuerzo adicional, más defectos escaparán al campo ". (Kan, 2003)

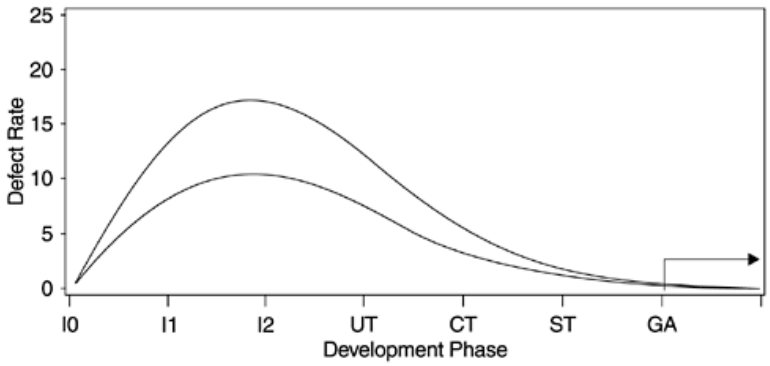

Ilustración 5- Modelo Rayleigh - Tomada de: https://flylib.com/books/en/1.428.1/basic_assumptions.html

Para finalizar esta publicación, debemos enfatizar los estudios que se han venido trabajando para la mejora de los procesos de aseguramiento de calidad de software (SQA), donde se ha venido mejorando los análisis y actividades de aseguramiento para las diferentes fases del ciclo de vida del desarrollo de software, dentro de los nuevos modelos de aseguramiento de software podemos nombrar:

TQM – Aseguramiento Total de la Calidad

Se deriva de un estilo japonés de gestión donde la garantía de calidad se implementó en todos los niveles de la empresa para mejorar la satisfacción del cliente. Los Principios son la gestión de la calidad del producto con la calidad del cliente a través de la mejora y el control de los procesos. Los elementos clave para TQM son:

- Un enfoque en el cliente para lograr la satisfacción total del cliente
- Mejora de procesos negocios y productos
- El elemento humano a la calidad, para generar una cultura de calidad de la empresa
- Medición y análisis de métricas de calidad para lograr el objetivo de mejorar la calidad
- Necesidad de liderazgo ejecutivo en la corporación

Paradigma de Mejora de la Calidad

Tiene como objetivo construir una organización basada en la mejora continua. sobre los objetivos en evolución y una evaluación de su estado en relación con estos objetivos. El enfoque utiliza evaluaciones y técnicas internas tales como Meta / Calidad / Metric GQM, construcción de modelos y análisis cualitativo / cualitativo para mejorar el producto a través del proceso.

Los pasos mas importantes para el paradigma de mejora de la calidad son:

- Caracterizar el entorno de auditorías del proyecto
- Establecer los objetivos
- Elegir el proceso apropiado
- Ejecutar el proceso
- Analizar los datos
- Documentar la experiencia para reutilizar

Modelo de Madurez de Capacidad del Proceso (CMM)

Fue desarrollado por el SEI en la Universidad Carnegie-Mellon. (Raynus, 1998) en su libro Mejora de Procesos de Software con CMM, nos da la siguiente definición: "CMM es un marco conceptual que representa la gestión de procesos del desarrollo de software. CMM contiene cinco niveles de madurez o etapas:

Ilustración 6- Niveles de CMM

Nivel 1: Las características para esta etapa incluyen costos caóticos e impredecibles, cronograma y calidad.

Nivel 2: Las características de esta etapa incluyen: Intuitivo - costo y calidad altamente variables, control razonable de horarios, métodos y procedimientos informales.

Nivel 3: Las Características de esta etapa incluyen: costos y horarios confiables mejorados, pero rendimiento de calidad impredecible.

Nivel 4: Las Características de esta etapa incluyen: control estadístico razonable sobre la calidad del producto, medición y análisis de procesos, administración de la calidad.

Nivel 5: Las Características de esta etapa incluyen: Base cualitativa para la inversión continua de capital en la automatización y mejora de procesos, prevención de defectos, innovación tecnológica, gestión de cambios.

El mejoramiento de la calidad de software en ambientes productivos reales

Muchas medianas y largas empresas han aplicado prácticas de mejoramiento de calidad en el desarrollo y testeo de software. Estas prácticas han sido expuestas a la opinión pública, donde se puede observar que realizan un manejo del TODO en la empresa, incluyen, no solo las áreas de producción de software, sino una proyección de todos los departamentos de la empresa y en especial de los departamentos de calidad de la empresa. La estructura estándar de una empresa está compuesta por los siguientes departamentos:

Ilustración 7 - Organigrama Empresa

Por ejemplo, una pequeña y mediana empresa que se encuentra en la industria de control de edificios, brindando soluciones de software y sistemas de administración de edificios (BMS). La empresa produce hardware, firmware y software que controla la calefacción, la ventilación y el aire acondicionad. La empresa controla el 70% del mercado de su país y entre el 2 y el 6% para del mercado en otros países de su continente. La compañía produce y vende soluciones de hardware de control de edificios y software BMS. La solución se compone de controladores electrónicos de entrada / salida y software propietario. El software se conecta con los controladores y también los programa. Los controladores pueden funcionar de forma independiente y controlar un edificio, pero inicialmente deben programarse utilizando el software basado en PC. Los controladores consisten en placas de circuitos impresos con interfaces de comunicaciones y sensores de entrada y salida electrónicos y dispositivos de control. Están alojados en plástico y funcionan con corriente alterna externa de 24 voltios. La función de los controladores se basa en la recopilación de datos del entorno del edificio y el cálculo de la salida para controlar el entorno. El software BMS está instalado y se ejecuta en un computador designado en el edificio para ser monitoreado. El software y los controladores suelen ser accesibles en la red Ethernet local, así:

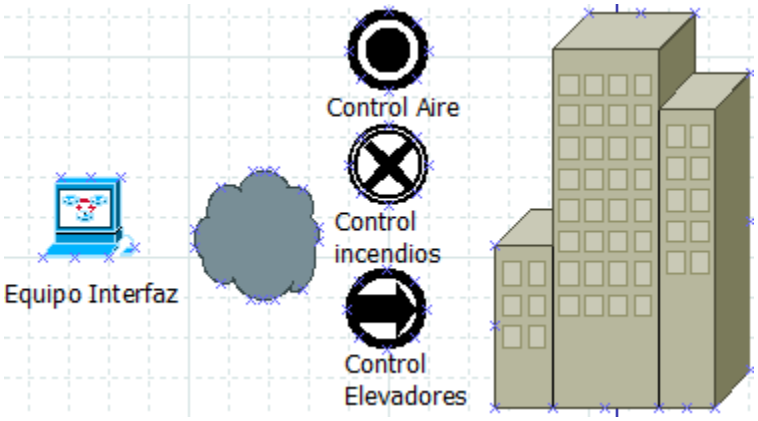

Ilustración 8 - Diagrama Sistema BSM

Hay tres niveles de en el sistema BMS: Un equipo PC interfaz, controladores de comunicaciones y controladores de campo. El PC BMS interactúa con un controlador de comunicaciones designado (elevadores, incendio, aire, etc). Un edificio puede contener muchos controladores de comunicaciones y una multitud de controladores de campo. Los controladores de comunicaciones son accesibles en la red Ethernet del edificio. Cada controlador de comunicaciones tiene su propia dirección IP y comparte la misma red. Una serie de controladores de campo están a su vez controlados por un controlador de comunicaciones. Los controladores de campo operan en su propia subred, cada uno con una dirección IP única.

Los controladores de campo son aquellos controladores que controlan directamente el entorno o sistema del edificio (por ejemplo, Iluminación, Calefacción). Los controladores de campo reciben datos en formatos analógico y digital. Los datos se reciben directamente de las señales de entrada de los sensores en el entorno del edificio o indirectamente de otro controlador de campo. El controlador calcula una operación de salida basada en estas entradas. La operación de salida está determinada por la programación del controlador, la estrategia es un programa que se ejecuta en el controlador mismo. Esta salida se usa para regular el entorno del edificio. La señal de salida se envía a los actuadores que operan la maquinaria de la planta que a su vez regulan el entorno del edificio, por ejemplo, la calefacción.

Cada controlador de campo debe ser programado directamente para cada entorno de construcción; esta programación se logra a través de una herramienta de ingeniería de software asistida por computadora (CASE). Esta programación se debe descargar a cada controlador a través de un computador y la red de comunicaciones del controlador. La programación del controlador de campo es ideada por un ingeniero especializado en la industria HVAC, el cual es fundamentalmente un programa de cálculos matemáticos que ejecuta el controlador. En el ejemplo

de un sistema de calefacción, los cálculos se basan en los sensores de entrada conectados al controlador (por ejemplo, sensores de calor) y actuadores de salida (circuitos de encendido para calderas de gas, bombas de agua, bobinas de ventiladores, etc). El objetivo es controlar el sistema de calefacción en función de la temperatura de una habitación y la necesidad de calentar o enfriar la habitación a través de un ventilador, caldera u otro dispositivo de calor, por ejemplo:

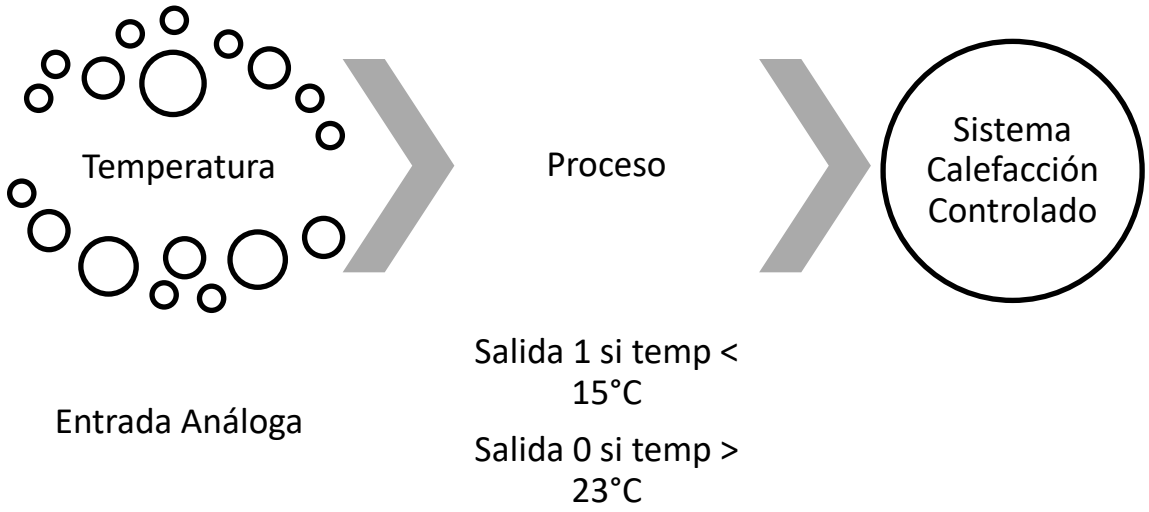

Ilustración 9 - Funcionamiento Controlador

Cuando la programación se ha completado y descargado a cada controlador de campo, los controladores se monitorean en sitio a través de un conjunto de aplicaciones en la red Ethernet del edificio. El hardware del controlador, el firmware y todo el software de soporte están diseñados y escritos en el departamento de Investigación y Desarrollo de la compañía. Es responsabilidad del departamento de control de calidad probar el firmware y todo el software antes de su lanzamiento a los clientes. El soporte técnico ofrece capacitación y soporte a los clientes. Los clientes están representados por dos sectores de la industria: Los usuarios finales son aquellos que supervisan

los edificios y las instalaciones de la planta. Los instaladores son aquellos clientes que compran a la empresa y actúan como intermediarios e instalan la solución del sistema para el usuario final.

Dentro de los problemas de calidad que se pueden presentar en la compañía está la principalmente preocupación con respecto a la calidad del software lanzado; la aplicación de herramientas de ingeniería y aplicaciones auxiliares. Otra preocupación es la tasa de progreso de los nuevos controladores en desarrollo.

La programación del software de control es un requisito fundamental para la operación de los controladores, ya que estos dependían de esta programación de control para su funcionamiento y éxito. Después del lanzamiento, los clientes informaron sobre un número alto de fallas del software.

La empresa tenía un sistema de calidad en funcionamiento donde se tenían estándares metodológicos basados en ISO 9000, además de cumplir con auditorias anuales de recertificación. Esta certificación se relaciona con los procedimientos de calidad de la empresa y su ejecución y no con la calidad de sus productos. Dentro del análisis realizado por el área de calidad junto con Investigación y desarrollo, se identificaron los problemas del cliente y se trató de identificar la causa desde el relanzamiento, pasando por el sistema de calidad hasta el inicio del proyecto.

La evaluación se realizó siguiendo tres etapas:

Tabla 2 - Etapas de evaluación del proceso

Etapa	**Actividades**
Planificación	• Identificar qué departamentos / equipos se evaluarán. • Evaluar a los participantes de cada departamento. • Solicitar comentarios de los clientes • Preparar la revisión de la documentación del proceso del departamento. • Preparar la revisión de la documentación del proyecto del departamento.

	• Preparar una entrevista con cada gerente de departamento.
Evaluación	• Generar cronograma para las evaluaciones para cada departamento. • Evaluar cada departamento • Revisar la documentación del proceso. • Revisar la documentación del proyecto. • Realizar la entrevista de los miembros del equipo y los gerentes. • Documentar los hallazgos de cada evaluación del equipo.
Reporte	• Compilar y presentar los hallazgos de la evaluación. • Actuar sobre los hallazgos y planificar mejoras en el proceso de calidad.

Los hallazgos de la evaluación inicial se agrupan en cinco áreas distintas:

- Las estadísticas generales de defectos para el proyecto de todos los departamentos.

  Para este ítem de evaluación se pueden generar los siguientes datos:

  - Horas / días de esfuerzo en el desarrollo
  - Horas / días de esfuerzo en las pruebas
  - Fecha de entrega planificada vrs fecha de entrega real
  - Percepción por parte de los clientes
  - Puntos de función vrs Líneas de código
  - Cantidad de defectos por punto de función.
  - Matriz de análisis del punto de función
  - Desglose de defectos.

- Un informe de calidad de atención al cliente basado en comentarios del cliente sobre el proyecto lanzado:

  - Encuesta de satisfacción del cliente
  - Reporte de fallas por parte del cliente
  - Encuesta de continuidad del cliente

- Diseño de caso de prueba y planificación de prueba para el proyecto de aplicación de ingeniería:
    - Verificación de los casos de prueba.
    - Evaluación de casos de prueba para determinar si las pruebas hubieran sido adecuadas para detectar los defectos en esas áreas.
    - Verificar si hay pruebas de límite
    - Verificar si hay pruebas de exploración de funcionalidad.
    - Verificar si hay pruebas de regresión de los defectos que se detectaron durante el curso de la prueba para asegurar que todos los defectos fueran corregidos.
    - Verificar las pruebas de la funcionalidad de la programación
    - Verificar cubrimiento de pruebas de caja blanca y caja gris.
    - Verificar pruebas de rendimiento o de usabilidad.
- Una auditoria interna de los departamentos de software, prueba, soporte y firmware en términos del proyecto
    - Auditar internamente equipos implicados (ejemplo Investigación y Desarrollo)
    - Auditar equipos de software, firmware, pruebas y atención al cliente
    - Auditar en temas como: documentación del proyecto, efectividad procedimientos de calidad, proceso de mejoramiento continuo.
- Una evaluación del proceso del ciclo de vida del desarrollo y del sistema de calidad en general:
    - Evaluación desde la perspectiva de desarrollo

- Evaluar que en el ciclo de vida de desarrollo que se lleve a cabo, se incluyan pruebas necesarias y también procesos de la atención al cliente.
- Verificar que el de prueba tenga el suficiente tiempo o recursos para las pruebas o pruebas del sistema que sean representativas de las expectativas del cliente respecto de un producto de calidad.
- Verificar que exista una revisión o firma de requisitos o documentos de diseño por departamentos que no sean los departamentos de software y firmware

Teniendo en cuenta esto, y como una aplicación práctica de esta monografía, se puede evidenciar la importancia de la necesidad de los procesos de calidad en toda la vida de los diferentes ciclos de desarrollo de software, donde se evidencia que en cada etapa del desarrollo es necesario aplicar acciones de mejora y de calidad, invertir tiempo y recursos en pruebas detalladas.

El método de evaluación de la calidad en las diferentes fases del ciclo de vida, debe incluir la revisión del trabajo por parte de miembros del equipo y sus pares, antes de su revisión por parte de expertos externos. Esta evaluación incluye la verificación de la documentación de diseño y la validación de las compilaciones de software, la calificación para la versión de las compilaciones, etc. El registro de métricas en cada etapa da un valor a la calidad de los resultados del proyecto.

Tener verificación y validación independientes en cada etapa del proyecto aumenta la tasa de detección de defectos y reduce la tasa de inyección de defectos. La integración de un equipo de calidad agrega un énfasis a la evaluación de la calidad en cada dominio. Un equipo de pruebas independiente que cuente con suficiente conocimiento en diseño de aplicaciones y conocimiento comercial puede planificar pruebas detalladas. Este conocimiento se puede utilizar en el diseño de casos de prueba estructurados, con una metodología detallada y que sean reutilizables para la

detección de defectos en cada etapa del ciclo de vida del software. Adicional, se requiere una estructura de informes que permita al equipo escalar problemas fuera del equipo de desarrollo del proyecto y con esto aplicar el mejoramiento continuo en las diferentes fases del proyecto y en los diferentes departamentos que puedan aportar a este proceso de mejoramiento continuo.

Capítulo 4

Opiniones Personales

Analizando este comportamiento dentro de nuestra la vida profesional en el sector del desarrollo de software, y teniendo en cuenta diferentes metodologías de construcción de software y donde Scrum y Cascada son las más representativas, hemos querido concluir que pese a que las metodologías agiles han mejorado significativamente los tiempos de desarrollo del software y que la metodología cascada no es tan eficiente en cuanto a tiempo para las soluciones, sí se tienen un proceso de pruebas muy bien estructurado que evalué todas las etapas del cualquier ciclo de desarrollo, es posible que la reducción de tempos totales sea significativa y lo que realmente es importante el nivel de calidad TOTAL generará reducciones no solo de tiempo sino de esfuerzos, pudiendo así emplear estos esfuerzos en otros desarrollos que realmente ayuden a los negocios o en ajustar los existentes y así podar generar ese valor agregado a la construcción de software.

Las compañías de desarrollo de software han puesto especial atención a los procesos de calidad del producto, adoptando con esto prácticas que se llevan a cabo en la gestión de proyectos y aplicándolas a los procesos del ciclo de vida del desarrollo de software, pensando en la calidad como un todo y teniendo especial consideración en la demanda del producto, el tiempo de desarrollo y los recursos requeridos para asegurar un producto de verdadera calidad. Por tanto, el testeo de software ya no es suficiente para el aseguramiento de calidad, sino que ahora es necesario incorporar los procesos de calidad en todas las fases del ciclo de vida del software y en cada entregable que se genere, donde se necesita de todos los miembros del equipo, además de los evaluadores, para abordar todos los factores de calidad.

Las pruebas de software y control de calidad de cada entregable de software requieren de una estructura y deben ser parte de un equipo de proyecto. El proceso de control de calidad debe

incorporarse al ciclo de vida del proyecto con la posibilidad de realizar mejoras al final del proyecto para retroalimentarlo en el próximo proyecto, esta continuidad del refinamiento del proceso ayuda con mejoras de calidad.

Personalmente opino que el proceso de aseguramiento de calidad debe consistir en un proceso combinado de desarrollo y prueba, el cual es más beneficioso para mejorar la calidad de cada fase del proyecto, donde con la experiencia del equipo de calidad y aplicando técnica de gestión de proyectos, se puede lograr fortalecer a todo el equipo del proyecto como un todo en la mentalidad de Garantía de calidad. Adicional, se requieren métricas de calidad de software para identificar los defectos y las mejoras de calidad en cada etapa del ciclo de vida del proyecto. Estas métricas se pueden usar para identificar tendencias a lo largo del ciclo de vida para ayudar con la gestión de la ejecución de la prueba y la iniciativa de calidad.

Lista de referencias

Boehm, B. W. (1981). *Software Engineering Economics 1st Edition.*

Brannick, T. (2005). *Doing Action Research In Your Own Organization.* SAGE Publications.

Conference, I. R. (2004). Innovations Through Information Technology.

Ferrance, E. (2000). *Action Research.* Northeast and Islands Regional Educational.

Galin, D. (2004). *Software Quality Assurance From Theory to Implementation.*

Gao, J. (2003). *Testing and Quality Assurance for Component-based Software.*

Holter IM, S.-B. D. (1993). *Action research: what is it?*

Javier J. Gutiérrez, M. J. (s.f.). *Hacia una propuesta de pruebas tempranas del sistema.* Obtenido de researchgate: https://www.researchgate.net/publication/254438551_HACIA_UNA_PROPUESTA_DE_PRUEBAS_TEMPRANAS_DEL_SISTEMA

Kan, S. H. (2003). *Metrics and Models in Software Quality Engineering* .

Millo, D. (1987). *Software Testing and Evaluation.* Cumming Publishing Co.

Myers, G. (1983). *El arte de probar el software.*

Raynus, J. (1998). *Software Process Improvement With CMM.*

Richter. (1982,). *H.P. A Comparison of Software Development Methodologies, Computers in Engineering* .

Ross, J. E. (27 de 08 de 2009). *totalqualitymanagement.wordpress.com.* Obtenido de http://totalqualitymanagement.wordpress.com/2009/08/27/definition-of-quality

Standar, A. Z. (2004). *Software engineering—Mk II Function Point Analysis—Counting Practices Manual* .

Stringer, E. T. (2007). *Action Research (Third Edition).* Londres: London: Sage Publications.