UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
DIRECCIÓN GENERAL DE LOS ESTUDIOS DE POSTGRADO
ÁREA DE CIENCIAS ADMINISTRATIVAS Y GESTIÓN
Postgrado en Gerencia de Proyectos

DISEÑO DE UN MODELO PARA EL CONTROL DE LA CALIDAD DE LOS PROYECTOS DE DESARROLLO DE SOFTWARE EN LAS ORGANIZACIONES DEL ESTADO VENEZOLANO.

Trabajo Especial de Grado presentado por

MONTAÑEZ GRATEROL LUIS EDUARDO

para optar al título de

Especialista en Gerencia de Proyectos

Asesor
López, Emmanuel

Caracas, Octubre 2006

INDICE

INDICE ii
Lista de Gráficos v
Lista de Tablas vi
RESUMEN vii
El Problema 1
1.1 Planteamiento del problema 1
1.2 Objetivo general 7
1.3 Objetivos específicos 7
Justificación 8
Marco Teórico 9
2.1 Bases Legales 11
2.2 Bases Teóricas 11
2.1 Calidad 11
2.2 Calidad del software 13
2.2.1 La calidad del software bajo distintos puntos de vista 15
2.2.2 Factores que determinan la calidad de software 16
2.2.3 La calidad a través de la normalización en la ingeniería del software y su problemática 19
2.3 Desarrollo de software y la gerencia de proyectos 22
2.4 Los modelos de calidad del software de gestión 23
2.5 Alcance de la función del aseguramiento de la calidad del software (SQA) 24
2.5.1 Control de Calidad 26
2.5.2 Costo de la calidad 26
2.6 Modelo de calidad IDEAL 30
2.6.1 Fase de Inicio 31
2.6.1.1 Estimulo para el cambio 32
2.6.1.2 Establecer el contexto 32
2.6.1.3 Buscar Patrocinio 33
2.6.1.4 Establecer infraestructura 33
2.6.2 Fase de Diagnóstico 33
2.6.2.1 Establecer el estado actual y el deseado 34
2.6.2.2 Elaborar recomendaciones 34
2.6.3 Fase de Establecimiento 34
2.6.3.1 Establecer prioridades 35
2.6.3.2 Desarrollar propuesta 35
2.6.3.3 Elaborar un plan de acción 35
2.6.4 Fase de Actuación 35
2.6.4.1 Crear solución 36
2.6.4.2 Probar solución 36
2.6.4.3 Refinar solución 36
2.6.4.4 Implementar solución 36

2.6.5 Fase de Aprendizaje 37
2.6.5.1 Analizar y validar 37
2.6.5.2 Proponer futuras acciones 37
Marco referencial 39
Marco metodológico 41
3.1 Tipo de investigación 41
3.2 Población, muestra y unidad de análisis 42
3.3 Variables: definición conceptual y operacional 42
3.3.1 Definiciones conceptuales 44
La Propuesta 45
4.1 Objetivo 45
4.2 Justificación 45
4.3 Estructura 46
4.3.1 Fase de análisis 46
4.3.1.1 Subfase de Visualización 46
4.3.1.1.1 Objetivos 46
4.3.1.1.2 Entradas 46
4.3.1.1.3 Procesos 47
4.3.1.1.3.1 Verificar estimación del proyecto 47
4.3.1.1.3.2 Elaborar matriz de riesgo 48
4.3.1.1.3.3 Realizar un análisis de factores a verificar 48
4.3.1.1.4 Salidas 49
4.3.1.1.5 Lista de verificación 50
4.3.1.1.6 Métricas 50
4.3.1.1.7 Matriz de comparación de resultados 51
4.3.1.2 Subfase de Conceptualización 51
4.3.1.2.1 Objetivos 51
4.3.1.2.2 Entradas 51
4.3.1.2.3 Procesos 52
4.3.1.2.3.1 Analizar la especificación funcional 52
4.3.1.2.3.2 Conducir una inspección formal 53
4.3.1.2.4 Salidas 54
4.3.1.2.5 Lista de Verificación 55
4.3.1.2.6 Métricas 55
4.3.1.2.7 Matriz de comparación de resultados para la métrica 55
4.3.2 Fase de Diseño 56
4.3.2.1 Objetivos 56
4.3.2.2 Entradas 56
4.3.2.3 Procesos 57
4.3.2.3.1 Análisis de Factores 57
4.3.2.3.2 Conducción de la Inspección de los Entregables de Diseño 58
4.3.2.4 Salidas 58
4.3.2.5 Lista de Verificación 59
4.3.2.6 Métrica 60
4.3.2.7 Matriz de comparación de resultados para la métrica 60
4.3.3 Fase de Desarrollo 61
4.3.3.1 Objetivos 61

4.3.3.2 Entradas 61
4.3.3.3 Procesos 62
4.3.3.3.1 Codificación 63
4.3.3.3.2 Depuración 63
4.3.3.3.3 Análisis de factores de codificación 64
4.3.3.3.4 Revisión de pares 66
4.3.3.3.5 Ejecutar pruebas 66
4.3.3.3.6 Verificar el registro de resultados de la prueba 68
4.3.3.4 Salidas 68
4.3.3.5 Lista de Verificación 68
4.3.3.6 Métricas 69
4.3.3.7 Matriz de comparación de resultados para la métrica 69
4.3.4 Fase de Cierre 70
4.3.4.1 Objetivos 70
4.3.4.2 Entradas 71
4.3.4.3 Procesos 71
4.3.4.3.1Verificar la instalación del nuevo software 72
4.3.4.3.2 Verificar nueva funcionalidad o cambio del software 72
4.3.4.3.3 Seguimiento 73
4.3.4.3.4 Documentar problemas 73
4.3.4.4 Salidas 73
4.3.4.5 Lista de Verificación 73
4.3.4.6 Métricas 74
4.3.4.7 Matriz de comparación de resultados para la métrica 74
4.4 Factibilidad 75
Análisis de los Datos 75
Recomendaciones y Conclusiones 78
5.1 Recomendaciones 78
5.2 Conclusiones 78
Bibliografía 80

Lista de Gráficos

Gráfico 1Punto de vista de la calidad del software según diversos autores 16
Gráfico 2 Costo relativo de corregir un error 28
Gráfico 3 Modelo I.D.E.A.L..... 31
Gráfico 4 Proceso del módulo de análisis 47
Gráfico 5 Proceso de la sub fase de conceptualización 52
Gráfico 6 Procesos del módulo de diseño 57
Gráfico 7 Procesos de la fase de desarrollo..... 62
Gráfico 8 Proceso de módulo de cierre 71

Lista de Tablas

Tabla 1 Operacionalización de las variables manejadas en el proyecto 43
Tabla 2 Lista de Verificación de la subfase visualización..... 50
Tabla 3 Matriz de interpretación de resultados para las métricas para la subfase de Visualización 51
Tabla 4 Lista de Verificación de la subfase de Conceptualización 55
Tabla 5 Matriz de interpretación de resultados para las métricas para la subfase de Conceptualización 55
Tabla 6 Lista de verificación de la fase de diseño..... 59
Tabla 7 Matriz de interpretación de resultados para las métricas para la fase de Diseño 60
Tabla 8 Lista de verificación de la fase de desarrollo 68
Tabla 9 Matriz de interpretación de resultados para las métricas para la fase de Desarrollo 69
Tabla 10 Lista de verificación de la fase de cierre 73

UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
DIRECCIÓN GENERAL DE LOS ESTUDIOS DE POSTGRADO
ÁREA DE CIENCIAS ADMINISTRATIVAS Y DE GESTIÓN
ESPECIALIZACIÓN EN GERENCIA DE PROYECTOS

Diseño de un modelo para el control de la calidad de los proyectos de desarrollo de software en las organizaciones del Estado Venezolano.

Autor: Luis E. Montañez G.
Tutor: Emmanuel López
Fecha: Octubre, 2006

RESUMEN

El software sin lugar a dudas es uno de los principales activos de toda organización, pues permite controlar y apoyar las actividades medulares para una mejor gestión. Para nadie es un secreto que el control de la calidad en el momento del desarrollo de estos productos, es bastante complicado, es por ello que las distintas organizaciones deben tener una metodología que permita la gestión de la misma y evitar la improvisación. Todo esto para poder cumplir con los objetivos de tiempo alcance costo y calidad. Para facilitar esta labor a los distintos organismos gubernamentales, se propone desarrollar una metodología que permita controlar la calidad de los proyectos de desarrollo de software en dichas organizaciones, partiendo de la base de distintas metodologías existentes.

En el desarrollo de esta investigación se pudo constatar que en el país existen pocas investigaciones orientadas al control de calidad en el de desarrollo de sistemas de información automatizados para el gobierno. O lo que es lo mismo, los entes gubernamentales carecen de una metodología de control de calidad a la hora de encarar un proyecto de desarrollo de software lo que permite afirmar que existen muy pocos casos de éxito de desarrollo de este tipo de proyectos en el gobierno.

Lo que se logro con este proyecto fue el desarrollo de una metodología bastante sencilla, que ayuda al control de la calidad de un proyecto de desarrollo de software en cada una de las fases de las mismas, indicando los productos de cada fase así como las métricas y una lista de verificación que permite medir el éxito o no de cada etapa, es decir, si en cada fase del desarrollo se lograron los objetivos.

Palabras claves: metodología, software, organización, calidad, proyectos

El Problema

Henry Wodsworth Longfellow, considerados unos de
los poetas americanos más famosos del siglo 19, afirmaba que
"lleva menos tiempo hacer una cosa bien que explicar por qué se hizo mal".

1.1 Planteamiento del problema

La era de la información es un período histórico caracterizado por una revolución tecnológica centrada en las tecnologías digitales de información y comunicación, concomitante, pero no causante, con la emergencia de una estructura social en red, en todos los ámbitos de la actividad humana, y con la interdependencia global de dicha actividad. Es un proceso de transformación multidimensional que es a la vez incluyente y excluyente en función de los valores e intereses dominantes en cada proceso, en cada país y en cada organización social (Castell, 2005).

En el siglo XXI es imposible ignorar la influencia e importancia de las tecnologías de información en todos los órdenes de la vida. El dilema no se encuentra en decidir si las aceptamos o la resistimos, sino en cómo las incorporamos de manera que se minimicen los riesgos inherentes a su implementación y uso y se maximicen sus beneficios en función de un objetivo previamente identificado (PNUD Venezuela, 2003).

Esta influencia e importancia de la tecnología de la información y la comunicación ha traído múltiples beneficios de los cuales ahora disfrutamos. Entre ellos se encuentran las mejoras en las comunicaciones, la globalización de los conocimientos, la materialización de los trabajos a distancia, la apertura de nuevos mercados, entre otros.

Gracias a la mejora de las comunicaciones entran a jugar nuevos conceptos que ahora no son nada novedosos, como es el caso de la globalización. Existen autores que la definen como

> "el proceso por el que la creciente comunicación e interdependencia entre los distintos países del mundo unifica mercados, sociedades y

culturas, a través de una serie de transformaciones sociales, económicas y políticas que les dan un carácter global. Así, los modos de producción y de movimientos de capital se configuran a escala planetaria, mientras los gobiernos van perdiendo atribuciones ante lo que se ha denominado la sociedad en red" (wikipedia, 2005).

Bajo el marco de la globalización, para que una organización tenga éxito, debe sin duda alguna, poder manejar y procesar la gran cantidad de datos que se generan a diario a fin de transformarla en información de manera eficiente y efectiva lo más rápido posible. El hombre ha creado varias herramientas que ayudan a esta labor, y una de ellas es, sin duda alguna, los sistemas automatizados de información, ya que colaboran en el procesamiento de datos y por consiguiente en la generación de la información, y esto a su vez, facilita la toma de decisiones.

A medida que la era de la información se ha desarrollado, en parte debido a la globalización, la complejidad en los procesos de automatización también ha ganado terreno, ya que la cantidad de información y los procesos de análisis de la misma han aumentado.

En este sentido, el aumento en la exigencia de una mejora de la calidad de los programas informáticos, se ha incrementado significativamente en los últimos años y, por esta razón, ha aumentado la complejidad en el desarrollo de los sistemas automatizados de información, por lo que resulta difícil, en la mayoría de los casos generar productos que cumplan cabalmente con los requerimientos del cliente.

Para responder a esta situación, han surgido una serie de herramientas, técnicas y modelos que facilitan a las organizaciones, encargadas del desarrollo de las tecnologías de la información, generar productos que satisfagan las expectativas del cliente e incluso las rebasen, herramientas que prometen ser "la solución a los problemas de calidad, costo y tiempos de desarrollo"; de éstas se puede mencionar la norma ISO 9000:2000, la ISO/IEC TR 15504, el modelo CMM[1] (Capability Maturity Model del Software Engineering Institute SEI), y el SQUID (Software Quality in Development) entre otros.

[1] Capability Maturity Model del Software Engineerig Institute SEI. El CMM es un modelo que propone una guía para que los desarrolladores puedan obtener control del proceso de desarrollo y mantenimiento del software.

Se puede encontrar mucha literatura donde se aprecian esfuerzos para especificar explícitamente los requisitos de calidad, como por ejemplo Jorgen Boegh en su libro "**A Method for Software Quality Planning, Control, and Evaluation**", la norma ISO/IEC 14598-3 aprobada en 1998, o la norma ISO/IEC 9126-1 creada en el 2001, entre otras. La importancia y aplicabilidad de estos esfuerzos son cada día más reconocidas en cada fase del ciclo de vida del software[2].

Todos estos esfuerzos y herramientas se desarrollan, entre otros motivos, debido a que la calidad del producto de software se diferencia de la calidad de otros productos de fabricación industrial, ya que los programas informáticos tienen sus propias características específicas. Ángel Cervera Paz (2002) establece, entre otras, las siguientes propiedades:

- El software es un producto mental.
- Se desarrolla, no se fabrica.
- Los costos del desarrollo de software se concentran en las tareas de ingeniería (creación).
- El software no se deteriora con el tiempo.
- Es artesanal en gran medida.
- El mantenimiento del software es mucho más complejo que el mantenimiento del hardware.
- Como disciplina, el desarrollo de software es aún muy joven.
- El software con errores no se rechaza.

Los organismos públicos no escapan a esta realidad, viéndose en la necesidad, muchas veces, de cambiar su estructura organizacional para adaptarse al dinamismo que la era les impone, todo esto para lograr una mejora en la atención al público al cual se deben y lograr una mejora en su gestión.

Venezuela no es un país aislado, por lo tanto también se ha visto obligado en los últimos años a trabajar en función de mejorar y adaptar a esta nueva era su gestión gubernamental, y una de las herramientas en las que se ha apoyado para lograr este objetivo son los sistemas automatizados de información. Existen, en la actualidad, varios proyectos que apuntan hacia la automatización de los procesos, o de la mayoría de ellos por parte de los distintos entes gubernamentales; un caso emblemático de éxito en la

[2] El ciclo de vida del software es el proceso que se sigue para diseñar, desarrollar, entregar y hacer evolucionar el software, desde la concepción de una idea hasta la entrega y el retiro del sistema.

administración pública de un sistema automatizado de información es SIGECOF[3] que actualmente es utilizado por la mayoría de los organismos públicos. Incluso existen leyes que obligan a los órganos públicos a hacer uso de la tecnología de la información para mejorar los procesos y hacerlos lo más confiable y transparentes posible. Este es el caso de la Ley Orgánica de la Administración Pública, vigente desde el 17 de Octubre del 2001, donde en su artículo 12 cita:

> "…A fin de dar cumplimiento a los principios establecidos en esta Ley, los órganos y entes de la Administración Pública deberán utilizar las nuevas tecnologías que desarrolle la ciencia, tales como los medios electrónicos, informáticos y telemáticos, para su organización, funcionamiento y relación con las personas. En tal sentido, cada órgano y ente de la Administración Pública deberá establecer y mantener una página en la Internet, que contendrá, entre otra información que se considere relevante, los datos correspondientes a su misión, organización, procedimientos, normativa que lo regula, servicios que presta, documentos de interés para las personas, así como un mecanismo de comunicación electrónica con dichos órganos y entes disponible para todas las personas vía Internet.".

Así mismo, esta misma ley, en su artículo 148 expresa:

> "Los órganos y entes de la Administración Pública podrán incorporar tecnologías y emplear cualquier medio electrónico, informático, óptico o telemático para el cumplimiento de sus fines. Los documentos reproducidos por los citados medios gozarán de la misma validez y eficacia del documento original, siempre que se cumplan los requisitos exigidos por ley y se garantice la autenticidad, integridad e inalterabilidad de la información".

Es en este punto donde empieza a tener sentido la gestión de la calidad en el desarrollo de software, ya que una de las premisas es que el desarrollo de software por parte de los entes públicos debe generar productos confiables, seguros en la transmisión y manejo de datos, y asegurar su integridad. El artículo 161 de la Ley Orgánica de la Administración Pública cita,

> "…El sistema garantizará la constancia, en cada asiento que se practique, de un número, epígrafe expresivo de su naturaleza, fecha de entrada, fecha y hora de su presentación, identificación del interesado,

[3] El Sistema Integrado de Gestión y Control de las Finanzas Públicas (SIGECOF) es un sistema automatizado de información desarrollado por el Ministerio de Ciencia y Tecnología y financiado por el Banco Mundial, que tiene como fin controlar las finanzas públicas de los distintos organismos públicos de la administración central, para que se ejecuten bajo la estructura de control que establece la ley.

> órgano administrativo remitente, si procede, y persona u órgano administrativo al que se envía, y, en su caso, referencia al contenido del escrito o comunicación que se registra.
>
> Asimismo, el sistema garantizará la integración informática en el registro general de las anotaciones efectuadas en los restantes registros del órgano administrativo.”.

Es evidente que uno de los requerimientos por parte de la ley, es que los sistemas automatizados de información, que apoyen la gestión de la administración pública de cualquier órgano, deben garantizar la integridad y la calidad de la información manejada, y una manera de asegurar esto es mediante una buena gestión de calidad en cada una de las fases del ciclo de vida de dichos sistemas.

Si bien es cierto que la metodología para la gerencia de proyectos proporcionada por el Project Manager Institute (PMI) pretende gestionar cualquier tipo de proyectos de forma exitosa, los proyectos informáticos, es decir, proyectos de desarrollo de sistemas automatizados de información requieren consideraciones que no son tomados en cuenta en esta metodología.

Duncan Haughey en un artículo escrito para la revista electrónica “Project Smart” dice que de acuerdo con Standish Group[4] ya para 1995 solamente cerca del 16% de los proyectos de desarrollo de software eran exitosos o cumplían con lo programado, 53% eran defectuosos en cuando a funcionalidad y contenido o se pasaban de lo presupuestados y un 31% de los proyectos eran cancelados antes de culminar. Además, según las estadísticas de la misma compañía, el promedio de los proyectos de desarrollo de software es entregado en más del doble de tiempo estipulado inicialmente, se pasan del presupuesto estimado en un 189% y solo entregan un 61% de las funcionalidades especificadas inicialmente.

El alcance de la responsabilidad del aseguramiento de la calidad, en el desarrollo de software, abarca a varios actores de una organización, tales como ingenieros de software, líderes de proyecto, clientes, analistas y personas que trabajan en el equipo.

---

[4] Sandish Group (http://www.standishgroup.com) es una compañía con amplia experiencia en el manejo de proyectos de desarrollo de tecnología de información ubicada en Massachussets, EE.UU. especialistas en el manejo de riesgos, costos y retorno de la inversión en tecnología de información.

El equipo que gestione la calidad en el desarrollo de software debe analizar éste desde diversos puntos de vista, respondiendo a algunas de las siguientes preguntas:

- ¿Satisface el software, de forma adecuada los principales factores de calidad?
- ¿Se ha realizado el desarrollo del software siguiendo los acuerdos preestablecidos?
- ¿Se han aplicado las técnicas y métodos apropiados para el desarrollo del software?
- ¿El producto generado satisface las expectativas del cliente? ¿En qué medida?

Imponer un modelo de calidad dentro de una organización no es tarea fácil, requiere de mucha dedicación, esfuerzo y dinero por parte de la misma. Para citar un ejemplo, hasta agosto del 2004 solo existían 1543 organizaciones (que manejaban aproximadamente 7848 proyectos) en proceso de certificación CMM a nivel mundial, de los cuales el 59% se encontraban fuera de los Estados Unidos. Es relevante en este punto señalar que CMM es una de las metodologías que más auge ha tenido en los últimos años, en cuanto a la implantación de un sistema de gestión de calidad para el desarrollo de software.

En Venezuela no existen antecedentes bibliográficos de gestión de calidad para los proyectos de software desarrollados en los entes públicos, pero existe en el Ministerio de Ciencia y Tecnología un "**plan nacional de migración a software libre de la administración pública nacional**" donde se pretende dar lineamientos sobre este tema. Al respecto en dicho documento se cita:

> "…Los estándares del modelo de calidad de producto software se refieren a la norma ISO: 9126. En lo referente a la evaluación de la calidad del software, los detalles al respecto se encuentran en la norma ISO 14598. En forma obligatoria los Ministerios deben certificarse en estos estándares como mínimo".

Imponer alguno de estos modelos en un ente gubernamental es aún más difícil, ya que, en la mayoría de los casos, entran a jugar factores e intereses políticos y sobre todo económicos que dificultan la labor, ya que una certificación de este tipo, requiere que la organización esté ganada a la idea, y una inversión económica considerable.

Considerando lo anteriormente expuesto, es evidente que no existe una metodología para la gestión de calidad de los proyectos de desarrollo de software en los entes gubernamentales, por lo tanto, el objetivo de este trabajo es definir un proceso para gestionar la calidad de los proyectos de desarrollo de software en dichos organismos venezolanos durante su fabricación, basado en el análisis y estudio de modelos de gestión de calidad de software ya existentes en el mercado, incluyendo los mencionados por el Ministerio de Ciencias y Tecnología, de manera que el modelo generado sea de fácil aplicación y sobre todo de fácil inserción en la estructura donde se aplique.

Es importante señalar que en la mayoría de los casos, el desarrollo de software no esta dentro de las actividades medulares de los organismos estadales, por lo que existe un alto nivel de contratación de empresas outsourcing para desarrollar aplicaciones o de compra de software prefabricados, es por ello que se hace aún más difícil la aplicación de manera estricta de un control de calidad para el desarrollo del mismo.

En base a lo anteriormente expuesto, se genera la siguiente interrogante: ¿Cuáles deben ser las características de un modelo, que permita gestionar la calidad de los proyectos de desarrollo de software en los entes públicos venezolanos, que sea de fácil aplicación y por lo tanto de fácil inserción en dichos organismos?

A fin de dar respuesta(s) a la interrogante planteada, se plantearon los siguientes objetivos para la investigación

1.2 Objetivo general

Diseñar, en base a metodologías aceptadas, una metodología para la gestión de la calidad para el desarrollo de sistemas automatizados de información en la administración pública venezolana.

1.3 Objetivos específicos

- Establecer los objetivos de cada una de las fases de desarrollo de software.

- Identificar las entradas que sirven como insumos para cada una de las fases de desarrollo de software.
- Describir las actividades requeridas en cada una de las fases de desarrollo de software.
- Determinar los productos de cada fase del desarrollo de software.
- Determinar métricas de control de calidad en cada una de las fases del desarrollo del proyecto.

Justificación

Es evidente que el gobierno esta haciendo grandes esfuerzos para modernizar todas las entidades gubernamentales para de esa manera agilizar los tramites burocráticos y hacer más transparente la gestión de gobierno. Para lograr todo esto se esta apoyando en el desarrollo de sistemas de información automatizados aplicando la filosofía del Software Libre.

Si dicho crecimiento tecnológico no es llevado con orden y con un buen control de calidad se convertiría en un cáncer para las entidades gubernamentales llegando a ser un obstáculo en vez de una herramienta para el uso diario de la gestión de cada uno de estos organismos, y a su vez crearía un rechazo al uso de tecnología por parte de la ciudadanía.

Este diseño pretende ser una guía de acción necesaria para el control de la calidad en el desarrollo de sistemas de información automatizados para asegurar que los productos desarrollados cumplan con las expectativas de los clientes y lleguen a buen término.

Desde el punto de vista teórico, la presente investigación se justificó porque permitió al investigador aplicar, a una situación real, los conceptos relativos Control de la Calidad de Proyectos.

Marco Teórico

A lo largo de toda la historia la búsqueda y el afán de perfección por parte del hombre ha sido constante, de tal forma, que el interés por el trabajo bien hecho y la necesidad de asumir responsabilidades sobre la labor efectuada, lentamente derivó en el concepto de calidad.

Un ejemplo temprano se encuentra entre los 2000 y 3000 años A.C. cuando los faraones egipcios edificaron una obra civil de gran envergadura como lo son las famosas pirámides de Egipto. Muchas de ellas tienen parámetros que las acercan casi a la perfección en la construcción pues en la orientación de la base con respecto a la alineación N-S, E-O el error máximo llega a ser de 6 minutos de arco, distando la base de algunas de ellas de ser un cuadrado perfecto por menos de 17,78 cm. Todo ello se conseguía gracias a los métodos de inspección empleados durante su construcción.

Mucho más tarde, ya entrada la edad media surgió en Europa el sistema de organización en gremios. Estos imponían los precios y especificaciones de los distintos productos de los que proveían a la sociedad. Los productos de calidad daban prestigio al artesano, así como al gremio de la zona cuando todos sus artesanos seguían sus especificaciones. Este hecho constituye una de las primeras pruebas de un organismo que se encarga tanto de fijar unas normas básicas, como de controlar su cumplimiento.

Con la revolución industrial comienza a desaparecer el artesanado, se crean grandes organizaciones y los antiguos artesanos se transforman en los trabajadores de las empresas. En esta época, específicamente en 1911, Taylor elaboró su teoría acerca de la "gestión científica del trabajo" en varias publicaciones, cuyo objeto fue la preparación de normas para que los trabajadores las cumpliesen. Comenzó con ello la instauración paulatina de la división del trabajo, lo que suponía que los operarios interviniesen solamente en algunas operaciones del proceso productivo. Este hecho provocó la necesidad de que surgiese la figura de los empleados dedicados a tareas de inspección,

aunque se prestaba más atención a la forma de realizar el trabajo (los procesos) que a la calidad de los productos.

Finalmente el control de calidad estadístico comenzó en los años 30 del siglo XX con la aplicación industrial del cuadro de control ideado por el Dr. W. A. Shewhart, de Bell Laboratories, que fue el inventor de los conocidos gráficos de control[5].

Continuando en este proceso cronológico destaca el hecho de que pasados unos años del final de la II Guerra Mundial los japoneses comienzan a hacer verdadero énfasis en la calidad. En 1950 la Unión de Científicos e Ingenieros Japoneses[6] realizó un seminario cuyo conferenciante, el Dr. W.E. Deming, desarrolló los siguientes temas: Cómo mejorar la calidad mediante el ciclo de planear, hacer, verificar y actuar; La importancia de captar la dispersión en las estadísticas; Control de procesos mediante el empleo de cuadros de control y su aplicación.

Cuatro años más tarde el Dr. J.M. Juran introdujo en Japón la idea de que la calidad de un producto o servicio residía en el grado de mentalización de todo el personal de la organización e impartió seminarios a los mandos altos y medios de las empresas niponas, explicándoles las funciones que les correspondían a cada uno en la promoción del control de calidad.

La visita de Juran al Japón, marcó una transición en las actividades de control de calidad y creó un ambiente en que se reconoció el Control de Calidad como un instrumento de gerencia abriéndose las puertas para el establecimiento del control total de calidad tal como se concibe hoy.

.

[5] El gráfico de control, también conocida como el "gráfico de Shewhart" o "gráfico del proceso-comportamiento" es una herramienta estadística prevista para determinar la naturaleza de la variación en un proceso y para facilitarla el pronóstico y gerencia.

[6] La Unión de Científicos e Ingenieros Japoneses es una asociación privada integrada por ingenieros e investigadores. Es una entidad independiente del gobierno y no lucrativa, que aúna a un grupo de empresarios, gente del gobierno y académicos, nacida en 1948, que reconocen las graves deficiencias de calidad de los productos japoneses, así como las repercusiones de dicho problema en el mercado exterior, por lo cual, deciden fundar un comité de investigación, el "Q. C. Rercarch Group", cuyo fruto inmediato son los primeros cursos del Control de Calidad en todo el país (Japón).

2.1 Bases Legales

El 28 de Diciembre del año 2004, se publica en Gaceta Oficial Nº 38.095 el Decreto 3.390, mediante el cual se dispone que la Administración Pública Nacional empleará prioritariamente Software Libre desarrollado con Estándares Abiertos, en Sistemas, Proyectos y Servicios Informáticos. Este Decreto da apoyo a una serie de leyes que se gestaron mucho antes que permiten y obligan el desarrollo informático a los distintos órganos del estado.

Dicho decreto lo que persigue es la aplicación de protocolos de comunicación considerados por el mercado como estándares abiertos, y la mejora de la calidad de los productos de software que apoyan la gestión diaria de los entes gubernamentales para, de esta manera, fortalecer la industria del software en el país, y que la interconexión de los productos a desarrollar fuese de fácil aplicación.

En base a lo anteriormente expuesto, no se puede dejar de lado la gestión de calidad en dichos productos para lograr los objetivos, por los tanto es de primordial importancia que se desarrolle una buena gestión de calidad y es la utilidad que este trabajo puede aportar.

2.2 Bases Teóricas

2.1 Calidad

Para entender el presente trabajo de investigación es recomendable comprender el concepto de calidad, ya que es el objetivo principal a lograr; una gestión de calidad en el desarrollo de un proyecto de software.

Mucho se ha escrito sobre el tema; diversos autores han tratado de definir lo que es calidad, y han surgido grandes tratados sobre el mismo. En la norma UNE-EN ISO 9000:2002 se ofrece una definición de calidad bastante genérica y de aplicación en distintos campos: la calidad es “el grado en el que un conjunto de características inherentes cumple con los requisitos”. Esta norma hace referencia a dos grupos de requisitos: las necesidades o expectativas establecidas y las implícitas u obligatorias (estos requisitos se presuponen que deben existir).

La American Society for Quality, una de las más prestigiosas instituciones internacionales en materia de calidad, fundada en en 1.946, a través de su site (http://www.asq.org) define la calidad como: “un termino subjetivo, para lo cual cada persona tiene su propia definición. En el uso técnico, la calidad puede tener dos significados: 1.- las características de un producto o servicio que les permite satisfacer una determinada necesidad. 2.- un producto o servicio libre de deficiencias.”.

Una vez revisados los antecedentes del concepto calidad y vistos unos conceptos previos, para concretar su significado se van a reproducir tres definiciones emanadas de personas y entidades reconocidas en el área.

Para Deming y Juran la calidad es un "conjunto de esfuerzos efectivos de los diferentes grupos de una organización para la integración del desarrollo, del mantenimiento y de la superación de la calidad de un producto, con el fin de hacer posible la fabricación y servicio a satisfacción completa del consumidor y al nivel más económico". Así mismo Deming también indica que el control de calidad no significa alcanzar la perfección. Significa conseguir una eficiente producción con la calidad que espera obtener en el mercado

Kaoru Ishikawa en 1988 define la calidad como “en su interpretación más estrecha, calidad significa calidad del producto, pero en su interpretación más amplia significa calidad del trabajo, calidad del servicio, calidad de la información, calidad del proceso, calidad de la dirección, calidad de la empresa”.

Para la consultora Arthur Andersen[7] la gestión de calidad en la empresa es el proceso de identificar, aceptar, satisfacer y superar constantemente las expectativas y necesidades de todos los colectivos humanos relacionados con ella, clientes, empleados, directivos, propietarios, proveedores y la comunidad con respecto a los productos y servicios que esta proporciona.

Sobre la definición de calidad existe un universo de definiciones, pero en resumidas cuentas la calidad de un producto o servicio es la percepción que el cliente

[7] Consultora fundad en Chicago, Illinois, EE.UU. en 1913 por Arthur Andersen and Clarence DeLany

tiene del mismo, es decir, es un conjunto de propiedades inherente al producto o servicio que permiten apreciarlo como igual, mejor o peor que el resto de los objetos de los de su especie.

De todas estas definiciones se extraen una serie de parámetros básicos que definen la calidad: si se desea producir productos de buena calidad para el consumidor será necesario decidir por adelantado la calidad del producto (o servicio) planificarlo (establecer la calidad de diseño) y producirlo (establecer la calidad de fabricación, que indudablemente es distinta a la calidad del producto final).

2.2 Calidad del software

A la hora de definir la calidad del software es muy importante hacer notar la diferencia entre la calidad del producto software y la calidad del proceso de desarrollo de éste (calidad de diseño y fabricación). No obstante, las metas que se establezcan para la calidad del producto van a determinar los objetivos a establecer de calidad del proceso de desarrollo, ya que la calidad del primero va a depender, entre otros aspectos, de ésta. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto.

En este punto es válida la pregunta, ¿son la calidad del diseño y la calidad de fabricación los únicos aspectos que deben ser considerados por el equipo de trabajo?. Robert Glass (1998) establece una relación más intuitiva:

Satisfacción del usuario = producto satisfactorio + buena calidad + entrega dentro del presupuesto acordado + entrega dentro del tiempo establecido

Particularmente, en el caso del software, existen muchas definiciones distintas en la bibliografía, sin embargo, en lo que a este trabajo respecta, se tomará la siguiente definición:

> "La calidad del software se define como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares y procesos de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente." (Presuman, 2001).

No hay duda de que la anterior definición puede ser modificada o ampliada. De hecho, no tendría fin una discusión sobre una definición de la calidad del software. Para los propósitos de este enfoque, la anterior definición sirve para hacer hincapié en tres puntos importantes:

- Los requisitos del software son la base de las medidas de la calidad. La falta de concordancia con los requisitos es una falta de calidad.
- Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la ingeniería del software o del conocimiento. En caso de no seguirse esos criterios, casi siempre habrá falta de calidad.
- Existe un conjunto de requisitos implícitos que a menudo no se mencionan (por ejemplo la necesidad de una interfaz intuitiva). Si el software se ajusta a sus requisitos explícitos pero falla en alcanzar los requisitos implícitos, la calidad del software se debilita.

Queda claro a partir de la definición de calidad del software, que ésta es siempre relativa a los requisitos o necesidades que se desean satisfacer. Por eso la evaluación de la calidad del software siempre va a implicar la comparación entre los requisitos y el producto generado.

También es importante destacar que la calidad de un producto software debe ser considerada en todos sus estados de evolución (especificaciones, diseño, códigos,...). No basta con verificar la calidad del producto una vez finalizado cuando los problemas de mala calidad ya no tienen solución o su reparación es muy costosa.

La problemática general a la que se enfrenta el software es:

- Aumento constante del tamaño y complejidad de los programas.
- Carácter dinámico e iterativo a lo largo de su ciclo de vida, es decir que los programas de software a lo largo de su vida cambian o evolucionan de una versión a otra para mejorar las prestaciones con respecto a las anteriores.
- Dificultad de conseguir productos totalmente depurados, ya que en ningún caso un programa será perfecto.

- Se dedican elevados recursos monetarios a su mantenimiento, debido a la dificultad que los proyectos de software entrañan y a la no normalización a la hora de realizar los proyectos.
- No suelen estar terminados en los plazos previstos, ni con los costes estipulados, ni cumpliendo los niveles deseables de los requisitos especificados por el usuario.
- Incrementos constantes de los costes de desarrollo debido entre otros, a unos niveles de productividad bajos.
- Los clientes tienen una alta dependencia de sus proveedores por ser en muchos casos aplicaciones a "medida".
- Procesos artesanales de producción con escasez de herramientas.
- Insuficientes procedimientos normalizados para estipular y evaluar la productividad, costes, y calidad.

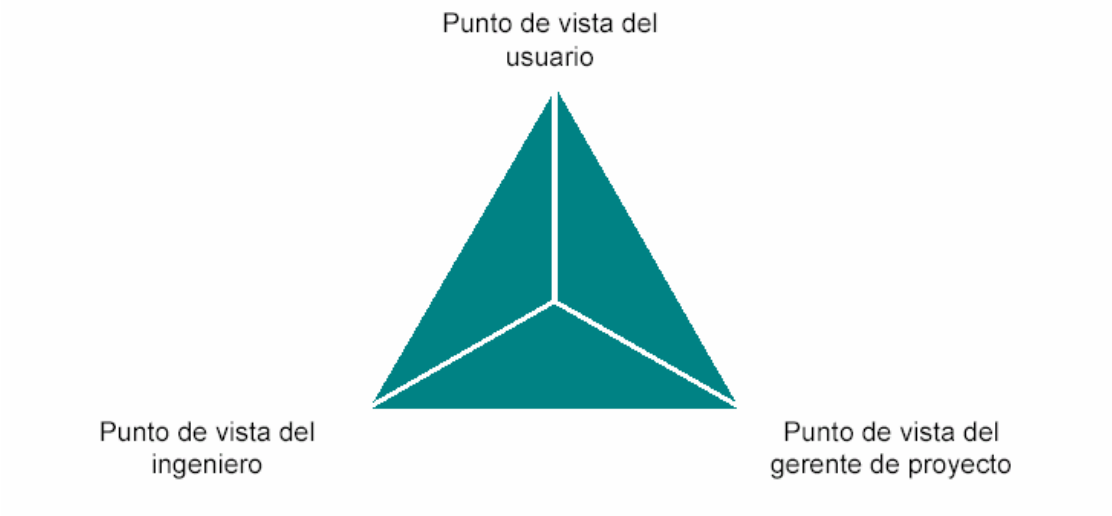
Todo lo anterior puede concretarse en:

- Ausencia de especificaciones completas, coherentes y precisas previas por parte del cliente, así como posteriores por parte de los proveedores del software.
- Ausencia de la aplicación sistemática de métodos, procedimientos y normas de ingeniería del software.
- Escasez o ausencia de entornos integrados de programación.
- Escasez de uso de técnicas actuales y automatizadas para la gestión de proyectos.
- Escasez de personal con formación y experiencia en los nuevos métodos, normas y uso de entornos y utilidades de programación.
- Otros derivados del grado de desarrollo técnico y organizativo de cada compañía.

2.2.1 La calidad del software bajo distintos puntos de vista

En la ingeniería del software, la visión de la calidad no es única, dependiendo del punto de vista desde el cual se la analice. Esto es importante entenderlo, ya que en muchos casos va a determinar el alcance del modelo de calidad a seguir. En la siguiente figura se presentan los distintos puntos de vista.

Gráfico 1Punto de vista de la calidad del software según diversos autores



Fuente: Diseño del investigador

Dependiendo del punto de vista, se priorizarán distintos factores del software:

- Punto de vista del usuario: El punto de vista del usuario, con respecto a la calidad, estará basado en los factores externos del producto, tales como funcionalidad y facilidad de operación.
- Punto de vista del ingeniero de software: El punto de vista del ingeniero del software, con respecto a la calidad, estará basado en los factores internos del producto, tales como modularidad y reusabilidad.
- Punto de vista del gerente del proyecto: El punto de vista del gerente del proyecto, con respecto a la calidad, estará basado en los factores relacionados con la gestión del proyecto, tales como costos y cronogramas acorde a lo planificado, etc.

Todos estos puntos de vista, como ya lo hemos mencionado, inciden en la calidad del producto.

2.2.2 Factores que determinan la calidad de software

Existen muchos factores que afectan a la calidad del software y se pueden clasificar de distintas formas (uno de ellos es el punto de vista del apartado anterior). En este trabajo se presentarán sólo a modo descriptivo, algunos factores de calidad que se han propuesto.

Mc Call, J., Richards, P., Walters, G en el libro "**Factors in Software Quality**" (1977) han propuesto los siguientes:

- Corrección. El grado en que un programa satisface sus especificaciones y consigue los objetivos de la misión encomendada por el cliente. Responde a la pregunta: ¿Hace lo que quiero?
- Fiabilidad. El grado en que se puede esperar que un programa lleve a cabo sus funciones esperadas con la precisión requerida (Hay que decir que se han propuesto otras definiciones de fiabilidad más completas). Responde a la pregunta: ¿Lo hace en forma fiable todo el tiempo?
- Eficiencia. La cantidad de recursos de computadora y de código requeridos por un programa para llevar a cabo sus funciones. Responde a la pregunta: ¿Se ejecutará en mi hardware lo mejor que se pueda?
- Integridad. El grado en que puede controlarse el acceso al software o a los datos, por personal no autorizado. Responde a la pregunta: ¿Es seguro?
- Facilidad de uso. El esfuerzo requerido para aprender un programa, trabajar con él, preparar su entrada e interpretar su salida. Responde a la pregunta: ¿Está diseñado para ser usado?
- Facilidad de mantenimiento. El esfuerzo requerido para localizar y arreglar un error en un programa (Se trata de una definición muy limitada). Responde a la pregunta: ¿Permite ser corregirlo con relativa facilidad?
- Flexibilidad. El esfuerzo requerido para modificar un programa operativo. Responde a la pregunta: ¿Permite ser cambiado con relativa facilidad?
- Facilidad de prueba. El esfuerzo requerido para probar un programa de forma que se asegure que realiza su función requerida. Responde a la pregunta: ¿Permite ser probado con relativa facilidad?
- Portabilidad. El esfuerzo requerido para transferir el programa desde un hardware y/o un entorno de sistemas de software a otro. Responde a la pregunta: ¿Podré usarlo en otra computadora?
- Reusabilidad: El grado en que un programa (o partes de un programa) se puede reusar en otras aplicaciones. Esto va relacionado con el empaquetamiento y el alcance de las funciones que realiza el programa. Responde a la pregunta: ¿Podré reusar alguna parte del software?
- Facilidad de interoperación. El esfuerzo requerido para acoplar un sistema a otro. Responde a la pregunta: ¿Podré hacerlo interactuar con otros sistemas?

Otra lista de factores de calidad es la desarrollada por R. Grady y D. Caswell en 1987 en el libro "**Software Metrics: Establishing a Company-Wide Program**". Los factores y sus atributos correspondientes son los siguientes:

- La funcionalidad se obtiene mediante la evaluación del conjunto de características y de posibilidades del programa, la generalidad de las funciones que se entregan y la seguridad de todo el sistema.
- La facilidad de uso se calcula considerando los factores humanos, la estética global, la consistencia y la documentación.

- La fiabilidad se calcula midiendo la frecuencia de fallos y su importancia, la eficacia de los resultados de salida, el tiempo medio entre fallos (MTBF), la posibilidad de recuperarse a los fallos y la previsibilidad del programa.
- El rendimiento se mide mediante la evaluación de la velocidad de proceso, el tiempo de respuesta, el consumo de recursos, el rendimiento total de procesamiento y la eficiencia.
- La capacidad de soporte combina la posibilidad de ampliar el programa (extensibilidad), la adaptabilidad y la utilidad (estos tres atributos representan un término más común — facilidad de mantenimiento), además de la facilidad de prueba, la compatibilidad, la posibilidad de configuración [posibilidad de organizar y controlar elementos de la configuración & software], la facilidad con la que se puede instalar un sistema y la facilidad con la que se pueden localizar los problemas.

Tanto el modelo de McCall como el de Grady, presentan además fórmulas, matrices, pesos ponderados que permiten cuantificar cada uno de los factores presentados. Más allá de eso, los siguientes puntos deben quedar claros:

- Los modelos presentados son sólo una muestra de los disponibles, hay muchos más y se actualizan frecuentemente.
- Al planificar la calidad de un producto de software se debe seleccionar cuales de los factores de calidad van a ser considerados requisitos, a su vez, para realizar esta selección, se debe tener en cuenta lo siguiente:
    - Las características particulares de la aplicación a desarrollar o de su entorno. Así por ejemplo si la aplicación se desarrolla para un entorno en el que el hardware evoluciona rápidamente, el factor “portabilidad” es importante; si se espera que las especificaciones del sistema cambien frecuentemente, la “flexibilidad” será esencial.
    - El costo de los factores de calidad frente al beneficio que proporcionan. Es decir realizar un análisis costo-beneficio.
    - Las interrelaciones entre factores: Algunos factores pueden ser conflictivos entre sí. La eficiencia, por ejemplo, está en conflicto con otros factores de calidad.
- Es necesario medir cada uno de los factores y atributos seleccionados. Algunos pueden ser medidos directamente y otros sólo pueden ser medidos indirectamente. En cualquiera de los dos casos la cuantificación es obligatoria. Respondiendo a otro de los principio del paradigma de la calidad (orientación a datos), las comparaciones se deben basar sobre datos y mediciones concretas y objetivas, no sobre opiniones o subjetividades.

2.2.3 La calidad a través de la normalización en la ingeniería del software y su problemática

La normalización consiste en un proceso donde se elaboran guías, normas y convenciones sobre una determinada materia, con el objeto de definir, simplificar y especificar las actividades relacionadas con la materia de que se trate (Cervera, 2002).

La Ingeniería del Software se ha ido desarrollando en los últimos 15 años, a través de la creación e implantación en la industria del software de métodos, procedimientos y técnicas que tratan de cubrir las necesidades de cada una de las etapas del ciclo de vida de un producto de software, desde la definición de sus requisitos hasta su mantenimiento una vez el producto comience a emplearse, y ello con las restricciones generales de todos los procesos modernos de ingeniería, esto es, la necesidad creciente de incrementar la productividad de la programación mejorando y garantizando, simultáneamente la calidad del producto resultante (wikipedia, 2005).

La creación e implantación de normas de desarrollo del software son un autentico desafío que tiene la ingeniería del software como medio de comunicación para transferir sus métodos, técnicas y procedimientos a la industria del software para el diseño y desarrollo de nuevos productos. Estas normas tienen como criterio general de desarrollo maximizar la comunicación entre los profesionales del software a través de la definición de documentos generales que se han de producir, proveyendo de guías que indican a nivel de detalle el contenido de dichos documentos y recomendaciones de las actividades que hay que realizar durante todo el proceso de producción del software. En pocas palabras, las normas de ingeniería del software son la solución a una de las mayores necesidades de la industria del software actual: la comunicación más adecuada y precisa entre sus profesionales.

A medida que ha ido aumentando la necesidad de un software más fiable, se ha reconocido que las normas de ingeniería del software son una contribución fundamental para asegurar la producción de software de calidad. Además una consecuencia del

objetivo genérico de mejorar la comunicación es que se reducen los costes por un aumento de productividad y una mejora de la calidad de los desarrollos de software.

En relación a las normas los profesionales se encuentran con un problema fundamental: la dispersión de las normas relativas al software que, con frecuencia, han sido creadas por organismos muy diversos, bajo enfoques distintos y destinadas a ámbitos de actuación diferentes. Muchas compañías, por su parte, se han visto obligadas a generar sus propias normas cuando no disponen de unas de ámbito general. De hecho muchas organizaciones desarrollan sus propios conjuntos de normas adecuándolas a sus fines específicos. Pero también se dan casos en que organizaciones distintas tienen los mismos objetivos por lo que resultaría razonable su colaboración y, en todo caso, la adopción de las normas de la organización que tenga más avanzados sus desarrollos y un ámbito de actuación más amplio. Puede afirmarse que en la actualidad se ha llegado a un nivel de madurez en la industria del software que ha permitido a todos los implicados que exista un interés por aunar sus experiencias y esfuerzos para crear normas generales que abarquen sus áreas de interés.

Estos esfuerzos varían en cuanto al tipo de industrias o usuarios así como en lo relativo a los logros alcanzados, pero la tendencia actual es hacia la normalización del proceso de desarrollo software a través de normas que conduzcan a homogeneizar los planes de garantía de calidad de él, los planes de gestión de la configuración del software, la documentación de sus pruebas, etc.

De hecho en EEUU existen varias entidades públicas y privadas que publican normas en diversos aspectos del software entre las que destacan: el American Institute of Aeronautics and Astronautics (AIAA), la American Nuclear Society (ANS), la American Society of Quality Control (ASQC), la Data Processing Management Association (DPMA) y la Electronic Industries Association (EIA).

Sin embargo las actividades de normalización en ingeniería del software son relativamente recientes, siendo tres las instituciones de mayor prestigio y difusión que las generan: el Instituto de Ingenieros Eléctricos y Electrónicos de los EE.UU. (IEEE) a través del Subcomité de Normas de Ingeniería del Software de la Computer Society, el National Bureau of Standards (NBS) que publica las normas FIPS (Federal Information

Proccessing Standards) o normas a aplicar en todos los Estados Federales y el Departamento de Defensa (DOD -USAF, ARMY y NAVY-) que publica normas en el terreno militar.

En cuanto al proceso de creación e implantación de las normas, se hará uso de las dictadas por el IEEE: una nueva idea puede tenerla y sugerirla cualquiera de sus miembros, enviandola a continuación al Comité de Normas de Ingeniería del Software, el cual cursará una Petición de Autorización de Proyecto (PAP) que se transmite a todos los grupos que hacen normas y al ANSI (American National Standard Institute) que se encargará de la coordinación.

La PAP define el objetivo, ámbito, descripción y principales contactos para llevar a cabo el proyecto. La etapa siguiente corresponde al desarrollo de la norma a cargo de un grupo de trabajo. Cualquiera puede adherirse al grupo y el IEEE se encargará de implicar a todos los interesados para que participen en reuniones con los demás miembros (unas cuatro reuniones anuales). También pueden intervenir a distancia detallando las contribuciones oportunas y remitiéndolas al grupo correspondiente. El proceso será más enriquecedor cuantos más profesionales de distinto nivel y más organismos intervengan.

Una vez terminado un desarrollo se concretará en un documento (borrador o draft) que pasará por varias revisiones como consecuencia del proceso de valoración y votación para aceptarlo. Durante este periodo habrán de votar al menos el 75% de los miembros del grupo de votación (unos 100 expertos) y a su vez el 75% debe ser positivo. Tras su aprobación sé distribuirá la nueva norma. En cuanto al tiempo medio que se tarda desde que se inicia la elaboración de una norma hasta que se aprueba es de tres años. Finalmente, como trabajo complementario se procede difundir la norma a través de Seminarios.

La etapa siguiente en este ciclo corresponde con la de uso de la norma por las empresas u organismos. Durante esta fase los usuarios proporcionan comentarios positivos o negativos acerca de la norma, según su propia experiencia, y en ocasiones aceptan o modifican las normas para sus fines específicos. Las modificaciones se remiten al consejo de normas del IEEE para su evaluación por el grupo de trabajo entrando en una nueva fase del ciclo de elaboración de normas. Finalmente se decidirá dejarla igual,

modificarla o eliminarla si ya ha sido superada por una nueva norma. El ciclo completo dura cinco años.

Como se puede ver, generar normas que apliquen en la ingeniería del software es un proceso largo y complicado, por lo que muchas organizaciones optan por adaptar las existentes de manera que satisfagan los requerimientos.

2.3 Desarrollo de software y la gerencia de proyectos

Evidentemente el desarrollo de software se puede considerar, en toda su magnitud, un proyecto, pues cumple con todos las características que lo distinguen según los criterios del PMI, es decir, tiene un tiempo definido, un alcance estipulado, y es, sin lugar a duda un esfuerzo temporal que genera un producto, servicio o resultado único.

Para corroborar lo dicho anteriormente, se puede decir que existen varios esfuerzos a nivel mundial para conseguir una manera de gestionar el desarrollo de software como un proyecto. Para tal caso, sin ir muy lejos, en el PMI capitulo Colombia, existen investigaciones donde sugieren agregar un capitulo al PMBOK que solo trate el tema de gerencia de proyectos para desarrollo de sistemas de información dado la particularidad del mismo (Montes, G. 2002).

Según la metodología utilizada para el desarrollo de software, existen distintos ciclos de vida, que van desde los más sencillos (análisis, diseño, desarrollo y entrega), hasta los ciclos de vida en espiral que buscan asegurar la calidad en el desarrollo versionado de los sistemas a través del tiempo.

Parte de esta investigación consiste en asociar el ciclo de vida de un proyecto con el ciclo de vida del desarrollo de software para lograr un mejor gestión de la calidad en el producto final, además esto me asegura que las gerencias gubernamentales dedicadas al desarrollo de sistemas de información que evidentemente no son estructuras proyectizadas, puedan gestionar el desarrollo de sistemas de información como un proyecto de manera transparente, solo siguiendo el ciclo de vida del desarrollo de software sin que ello implique algún cambio dentro de la estructura organizacional.

2.4 Los modelos de calidad del software de gestión

En este apartado se presenta, como propuesta de ayuda para mejorar la calidad del software de gestión, una síntesis del modelo de McCall por ser uno de los más difundidos y porque además ha servido de base para otros modelos (el modelo de Boehm y el Software Quality Management –SQM- de Murine).

En general los modelos de calidad definen a ésta de forma jerárquica, es decir la calidad se produce como consecuencia de la evaluación de un conjunto de indicadores o métricas en diferentes etapas.

En el nivel más alto de la jerarquía se encuentran los factores de calidad definidos a partir de la visión del usuario del software, y conocidos también como atributos de calidad externos.

Cada uno de los factores se descompone en un conjunto de criterios de calidad, o sea aquéllos atributos que cuando están presentes contribuyen a obtener un software de la calidad. Se trata de una visión de la calidad técnica, desde el punto de vista del producto software y se les denomina también atributos de calidad internos. Por lo general estos atributos no son requerido de manera explicita por el usuario del software, pero son atributos que si no aparecen en escena, probablemente el software desarrollado no es un producto de aceptable, así cumpla con todos los atributos de calidad externos. Un ejemplo de este tipo de atributo es el tiempo de respuesta que una aplicación puede tener. Este tipo de especificación, pocos usuarios la solicitan, sin embargo, aunque un producto de software cumpla con todos los requerimientos explícitos del cliente, si tiene un tiempo de respuesta muy alto, pues será considerado de baja calidad.

Finalmente para cada uno de los criterios de calidad se definen un conjunto de métricas o medidas cuantitativas de ciertas características del producto que indican el grado en que dicho producto posee un determinado atributo de calidad.

De esta manera, a través de un modelo de calidad se concretan los aspectos relacionados con ella de tal manera que se puede definir, medir y planificar. Además el empleo de un modelo de calidad permite comprender las relaciones que existen entre diferentes características de un producto software.

En contra de los modelos de calidad pesa que aún no ha quedado demostrada la validez absoluta de ninguno de ellos, pues el éxito de estos modelos depende del equipo que lo implemente. Este punto es importante señalarlo, ya que, si el equipo de proyecto no esta ganado a la idea de generar productos de calidad, por muy bueno que sea el modelo a aplicar, es imposible tener un producto de calidad. En consecuencia, se puede decir que un buen modelo de calidad tiene que ir complementado, necesariamente con un cierto grado de aceptación por parte del equipo de proyecto que lo va a aplicar.

2.5 Alcance de la función del aseguramiento de la calidad del software (SQA[8])

La calidad del software no es algo en lo que se empieza a pensar una vez que se ha generado el código. Según R. Pressman(2001) el aseguramiento de calidad del software (SQA por sus siglas en ingles) es:

Una "actividad de protección" que se aplica a lo largo de todo el proceso de ingeniería software y engloba:

- Un enfoque de gestión de la calidad.
- Tecnología de ingeniería del software o del conocimiento efectivo (métodos y herramientas).
- Revisiones técnicas formales que se aplican durante cada paso de la ingeniería del software o del conocimiento.
- Una estrategia de prueba en múltiples niveles.
- El control de la documentación del software y de los cambios realizados.
- Un procedimiento que asegure un ajuste a los estándares de desarrollo del software (cuando sea posible).

[8] Software Quality Assurance.

- Mecanismos de medición y de información.

Lo anteriormente expuesto involucra tanto a los ingenieros de software como al equipo de SQA. Los ingenieros de software deben aplicar métodos técnicos y mediciones sólidas, conducir revisiones técnicas formales y ejecutar pruebas de software bien planificadas. Por otro lado, de acuerdo al Software Engineering Institute (SEI) (Carnegie Mellon University - Software Engineering Institute, 1994), el equipo de SQA tiene como propósito proveer a la gerencia la visibilidad apropiada del proceso que está siendo usado y de los productos siendo desarrollados. El propósito involucra:

- Revisar y auditar los productos de software y actividades para asegurar que obedecen a los procedimientos y estándares aplicables.
- Proveer al gerente del proyecto de software y a otras gerencias, según corresponda, los resultados de las revisiones y auditorias. Las discrepancias son primero planteadas dentro del proyecto de software y resueltas allí, si es posible. Si no se pueden resolver, se debe escalar a niveles superiores para su resolución.

Las actividades del equipo de SQA, acorde al SEI (Carnegie Mellon University - Software Engineering Institute, 1994) comprenden:

- Preparar un plan de SQA para el proyecto: El plan es desarrollado durante la planificación del proyecto y es revisado por todas las partes interesadas, para asegurar que cumple con las expectativas de los involucrados. Las actividades de aseguramiento de la calidad a desempeñar por el equipo de ingenieros de software y el equipo de SQA son regidas por este plan. El plan identifica:
    - Evaluaciones a realizar.
    - Auditorias y revisiones a realizar.
    - Estándares aplicables al proyecto.
    - Procedimientos para reporte y seguimiento de errores.
    - Documentos a ser producidos por el equipo de SQA.
    - Retro-alimentación a proveer al equipo del proyecto de software.
- Participar en el desarrollo de la descripción del proceso de software del proyecto: El equipo de software selecciona un proceso de software para el proyecto, el equipo de SQA revisa la descripción del proceso, verificando que cumpla con las políticas de la organización, estándares de software internos y estándares impuestos externamente, entre otros.
- Revisar las actividades de ingeniería del software o de conocimiento para verificar que cumplan con el proceso de software definido: El equipo de SQA identifica, documenta y hace el seguimiento de cualquier desviación del proceso y verifica las correcciones realizadas.
- Auditar productos de trabajo de software designados, para verificar que cumplan con aquellos definidos como parte del proceso de software: El equipo de SQA revisa productos seleccionados; identifica, documenta y

hace el seguimiento de las desviaciones; verifica las correcciones realizadas y periódicamente informa los resultados de su trabajo al gerente del proyecto.

- Asegurar que las desviaciones detectadas sean documentadas y manejadas de acuerdo a procedimientos documentados: Las desviaciones pueden encontrase en el plan de proyecto, descripciones del proceso, estándares aplicables o productos de trabajo técnico.
- Registrar cualquier incumplimiento y reportarlo a la alta gerencia: A los incumplimientos se les debe hacer seguimiento hasta que sean resueltos.

El equipo de SQA participa también en la tarea de recolectar y analizar métricas de software como también en establecer y revisar procedimientos, planes y estándares.

2.5.1 Control de Calidad

El control de calidad consiste en una serie de inspecciones, revisiones y pruebas utilizadas a lo largo del proceso de desarrollo del software para asegurar que cada producto cumple con los requisitos que le han sido asignados. El control de calidad incluye un bucle de realimentación (feedback) del proceso que creó el producto. La combinación de medición y realimentación permite afinar el proceso cuando los productos de trabajo creados fallan al cumplir sus especificaciones. Este enfoque ve el control de calidad como parte del proceso de fabricación (Presuman, 2001).

En este sentido se puede afirmar que las actividades de control de calidad pueden ser manuales, completamente automáticas o una combinación de herramientas automáticas e interacción humana. Un concepto clave del control de calidad es que se hayan definido todos los productos y las especificaciones mensurables en las que se puedan comparar los resultados de cada proceso. El bucle de realimentación es esencial para reducir los defectos producidos.

2.5.2 Costo de la calidad

La mayoría de las empresas desarrolladoras de software no tienen un estimado de cuanto es el costo que genera corregir un error luego que el producto está finalizado. Muchos ignoran que ese costo, a la larga, puede ser mucho mayor que el costo total del proyecto.

En las gerencias encargadas del desarrollo de software de los organismos públicos pasa lo mismo. Este punto es importante resaltarlo, ya que la mayoría de las veces, el tema económico puede afectar la factibilidad de un proyecto, y con tan solo planificar la calidad, y aplicando un modelo de control de calidad en la gerencia del proyecto, se puede reducir significativamente estos costps.

El costo de la calidad incluye todos los gastos de cualquier índole generados por la búsqueda de la calidad o en las actividades relacionadas en la obtención de la misma. Se realizan estudios sobre el costo de calidad para proporcionar una línea base del costo actual de calidad, para identificar oportunidades de reducirlo, y para proporcionar una base normalizada de comparación. La base de normalización siempre tiene un precio. Una vez que se han normalizado los costos de calidad sobre un precio base, tenemos los datos necesarios para evaluar el lugar en donde hay oportunidades de mejorar nuestros procesos. Es más, se puede evaluar cómo afectan los cambios en términos de dinero.

Según Roger Presuman (2001), los costos de la calidad se pueden dividir en costos asociados con la prevención, la evaluación y los fallos. Entre los costos de prevención se incluyen:

- planificación de la calidad,
- revisiones técnicas formales,
- equipo de pruebas,
- formación.

Entre los costos de evaluación se incluyen actividades para tener una visión más profunda de la condición del producto «la primera vez a través de» cada proceso. A continuación se incluyen algunos ejemplos de costos de evaluación:

- inspección en el proceso y entre procesos,
- calibrado y mantenimiento del equipo,
- pruebas.

Los costos de fallos son los costos que desaparecerían si no surgieran defectos antes del envío de un producto a los clientes. Estos costos se pueden subdividir en costos

de fallos internos y costos de fallos externos. Los internos se producen cuando se detecta un error en el producto antes de su envío. Entre estos se incluyen:

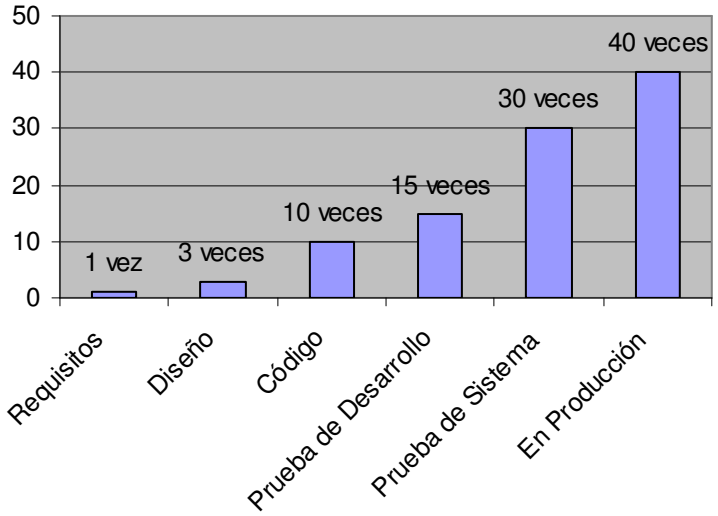
- retrabajo (revisión),
- reparación,
- análisis de las modalidades de fallos.

Los costos de fallos externos son los que se asocian a los defectos encontrados una vez enviado el producto al cliente. A continuación se incluyen algunos ejemplos de costes de fallos externos:

- resolución de quejas,
- devolución y sustitución de productos,
- soporte de línea de ayuda,
- trabajo de garantía.

Como es de esperar, los costos relativos para encontrar y reparar un defecto aumentan dramáticamente a medida que se cambia de prevención a detección y desde el fallo interno al externo. El gráfico siguiente, basado en datos recopilados en el libro "Software Engineering Economics" de Barry W. Boehm, publicado en el año 1981, ilustra este fenómeno.

Gráfico 2 Costo relativo de corregir un error



Fuente: Presuman, R. (2001)

Kaplan y sus colegas en el libro “Secrets of Software Quality: 40 Innovations from IBM” en 1995, refuerzan las estadísticas de costes anteriores informando con datos anecdóticos basados en un trabajo realizado en las instalaciones de desarrollo de IBM en Rochester:

> “Se han dedicado 7.053 horas inspeccionando 200.000 líneas de código con el resultado de 3.112 errores potenciales descubiertos. Dando por sentado un coste de programador de 40 dólares por hora, el coste de eliminar 3.112 defectos ha sido de 282.120 dólares, o aproximadamente unos 91 dólares por defecto.
>
> Compare estos números con el coste de eliminación de defectos una vez que el producto se ha enviado al cliente. Suponga que no ha habido inspecciones, pero que los programadores han sido muy cuidadosos y solamente se ha escapado un defecto por 1.000 líneas de código (algo significativamente mejor que la media en industrial) en el producto enviado. Eso significaría que se tendrían que corregir todavía 200 defectos en la casa del cliente o después de la entrega. A un coste estimado de 25.000 dólares por reparación de campo, el coste será de 5 millones de dólares, o aproximadamente 18 veces más caro que el coste total del esfuerzo de prevención de defectos.”

Es verdad que IBM produce software utilizado por cientos de miles de clientes y que sus costos por reparación en casa del cliente o después de la entrega pueden ser más altos que los de organizaciones de software que construyen sistemas personalizados, o para este caso especifico, mucho más elevado que las organizaciones gubernamentales que desarrollan software a la medida para mejorar su gestión diaria. Esto, de ninguna manera, niega los resultados señalados anteriormente. Aunque la organización media de software tiene costos de reparación después de la entrega que son muchas veces muy inferiores a de los de IBM (¡la mayoría no tienen ni idea de cuales son sus costos!), se están imponiendo ahorros en el costo asociados con actividades de garantía y control de calidad.

2.6 Modelo de calidad IDEAL

La siguiente sección tiene como finalidad dar un breve bosquejo al lector acerca del modelo de calidad IDEAL[9].

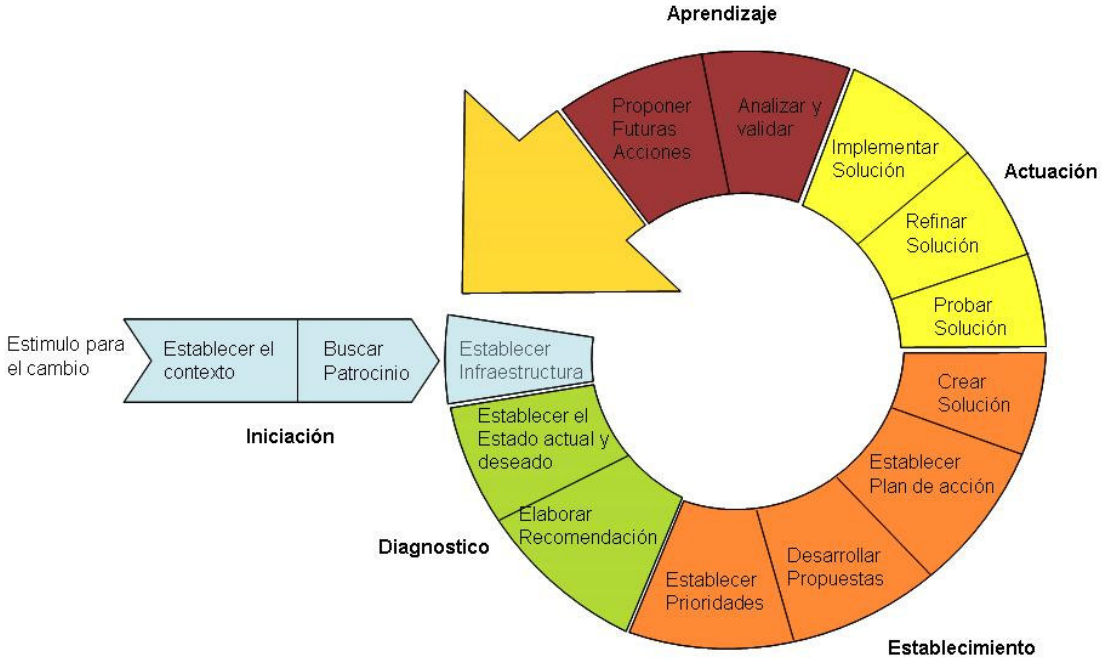
Esta metodología fue desarrollada en 1995, y entre los puntos más resaltantes que tiene, es que cuenta con la ventaja de tomar los puntos fuertes de las metodologías antecesoras. Originalmente fue concebido como un modelo de ciclo de vida para la mejora de los proceso del software basado en el modelo CMM y por esta razón el modelo utilizo términos de la mejora continua de procesos. Reconociendo que el modelo tenía grandes potenciales fuera del marco CMM, el SEI ha revisado el modelo IDEAL para ampliar su utilización. La nueva versión del modelo ha sido diseñada como la versión 1.1 para enfatizar que es el primer paso hacia la ampliación de su aplicabilidad (Gremba, J., Myers, C., 2005).

Esta es considerada una metodología de mejoramiento continuo de la calidad de software (SPI, Software process improvement) y divide el trabajo en 5 grandes procesos o fases, y cada una de esas fases esta conformada por algunas actividades.

Este es un modelo de mejoramiento continuo, lo que da a entender que las actividades se ejecutan de manera cíclica y lo que busca es que en cada iteración el proceso mejore. Veamos gráficamente lo dicho anteriormente.

[9] El método IDEAL debe su nombre debido a que esta constituido por 5 grandes fases llamadas: **I**nitiating Phase, **D**iagnosing Phase, **E**stablishing Phase, **A**cting Phase, **L**earning Phase.

Gráfico 3 Modelo I.D.E.A.L.



Fuente: Carnegie Mellon Software Engineering Institute (http://www.sei.cmu.edu/ideal/)

A continuación se explicará brevemente cada una.

2.6.1 Fase de Inicio

La fase de inicio del modelo IDEAL es el punto de partida. Es aquí donde la estructura para el mejoramiento es establecida, es decir, se establecen los roles y las responsabilidades y se asignan los recursos iniciales. Es en esta fase donde se crea un plan para el mejoramiento del software que guiará a la organización a través de esta fase, a través de la fase de diagnostico y a través de la fase de desarrollo.

Un trabajo base es completado en esta fase. Las razones de la organización de emprender un esfuerzo de mejoramiento son claramente articuladas. Las metas, objetivos, contribuciones y esfuerzos del negocio son identificados, así como las relaciones laborales dentro del organismo. La ayuda de los gerentes críticos es asegurada, así como la asignación de recursos en un orden de magnitud. Finalmente una infraestructura para manejar la implementación es creada.

Es en esta fase donde se construyen los equipos de trabajo. Típicamente se definen dos grupos que son claves durante la ejecución de este modelo, el equipo de manejo de la gerencia (MSG, management steering group) y el equipo de procesos de ingeniería del software (SEPG, software engineering process group). También durante esta fase, el plan de comunicación que se utilizará en todo el proyecto es establecido.

Esta fase es equivalente a la fase de inicialización que establece el PMI, donde se distribuyen las responsabilidades, se asignan los roles y se establecen los compromisos.

2.6.1.1 Estimulo para el cambio

Es importante reconocer las razones del negocio de cambiar las prácticas de una organización, para luego sacarle provecho. El estímulo para el cambio podría ser acontecimientos o circunstancias inesperadas, una orden de alguien en una posición alta en la jerarquía de la organización, o la información generada de las actividades de benchmarking como parte de un acercamiento continuo de la mejora. Cualquiera que fuese el estímulo, este puede tener influencia de gran envergadura dentro de las actividades en la organización. En general, cuando las razones del negocio del cambio son más evidentes, es más aceptada la idea dentro de la organización y hay mayores ocasiones para el éxito, es allí donde radica su importancia.

2.6.1.2 Establecer el contexto

Una vez que el estimulo ha sido identificado, la gerencia de la organización debe fijar el contexto donde se ejecutará el trabajo. “Establecer el contexto” significa estar claro en donde se ejecutará el esfuerzo dentro de la organización para que la estrategia aplicada surja efecto. ¿Qué metas y objetivos específicos del negocio serán observados o apoyados por este cambio? ¿Cómo afectará otras iniciativas y el trabajo en curso? ¿Qué beneficio se obtendrá (por ejemplo retorno de la inversión o que capacidades mejoradas)? El contexto y las implicaciones llegan a ser a menudo más evidentes mientras que trabajo avanza, pero es importante estar claros tanto como sea posible con respecto al contexto que se debe atacar para lograr una mejora en la calidad.

2.6.1.3 Buscar Patrocinio

Un patrocinio eficaz es uno de los factores más importantes para lograr el mejoramiento continuo. Es necesario mantener buenos niveles de patrocinio a través de todo el proceso, sobre todo debido a la incertidumbre y caos que tiene que hacer frente la organización al inicio de todo el proceso. El otorgamiento de recurso por parte de los niveles de patrocinio, es también un punto bien importante al momento de iniciar el esfuerzo. Los patrocinantes pueden ser más efectivos si dan atención personal al esfuerzo y se unen al esfuerzo en sus momentos más difíciles.

2.6.1.4 Establecer infraestructura

Una vez la razón del cambio y el contexto se establecen y los patrocinadores dominantes están confiados en el esfuerzo, la organización debe instalar un mecanismo para manejar los detalles de la puesta en práctica del esfuerzo de mejoramiento. La infraestructura puede ser temporal o permanente, y su tamaño y complejidad puede variar substancialmente dependiendo de la naturaleza de la mejora. Para un esfuerzo pequeño, la infraestructura puede ser solo un empleado por horas; para un esfuerzo grande y complejo, tal como mejora de proceso del software, puede implicar a unos 2-3% de la gente de la organización divididos en un número de grupos. Establecer la infraestructura implica el desarrollar de los acuerdos escritos de manera explícita que documentan y clarifican expectativas y describen responsabilidades.

Las actividades de la fase que inicia son críticas. Si se hacen bien, las actividades subsecuentes pueden ejecutarse con la interrupción y cambios mínimos. Si se hacen mal o incompleto, entonces el tiempo, el esfuerzo, y los recursos serán perdidos en fases subsecuentes.

2.6.2 Fase de Diagnóstico

En esta fase del método, la organización que aplica dicha metodología, empieza su proceso continuo de mejoramiento en el desarrollo del software. Esta fase coloca las bases para las futuras fases, y empieza a ejecutar, de acuerdo a la visión, objetivos de la organización, lecciones aprendidas de proyectos pasados, lo que se acordó en la fase inicial.

En esta fase se ejecuta una serie de actividades de diagnostico o valoración para establecer una línea base del estado actual de la organización o empresa. Los resultados y recomendaciones producto de este diagnostico serán incluidos dentro de las actividades o plan para el mejoramiento del software si así lo amerita, es decir, se completa el plan de mejoramiento continuo del software.

2.6.2.1 Establecer el estado actual y el deseado

Establecer el estado actual y el deseado es similar a identificar el origen y destino de un viaje. Establecer estos dos estados pude ser hecho de manera más fácil utilizando como referencia estándares reconocidos como el CMM para software. Donde no existan estándares o estos no estén disponibles, un buen punto de partida son los factores identificados en la actividad de identificación de “estímulos al cambio”. Esta actividad debe concentrarse en los elementos críticos de los cambios que son introducidos y no en cada aspecto del trabajo de la organización.

2.6.2.2 Elaborar recomendaciones

Las recomendaciones que son desarrolladas como parte de esta actividad, sugieren una manera de proceder en las actividades subsecuentes. Las actividades de diagnostico, a menudo son llevadas a cabo por grupo experto en la materia. Sus recomendaciones son frecuentemente tomadas en cuenta por los encargados y lo patrocinantes del proceso.

2.6.3 Fase de Establecimiento

El propósito general de esta fase es establecer un plan de trabajo detallado. Se fijan las prioridades que reflejen las recomendaciones hechas en la fase de diagnostico

Durante esta fase es donde se desagregan las metas más generales establecidas en las fases previas y se establece una manera de medir su progreso. Las distintas métricas son establecidas aquí para las distintas metas y los recursos necesarios para

lograrlo son asignados a los equipos técnicos de trabajo, es decir, las acciones, los productos entregables y las responsabilidades son agregados al plan de acción.

2.6.3.1 Establecer prioridades

La primera actividad de esta fase es fijar las prioridades para el esfuerzo del cambio. Estas prioridades deben tomar varios factores en cuenta: los recursos son limitados, las dependencias existentes entre las actividades recomendadas, los factores externos pueden intervenir, y las prioridades más globales de la organización deben ser tomadas en cuenta.

2.6.3.2 Desarrollar propuesta

Combinar la comprensión creciente del alcance del trabajo (elaborado en la fase que diagnostico) con un sistema de prioridades conduce al desarrollo de una estrategia para lograr el trabajo e identificar disponibilidad del recurso. Los factores técnicos pudieran incluir los requerimientos de instalar una nueva tecnología y las nuevas habilidades y conocimiento requeridos para usar esa tecnología. Los factores no técnicos, incluyendo la cultura de la organización, fuentes de resistencia al cambio, niveles del patrocinio, e incluso el mercado, también deben ser considerados.

2.6.3.3 Elaborar un plan de acción

Con la propuesta desarrollada, un plan detallado de la puesta en práctica puede ser desarrollado. Este plan incluye el horario, tareas, hitos, puntos de decisión, recursos, responsabilidades, métricas, mecanismos de seguimiento y medición, riesgos y las estrategias de mitigación, y cualquier otro elemento requerido por la organización.

2.6.4 Fase de Actuación

En esta fase es donde se ejecuta todo lo anteriormente planeado, es decir, es donde se atacan las áreas que nos ayudarán a la mejora de la calidad del desarrollo de software, se ejecuta todo lo planeado y conceptualizado en las fases anteriores. Esta es la fase que más consume tiempo y recursos en todo el proceso.

Una vez ejecutados los pilotos de prueba para el mejoramiento de software y medidos, según la métrica que se ha establecido, se difunde a través de toda la

organización para que el desarrollo y las aplicaciones de las soluciones de lo anteriormente probado sean de uso común y contribuya a alcanzar los objetivos.

2.6.4.1 Crear solución

Esta fase empieza con todos los elementos claves disponibles para crear la mejor solución que ayude a solventar las necesidades previamente detectadas, de la organización. Estos elementos claves pueden incluir herramientas existentes, procesos, conocimiento, y habilidades del personal, así como nuevos conocimientos requeridos, y la ayuda de un ente externo. La solución, que puede ser absolutamente compleja y multi-facetica, es creada a menudo por un grupo de trabajo netamente técnico.

2.6.4.2 Probar solución

Una vez que se haya creado una solución, debe ser probada, pues rara vez las soluciones trabajan exactamente tal y como se a previsto. Esto se logra a menudo a través de una prueba experimental o prueba piloto, pero cualquier otro medio puede ser utilizado.

2.6.4.3 Refinar solución

Una vez que se haya probado la solución, debe ser modificada para reflejar el conocimiento, la experiencia, y las lecciones aprendidas que fueron ganadas de la prueba. Varias iteraciones del proceso de prueba-refinación pueden ser necesarias para alcanzar una solución satisfactoria. Una solución debe ser realizable antes de que se ponga en practica, pero esperar una solución "perfecta" puede retrasa innecesariamente la ejecución del plan de mejora.

2.6.4.4 Implementar solución

Una vez que la solución se a refinado y es realizable, puede ser puesta en en practica en toda la organización. Existen varias maneras para llevar a cabo esto, incluyendo la aplicación de la solución de arriba hacia abajo (comenzando en el nivel más alto de la organización y bajando paulatinamente) o mediante la filosofia de just-in-time o justo a tiempo (poniendo en practica la solución proyecto por proyecto en el momento apropiado dentro del ciclo de vida de este). No hay una manera única de aplicar el conocimiento; la manera de implementar la solución se debe elegir basándose en la

naturaleza de la mejora y de las circunstancias de la organización. Para un cambio importante, la puesta en práctica puede requerir tiempo y recursos substanciales.

2.6.5 Fase de Aprendizaje

Esta fase completa el ciclo de mejoramiento. Uno de los objetivos del modelo IDEAL es estar continuamente mejorando la habilidad para implantar los cambios en el desarrollo de software. El objetivo de esta es hacer que la próxima iteración o aplicación de este método sea más efectivo. Cuando se ejecuta esta fase, las soluciones han sido desarrolladas y aplicadas, las métricas han sido establecidas, y se evalúa si las metas han sido logradas. Luego se hace esta información se almacena en una base de datos que servirá como fuente de documentación para los equipos que ejecuten procesos de mejoras en lo sucesivo con este método.

Ejecutando esta fase correctamente y ajustando las estrategias, la siguiente ejecución será aún más precisa y efectiva en lograr el objetivo que no es otro que la mejora de la calidad del software.

2.6.5.1 Analizar y validar

Esta actividad contesta a varias preguntas: ¿De qué maneras el esfuerzo logró su propósito previsto? ¿Qué salió bien? ¿Qué se podía hacer más con eficacia o eficiencia? Se recogen, se analizan, se resumen, y se documentan las lecciones. Las necesidades del negocio identificadas durante la fase inicial se reexaminan para considerar si se han resuelto.

2.6.5.2 Proponer futuras acciones

Durante esta actividad, las recomendaciones basadas en análisis y las validaciones se desarrollan y se documentan. Las ofertas para mejorar las puestas en práctica futuras del cambio se proporcionan a los niveles apropiados de la gerencia para su consideración.

Como se puede ver, este método es un ciclo continuo de aprendizaje que asegurará la mejora de la calidad del software, aplicando varias iteraciones de este método, es por ello que se afirma que el método IDEAL ayuda a la mejora continua de la calidad del software.

No hace falta un análisis profundo para notar la similitud que tiene esta metodología con la metodología aplicada por el PMI para la ejecución de un proyecto, y es que en efecto, la mejora continua de la calidad del software es un proyecto que debemos administrar de manera eficiente para asegurar su éxito y cada vez que se ejecute un proyecto, debemos documentar lo mejor posible las lecciones aprendidas para que sirva esto como una fuente de información para los sucesivos proyectos y no cometer los mismos errores, es decir, se debe lograr con cada ejecución del método IDEAL mejorar la calidad, de manera más precisa y efectiva, en el desarrollo de software.

La explosión del desarrollo tecnológico ha conducido a las organizaciones a adoptar nuevas tecnologías en una tasa de aumento. Así mismo la complejidad de estas tecnologías está en aumento, y el método IDEAL puede ayudar a las organizaciones a adoptar dichas mejoras.

Marco referencial

Al área de tecnología de información del Estado Venezolano, desde sus inicios, nunca se le dio la importancia que debió tener. Revisando la historia del desarrollo científico del país se nota que para el año de 1.959, se crea el Instituto Venezolano de Investigaciones Científicas (IVIC), adscrito al Ministerio de Sanidad y Asistencia Social de la época, y solo se desarrolla en las áreas de biología, matemática, física, química y medicina. Para el año de 1.972 se creó el Centro Tecnológico del IVIC con el objeto de llevar a cabo desarrollos tecnológicos en diversas ramas de las ingenierías y servir de base para la prestación de servicios. La mayor parte del grupo de ingeniería se separó del IVIC en 1982 para formar la Fundación Instituto de Ingeniería, donde se puede encontrar en una de sus ramas de investigación, el desarrollo del área de tecnología de información para el estado venezolano.

Pero es en los últimos tiempos que se han visto los esfuerzos en el sector público por usar los beneficios de la tecnología de información para el mejoramiento de la gestión diaria de los organismos del gobierno. Dichos esfuerzos se ven reflejados en la divulgación de las distintas leyes del estado venezolano donde se obliga al uso de dicha tecnología y al mejoramiento constante de la misma, así como también en el incremento en la inversión en esta área, viéndose esto reflejado en el presupuesto otorgado al Ministerio de Ciencias y Tecnología para el año 2.004 el cual ascendió a Bs. 272.040.390.800,00 y para el año 2.005 fue de Bs. 539.551.853.523 ,00, todo esto con el único fin de mejorar el servicio que se le presta al ciudadano común, siendo un ejemplo emblemático, entre otros, la modernización del poder judicial y el desarrollo del sistema SIGECOF para el control presupuestario de los distintos organismos públicos.

Este largo despertar por parte del Estado, hacia la manera de utilizar los sistemas de información, ha provocado que la tecnología actual en muchos de los casos sea obsoleta y otras veces, esta ausente dentro de los organismos públicos.

Al decir que la tecnología es obsoleta, nos referimos a que en la mayoría de los organismos donde se utilizan sistemas de información implantados, es un denominador común que dichos sistemas en un momento dado fueron tecnología de punta, pero no se tuvo la preocupación por actualizar dicha plataforma.

En este punto cabe destacar que en esta área no existe mucha investigación que ayude a impulsar este recurso y a explotarlo como es debido.

Como se cito anteriormente, es ahora, cuando se le esta dando mayor peso a los sistema automatizados de información para la gestión diaria de las entidades gubernamentales, y es por ello que se hace necesario desarrollar la investigación en el área de control de calidad en el desarrollo de software para que los sistemas que están naciendo o por nacer, tengan un control mínimo de calidad que asegure su funcionalidad y la inversión que se esta ejecutando en el desarrollo de dichos sistemas.

Marco metodológico

A través de este capitulo se demuestra al lector los pasos a seguir para desarrollar el presente trabajo, y para demostrar que se está trabajando de manera ordenada y con un basamento científico. Es importante tener claro con que metodología de investigación se va a trabajar, sobro todo para saber contra que otro tipo de trabajo puede ser comparado los resultados generados por éste.

3.1 Tipo de investigación

Dado que el objetivo general de la presente investigación fue "Diseñar, en base a metodologías aceptadas, una metodología para la gestión de la calidad para el desarrollo de sistemas automatizados de información en la administración pública venezolana", ésta se enmarcó en la modalidad de proyecto factible en fase de elaboración de propuesta apoyada en una investigación de tipo documental, ya que el investigador, al proponer una metodología para el control de la calidad en el desarrollo de sistemas de información automatizados, busca mejorar la calidad de los desarrollos de los sistemas de información, y de esa manera apoyar la tendencia del gobiernos en cuanto al desarrollos de sistemas de información automatizados bajo la filosofía del software libre y con una buena calidad.

Ello se sustenta en el concepto de proyecto factible, que según el Manual de Trabajos de Grado de Especialización, Maestrías y Tesis Doctorales de la UPEL (1998):

> Consiste en la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viable para solucionar problemas, requerimientos o necesidades de organizaciones o grupos sociales, puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos. El proyecto debe tener apoyo en una investigación de tipo documental, de campo o ambas modalidades (p. 7)

3.2 Población, muestra y unidad de análisis

La unidad de análisis con que se va a trabajar en la presente investigación es la calidad durante el desarrollo de sistemas de información, pues es la característica principal a estudiar, o la característica que será sometida a análisis.

La población del presente trabajo son las metodologías de control de calidad en el desarrollo de software existentes en el mercado, y la muestra de dicha población son 3 metodologías seleccionadas por estar entre las más utilizadas en la actualidad. Dichas metodologías son: metodología IDEAL, metodología SQUID, y metodología de calidad ISO 9001.

3.3 Variables: definición conceptual y operacional

A esta altura en el desarrollo de la presente investigación, se muestra las variables manejadas en este proyecto y la manera de operacionalizarlas, para hacerla más fácil de medir y así lograr los objetivo planteados. Para esta ocasión se relacionan las fases del proyecto con las fases planteadas del desarrollo de software. De esta manera lograr los objetivos específicos planteados al inicio, es mucho más fácil

Tabla 1 Operacionalización de las variables manejadas en el proyecto

Fases del proyecto	Fases de Desarrollo de software	
Inicio	Visualización	Objetivos Entradas Procesos Salidas Lista de verificación Métricas
	Conceptualización	Objetivos Entradas Procesos Salidas Lista de verificación Métricas
Planificación	Diseño	Objetivos Entradas Procesos Salidas Lista de verificación Métricas
Ejecución	Desarrollo	Objetivos Entradas Procesos Salidas Lista de verificación Métricas
Cierre	Entrega	Objetivos Entradas Procesos Salidas Lista de verificación Métricas

Fuente: Diseño del investigador

3.3.1 Definiciones conceptuales

- Objetivos: descripción del propósito del módulo o fase, para poder cotejar contra lo realizado y poder medir el progreso del mismo.
- Entradas: documentos y/o información necesaria para completar el módulo. El nombre de los documentos van a ser genéricos, sin embargo estos nombres deben ser adaptado según el criterio del equipo de aseguramiento de calidad que este implementando la metodología.
- Procesos: descripción de las tareas y acciones que deben ejecutar el o los integrantes del equipo de aseguramiento de calidad para dar por cumplido el módulo.
- Salidas: productos que deben ser generados al final del módulo y que servirán en la mayoría de los casos de insumo (entrada) para los subsiguientes módulos o fases.
- Lista de verificación: lista que utilizan los integrantes del equipo de aseguramiento de calidad para verificar que se ha cumplido el módulo o fase correctamente.
- Métricas: Mediciones a realizar durante la ejecución del módulo. Estas métricas se archivarán convenientemente de forma tal que permitan evaluar el avance y mejora de la calidad del proyecto en curso. Asimismo se utilizarán como base para estimaciones y comparaciones para proyectos futuros.

La Propuesta

Una vez finalizada la investigación del presente trabajo que se desarrolla bajo un proyecto factible de tipo documental, se procedió al procesamiento de cada una de las metodologías anteriormente expuestas y al desarrollo de una "Metodología para el Desarrollo de Sistemas Automatizados de Información en las Organizaciones del Estado Venezolano" para de esta manera dar respuesta a cada uno de los objetivos específicos planteados en el capítulo del Problema.

4.1 Objetivo

Con el desarrollo de esta metodología se busca mejorar, los costos y tiempo de respuesta en el desarrollo de software dentro del marco Gubernamental Venezolano, pero a su vez se quiere lograr una mejora en la satisfacción de los usuarios de estos productos, mejorando la calidad y garantizando que se cumplan los requerimientos establecidos.

4.2 Justificación

Como parte de la modernización del estado, que lleva el gobierno nacional, esta el desarrollo de sistemas de información automatizados. Para esto, el Ministerio de Ciencia y Tecnología, como órgano competente para dar lineamientos sobre el desarrollo de dichos sistemas, ha creado un plan nacional de migración a software libre de la administración pública nacional.

El desarrollo de la presente metodología se justifica, ya que ayudará a crear sistemas de información automatizados que se acerquen a las normas de calidad ISO:9126 y por lo tanto, ayudará a la aplicación de dicho plan, que se ajusta al decreto 3.390, donde se obliga a todos los organismos a nivel nacional a desarrollar sistemas de información automatizados bajo la filosofía de Software Libre y con una calidad que se adapte a las normas ISO 14598.

4.3 Estructura

4.3.1 Fase de análisis

La fase de análisis esta divida en dos sub-fases; visualización y conceptualización.

Se definiran objetivos, entradas, procesos, salidas, lista de reverificación y posibles métricas a aplicar en esta fase del desarrollo del sistema.

4.3.1.1 Subfase de Visualización

4.3.1.1.1 Objetivos

Los objetivos de esta subfase, son:

- Determinar que con que recursos se cuenta para producir el software. Los recursos pueden ser, recursos humanos y recursos tecnológicos.
- Tener un primer acercamiento del tiempo requerido para producir dicho software. Este objetivo, puede ser alcanzado a través de entrevistas con los usuarios que tienen la necesidad.
- Determinar si los requerimientos del usuario, son realmente lo que el mismo necesita.
- Separar los requerimientos, entre obligatorios o indispensables y opcionales.
- Verificar que cada requerimiento este
    - debidamente documento, con una correcta y clara definición.
    - dentro del alcance del sistema

4.3.1.1.2 Entradas

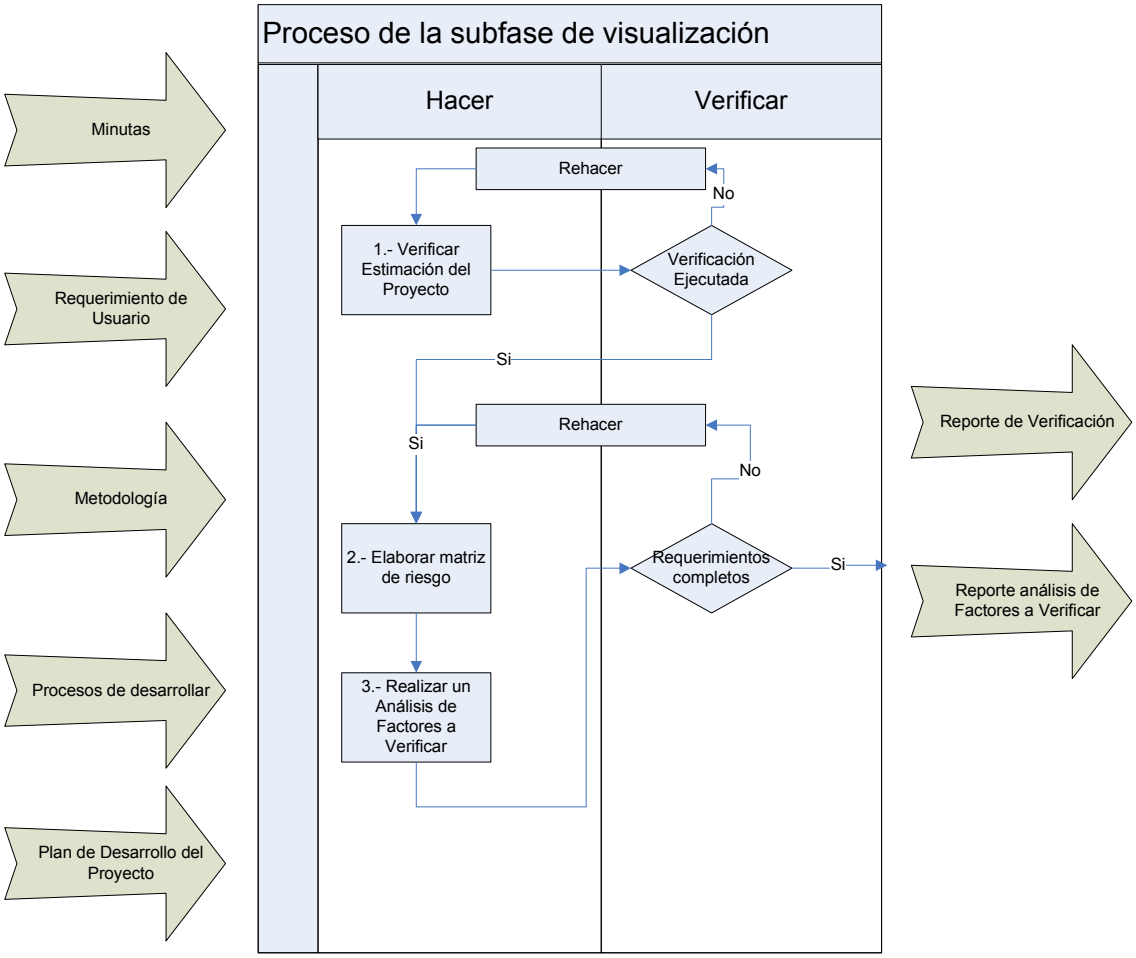
Las entradas para esta subfase son:

- Cada una de las minutas de las reuniones con los futuros usuario.
- Documento de requerimiento del usuario.
- Metodología utilizada para desarrollar el proyecto.
- Descripción de los procesos de desarrollo a utilizar en el proyecto.

En este punto es conveniente señalar, que el documento de requerimiento del usuario es importante que sea desarrollado lo más explícitamente por parte de los usuarios que requieren el sistema de información, pues será piedra angular para tomar futuras decisiones.

4.3.1.1.3 Procesos

Gráfico 4 Proceso del módulo de análisis



Fuente: el autor

4.3.1.1.3.1 Verificar estimación del proyecto

Muchos proyectos de software son esencialmente innovadores, pero una ineficaz estimación del sistema, puede llevar a un exceso de costos. Estimar costos de software es un proceso complicado porque el desarrollo del proyecto es influenciado por un largo número de variables, muchas de ellas son subjetivas, no cuantificables e interrelacionadas en forma compleja.

Una estimación inapropiada de costos puede dañar más a la calidad del proyecto de software que a cualquier otro factor. Si la estimación es incorrecta, el equipo de proyecto hará lo imposible para coincidir con la estimación. Todo esto provoca un aumento de esfuerzo de desarrollo, insatisfacción del cliente, esfuerzo adicional en el área del cliente para compensar la debilidad del sistema, y desanimar al personal del proyecto. La verificación puede aumentar la validez de la estimación. La estimación del software es un proceso de tres partes como se describe a continuación:

- Validar la sensatez del modelo de estimación.
- Validar que el modelo incluya todos los factores necesarios.
- Verificar la efectividad del modelo estimado de costo.

4.3.1.1.3.2 Elaborar matriz de riesgo

La Matriz de Riesgo es una herramienta diseñada para evaluar el control en los sistemas de información.

Uno de los mayores beneficios de la matriz de riesgo es, no sólo la identificación de los mismos, sino también qué es lo que el sistema debe hacer para ó con cada uno de ellos. En principio, la matriz de riesgo es una herramienta de diseño, pero puede ser usada como herramienta de verificación.

En forma ideal la matriz de riesgo comienza en la fase de requerimientos, se desarrolla en la fase de análisis y se termina en la fase de diseño.

4.3.1.1.3.3 Realizar un análisis de factores a verificar

Debe llevarse a cabo un proceso para estimar las incumbencias (“concerns”) asociadas a la fase de requerimientos del desarrollo del sistema. Debe incluirse un programa de verificación para cada factor a considerar (son 15), cubriendo cada fase del proceso de desarrollo.

El programa de verificación enumera aquellos criterios, que aseguran al equipo de aseguramiento de la calidad, que la magnitud de esa preocupación es mínima. A estos criterios debe responder el equipo de aseguramiento de la calidad. También se debe

realizar una verificación suficiente para evaluar si el equipo de proyecto ha manejado adecuadamente cada criterio de verificación.

Los factores a verificar en la etapa de requerimientos son:

- Requerimientos compatibles con la metodología (Factor de prueba: Metodología)
- Funcionalidad de las especificaciones definidas (Factor de prueba: Precisión)
- Usabilidad de las especificaciones (Factor de prueba: Facilidad de uso)
- Mantenimiento de las especificaciones (Factor de prueba: Mantenibilidad)
- Necesidades de portabilidad (Factor de prueba: Portabilidad)
- Interfaces del sistema (Factor de prueba: Acoplamiento)
- Criterios de performance (Factor de prueba: Performance)
- Necesidades operativas (Factor de prueba: Facilidad de operación)
- Tolerancia (Factor de prueba: Fiabilidad)
- Reglas de autorización (Factor de prueba: Autorización)
- Requerimientos de integridad de archivos (Factor de prueba: Integridad de archivos)
- Recuperación ante fallas en los requerimientos (Factor de prueba: Auditoria)
- Impacto de fallas (Factor de prueba: Continuidad de procesamiento)
- Nivel de servicio deseado (Factor de prueba: Nivel de Servicio)
- Permisos y accesos (Factor de prueba: Seguridad)

4.3.1.1.4 Salidas

- Matriz de riesgo.
- Análisis de factores a verificar

4.3.1.1.5 Lista de verificación

Tabla 2 Lista de Verificación de la subfase visualización

		Respuesta			
número	Ítem	Si	No	No aplica	Observaciones
1	¿El equipo del proyecto ha realizado una buena verificación para determinar la validez de las estimaciones realizadas?				
2	¿El equipo del proyecto esta desacuerdo con la valides de las estimaciones realizadas?				
3	¿El equipo del proyecto posee un sistema de reportes razonable para informar el estado del mismo?				
4	¿Se determino cuales son los factores más importantes en la evaluación de la estimación realizada? (Ejemplo: tamaño del softwara, tiempo, etc)				
5	¿El proceso de desarrollo está totalmente documentado?				
6	¿La estimación efectuada es razonable como para completar el proyecto según lo especificado en el plan?				
7	¿Se definio un método o procedimiento para determinar e informar el estado del proyecto?				
8	¿Los requerimientos definidos son verificables?				
9	¿el o los usuarios estan de acuerdo con los requerimientos definidos?				
10	¿Los desarrolladores entiendes los requerimientos?				
11	Se identificaron los riesgos del proyecto?				

Fuente: Diseño del investigador

4.3.1.1.6 Métricas

Eficacia en la estimación del proyecto: Duración actual del proyecto / Duración estimada del proyecto

4.3.1.1.7 Matriz de comparación de resultados

Tabla 3 Matriz de interpretación de resultados para las métricas para la subfase de Visualización

Eficacia en la estimación del proyecto: con esta métrica se busca medir la efectividad a la hora de estimar la duración de un proyecto por parte del equipo que desarrolla el proyecto.		
Resultado	Interpretación	Responsable
= 0	El proyecto aún no ha empezado.	Gerente de proyecto.
< 1	El proyecto se encuentra dentro del tiempo de ejecución previsto.	
= 1	Si esta métrica es aplicada al finalizar el proyecto, la eficacia en la estimación es del 100%. De lo contrario, quiere decir, que el proyecto posiblemente finalizará con retrazo en la entrega. Por lo tanto el responsable debe tomar las medidas necesarias para su ajuste.	Gerente del proyecto.
> 1	La eficacia de la estimación no es buena, el proyecto esta retrazado. El responsable del proyecto debe tomar las medidas necesarias para reevaluar el alcance del proyecto o llegar a un acuerdo con el cliente final para una entrega tardía.	Gerente de proyecto, líder del proyecto.

Fuente: Diseño del investigador

4.3.1.2 Subfase de Conceptualización

4.3.1.2.1 Objetivos

Los objetivos de esta subfase son:

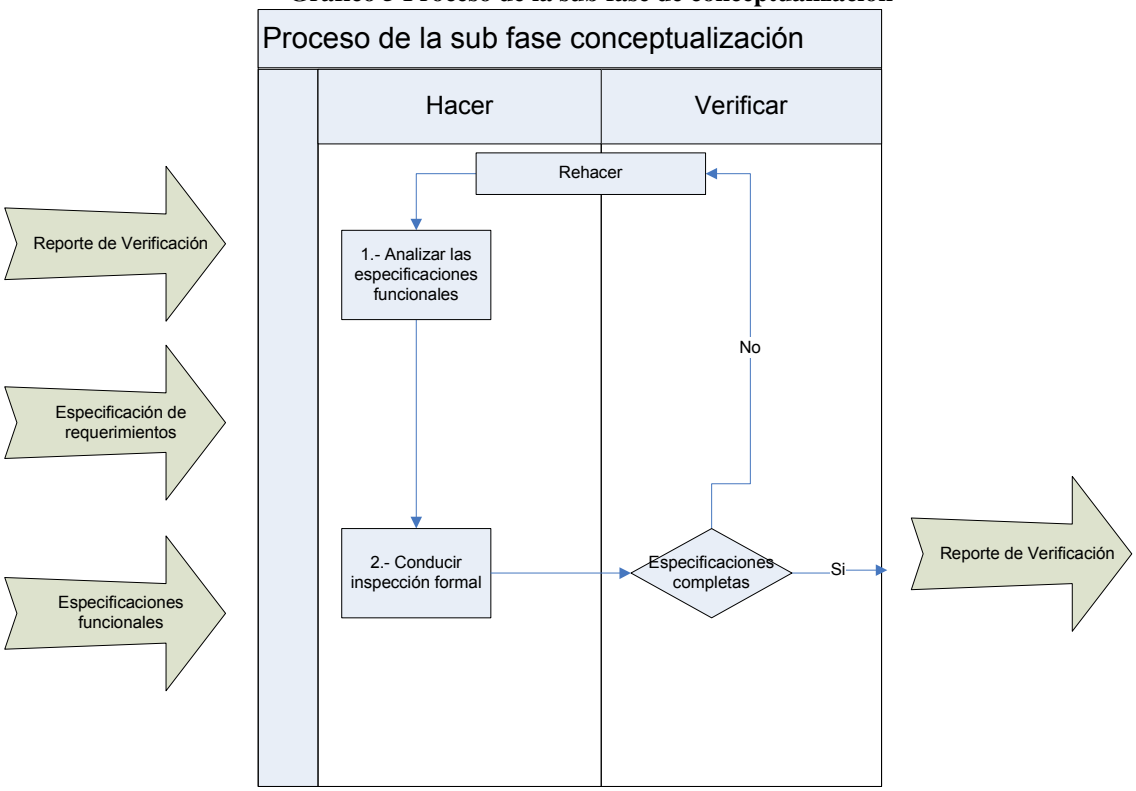
- Describir la funcionalidad del producto que va a ser entregado.
- Definir el ambiente en el que va a operar.
- Identificar las necesidades de soporte u operación, como hardware, software, etc.
- Definir para cada requerimiento, qué prueba se va a ejecutar para asegurar que se cumpla con el mismo.

4.3.1.2.2 Entradas

- Reporte de Verificación.
- Documento de especificación de requerimientos revisado.
- Documento de especificaciones funcionales

4.3.1.2.3 Procesos

Gráfico 5 Proceso de la sub fase de conceptualización



Fuente: el autor

4.3.1.2.3.1 Analizar la especificación funcional

La especificación funcional, debe ser vista como un proceso de representación, a fin de poder llegar a una exitosa implementación de software. Algunos principios para especificar, podrían ser:

- Separar requerimientos funcionales de implementación.
- Desarrollar un modelo del comportamiento deseado del sistema, que comprenda datos y la respuesta funcional del sistema a los estímulos del ambiente.
- Crear un modelo cognitivo en lugar de un diseño o un modelo de implementación. El modelo cognitivo describe el sistema como es percibido por los usuarios de la organización.

Se verificará que la especificación funcional cumpla, con al menos, los siguientes lineamientos:

- Los diversos formatos de diagramas a utilizar y otras formas de notación, deberán ser restringidas al mínimo en número y consistentes en su uso. La información confusa e inconsistente, sea tanto simbólica o gráfica, perjudica el entendimiento y propicia errores. Se recomienda el uso de diagramas de casos de uso, diagramas de secuencia, diagrama de modelo de objetos, como mínimo.
- La especificación funcional, debe ser verificable.

Se verificará que la especificación funcional cumpla, con al menos, el siguiente contenido:

- Resumen del Sistema.
- Entregables.
- Ambiente donde se ejecutará.
- Usuarios.
- Seguridad.
- Flujo de datos.
- Flujo de control.
- Narrativa de proceso general.
- Restricciones / Limitaciones.
- Clases de pruebas requeridas.
- Respuesta esperada del software.
- Consideraciones especiales.

4.3.1.2.3.2 Conducir una inspección formal

La revisión de una especificación funcional es conducida por el **desarrollador** y el **cliente**.

Dado que las especificaciones forman el punto fundamental y a partir del cual se realizará el diseño y subsecuentemente las actividades de la ingeniería del software, se deberán extremar los cuidados para conducir esta revisión. La revisión en una primera

instancia es conducida en un nivel macro. En este nivel, el equipo de trabajo intenta garantizar que la especificación esté completa, consistente y exacta.

Se tendrán en cuenta los siguientes lineamientos para una revisión detallada:

- Detectar y remplazar términos vagos.
- Cuando se especifiquen rangos (numéricos, etc.) asegurarse que no contengan valores asumidos sin ser definidos expresamente.
- Estar alerta de verbos y pronombres ambiguos.
- Identificar afirmaciones que no incluyan certeza.
- Cuando una estructura es descripta en palabras, asegurar que se realice un diagrama para una mejor comprensión.
- Cuando sea necesario realizar un cálculo especificado, asegurar que se presente al menos un ejemplo del mismo.

Con respecto a los cambios en los requerimientos, una vez que los mismos han sido finalizados, el usuario deberá notar que cada uno de ellos, posiblemente es una ampliación o modificación del alcance del software y por ende, un cambio en la especificación funcional. Esto modificará el tiempo establecido para su finalización.

Aún cuando se realicen las mejores revisiones, cierto número de problemas en la especificación persiste. La especificación es difícil de verificar, por ende las inconsistencias u omisiones pueden pasar desapercibidas. Durante la revisión, pueden ser recomendados más cambios en la especificación funcional. Puede ser extremadamente difícil estimar el impacto global del cambio, esto es, cómo un cambio en una función afecta a los requerimientos en otra función. Esto último deber ser tomado muy en cuenta a la hora de “impactar” dicho cambio.

4.3.1.2.4 Salidas

- Informe de la Inspección.
- Resumen de la Inspección

4.3.1.2.5 Lista de Verificación

Tabla 4 Lista de Verificación de la subfase de Conceptualización

		Respuesta			
número	Item	Si	No	No aplica	Observaciones
1	¿Los objetivos y metas del sistema se mantienen consistentes con las políticas de software de la organización?				
2	¿Son claros los diagramas? ¿Pueden ser explícitos sin necesidad de ser descriptos o narrados?				
3	¿Las funciones principales del sistema, están dentro del alcance? ¿Cada una de ellas ha sido adecuadamente definida?				
4	¿Ha sido considerado el riesgo tecnológico del desarrollo?				
5	¿El proceso de desarrollo está totalmente documentado?				
6	¿Se han considerado requerimientos alternativos de software?				
7	¿El usuario final ha estado involucrado?				
8	¿El usuario revisó el prototipo y/o manual del usuario?				

Fuente: Diseño del investigador

4.3.1.2.6 Métricas

Tasa de efectividad de verificación: $\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$

4.3.1.2.7 Matriz de comparación de resultados para la métrica

Tabla 5 Matriz de interpretación de resultados para las métricas para la subfase de Conceptualización

Tasa de efectividad de verificación: Esta métrica indicará que tan avanzado se encuentra el proyecto en un momento dado y que tan efectiva ha sido la aplicación de la lista.		
Resultado	Interpretación	Responsable
= 0	El proyecto esta en sus inicios y no se ha ejecutado la fase.	
< 1	Algunos ítems no fueron cubiertos. Puede ser por varios motivos. Es conveniente revisar de nuevo los ítems para futuras ocasiones. El responsable y el líder del proyecto deben depurar y corregir la lista para futuras ocasiones.	El gerente de proyecto debe revisar la lista.
= 1	Todos los ítems han sido cubiertos. Es el resultado ideal. La fase se ha ejecutado correctamente.	

Fuente: Diseño del investigador

4.3.2 Fase de Diseño

4.3.2.1 Objetivos

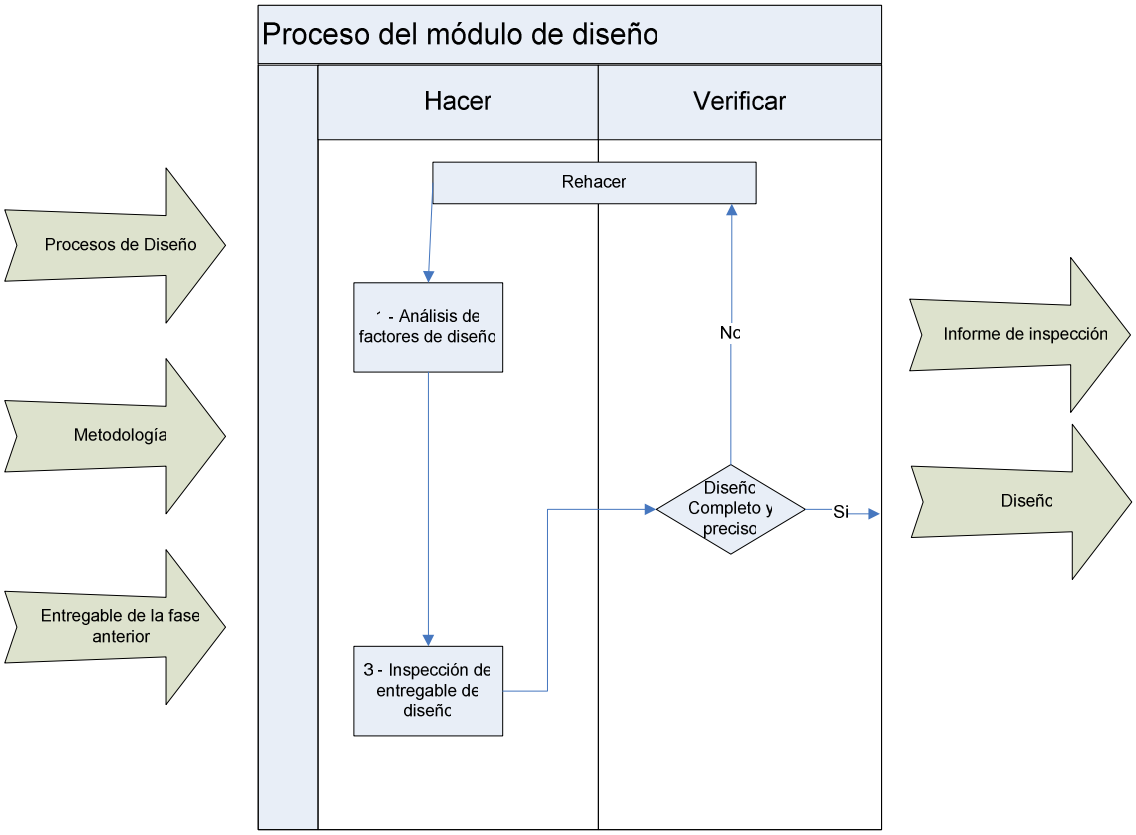
- Establecer los factores que conducen a un diseño correcto y preciso.
- Conducir revisiones de diseño.
- Conducir inspecciones de los entregables del diseño

4.3.2.2 Entradas

- Comprensión del diseño por parte del equipo de trabajo.
- Especificaciones de entrada
- Especificaciones de procesamiento
- Especificaciones de salida
- Diagramas de flujo del sistema.
- Requerimientos de software y hardware.
- Política de resguardo de la información.
- Documento de diseño de alto nivel (Lógico).
- Documento de diseño de detalle (Físico y lógico).

4.3.2.3 Procesos

Gráfico 6 Procesos del módulo de diseño



Fuente: el autor

4.3.2.3.1 Análisis de Factores

El o los encargados de dirigir la verificación podrán seleccionar los factores de interés apropiados, para ajustar la verificación, teniendo en cuenta los siguientes objetivos generales para la fase de diseño:

- Desarrollar una solución al problema de negocio planteado.
- Determinar el rol de la tecnología para resolver el problema.
- Desarrollar especificaciones para los segmentos manuales si los hubiese, y automatizados del sistema.
- Cumplir con el plan de acción, procedimientos, estándares y regulaciones.
- Completar el proyecto dentro de las restricciones del personal y cronograma establecidos.

Los factores a verificar en la Fase de Diseño son:

- Controles de integridad de datos en base de datos.
- Reglas de autorización y manejo de datos.
- Procedimientos de acceso a los datos.
- Diseño acorde con la metodología establecida.
- Diseño acorde con los requerimientos establecidos.
- Facilidad de uso.
- Interfaces de diseño.
- Diseño acorde con criterios establecidos.

Evidentemente este proceso, originará cambios en el diseño de la aplicación, en aras de la aplicación de la metodología a seguir, y de generar un mejor producto, por lo que se hace necesario llevar a cabo una nueva revisión o inspección de los entregables de esta fase.

4.3.2.3.2 Conducción de la Inspección de los Entregables de Diseño

La Inspección es un proceso por el cual el diseño nuevamente es evaluado de manera genérica para saber si los cambios especificados fueron implementados correctamente.

Para lograr esto, el equipo de proyecto examinará el diseño antes de las modificaciones, las especificaciones de cambio, y el diseño luego del cambio para determinar los resultados. Buscan tres tipos de defectos: Incorrectos (el cambio no se ha hecho correctamente), faltantes (algo que debería haberse cambiado, no se cambió), y adicionales (algo fue cambiado o agregado sin la intención de hacerlo).

4.3.2.4 Salidas

- Informe de inspección
- Diseño

4.3.2.5 Lista de Verificación

Tabla 6 Lista de verificación de la fase de diseño

		Respuesta			
número	Ítem	Si	No	No aplica	Observaciones
1	¿Se han recibido todos los entregables necesarios para llevar a cabo la elaboración y verificación del diseño?				
2	¿Los usuarios están de acuerdo con el diseño realizado?				
3	¿El equipo de proyecto esta de acuerdo con el diseño elaborado?				
4	¿La gerencia apoya el uso de la metodología de creación de diseño?				
5	¿El proceso de revisión de diseño fue conducido en un momento?				
6	¿Está de acuerdo el equipo de trabajo que los ítems o puntos identificados, necesitan ser modificados?				
7	¿Se ha provisto de tiempo suficiente para la inspección del diseño?				
8	¿Han sido planificados los procesos de inspección?				
9	¿Se ha elaborado una lista de defectos encontrados?				
10	¿Estuvo de acuerdo todo el equipo de trabajo con la lista elaborada?				
11	¿Ha participado el usuario final en la identificación de posibles problemas con el diseño elaborado?				
12	¿Cumple el diseño con las especificaciones técnicas de la organización?				
13	¿Cumple el diseño con las normas de desarrollo de software preexistentes en la organización?				

Fuente: Diseño del investigador

4.3.2.6 Métrica

- Estabilidad del diseño: Cantidad de cambios introducidos en el diseño / Cantidad total de cambios en el proyecto
- Eficiencia en la detección de defectos: Cantidad de defectos detectados hasta el diseño / Cantidad total de defectos hasta el diseño

4.3.2.7 Matriz de comparación de resultados para la métrica

Tabla 7 Matriz de interpretación de resultados para las métricas para la fase de Diseño

Estabilidad del diseño: Esta métrica determinará que tan efectivo es el diseño propuesto.		
Resultado	Interpretación	Responsable
< 1	Mientras el valor se aproxime a cero, quiere decir que hay menos cambios en esta fase, lo cual es excelente, ya que sugiere que el diseño es el correcto. Por el contrario, si el valor se acerca a uno, quiero decir que la mayor cantidad de cambios se efectuaron en esta fase y no en fases anteriores lo cual sugiere que la estabilidad del diseño, no es buena.	Líder del proyecto.
= 1	Si el valor es igual a uno, todos los cambios se efectuaron en esta fase lo que sugiere una revisión de incluso, fases anteriores, pues la estabilidad del diseño queda en duda. Es competencia de los responsables del proyecto determinar si hacer o no una nueva ejecución de la fase.	Líder del proyecto.
Eficiencia en la detección de defectos: ayudará a medir la calidad del diseño desarrollado.		
Resultado	Interpretación	Responsable
>= 0	Este valor se puede dar por varias razones, o no se detectaron defectos en el diseño o aún no se ha comenzado la verificación de defectos en esta fase. Pocas veces se crea un diseño libre de errores.	
< 1	Mientras el valor se aproxime a cero, denota que el trabajo anterior, es decir, el trabajo realizado en la fase anterior es el correcto y genera como consecuencia la elaboración del diseño correcto. Por el contrario si se acerca a uno, indica que la mayor cantidad de defectos se encontró en esta fase lo que sugiere una revisión de la metodología para futuras ocasiones. Si la cantidad de defectos encontrados es alta (y este número dependerá de la magnitud del proyecto), el responsable del proyecto junto con el líder deben tomar decisiones al respecto.	Tanto el gerente de proyecto como el líder del proyecto deben revisar los defectos encontrados y deben tomar la decisión de rehacer algunas o todas las fases anteriores.
= 1	Si el valor es exactamente uno, quiere decir que todos los defectos se encontraron en esta fase. Dependiendo de la cantidad de defecto encontrados en el proyecto es un resultado poco deseado. Los responsables del proyecto debe tomar las medidas necesarias.	El líder del proyecto debe decidir si continuar con la siguiente fase o rehacerla.

Fuente: Diseño del investigador

4.3.3 Fase de Desarrollo

4.3.3.1 Objetivos

El principal objetivo de esta fase es asegurar que las especificaciones de diseño hayan sido correctamente implementadas y verificar que se haya probado la ejecución del software en un ambiente separado (test), siendo aproximadamente el mismo entorno operacional en que será ejecutado luego (producción).

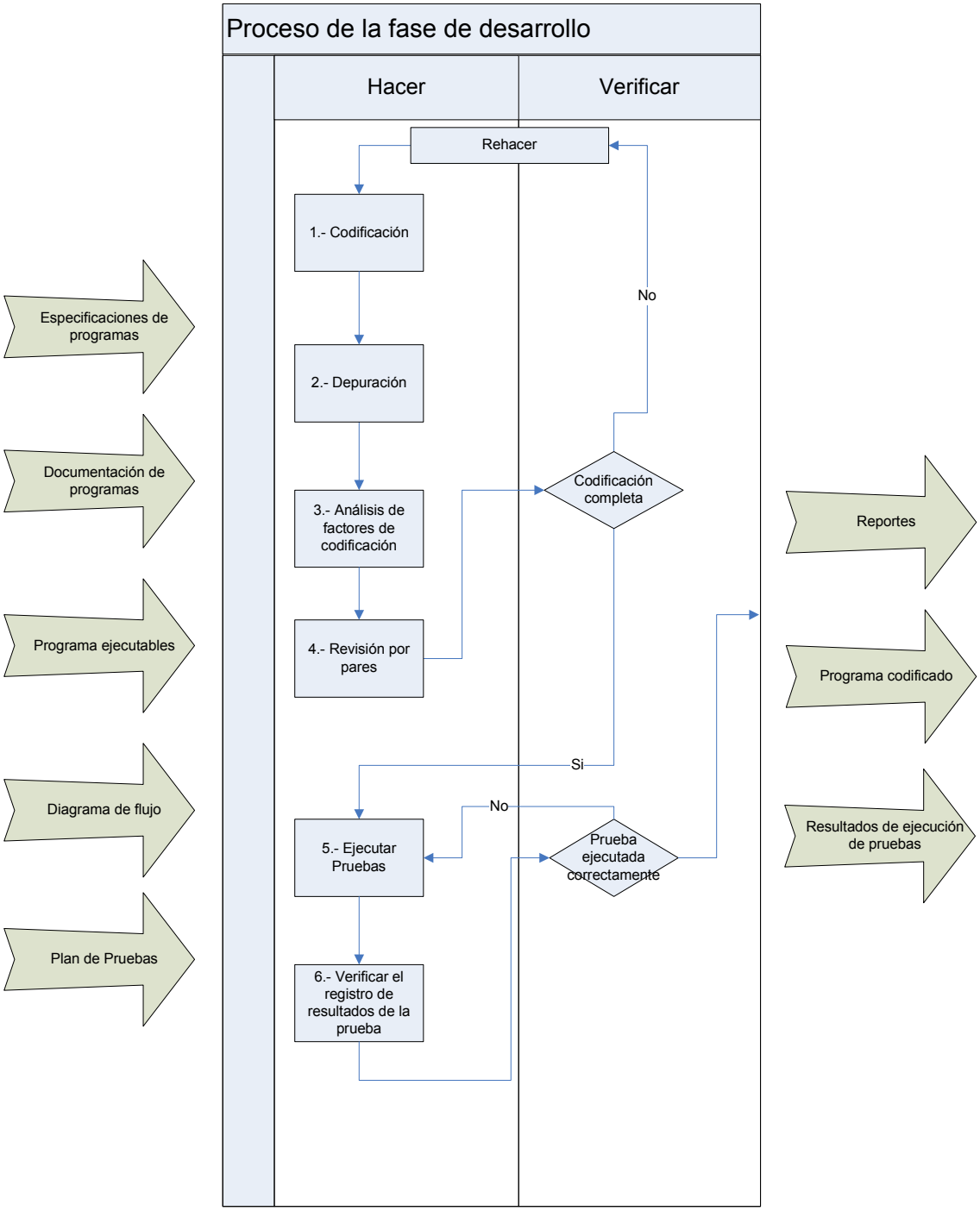
4.3.3.2 Entradas

Los entregables que funcionan como entrada al proceso de desarrollo son los siguientes:

- Especificaciones de programas.
- Documentación de programas a utilizar.
- Programas ejecutables.
- Diagramas de flujo.
- Estándares de desarrollo.
- Planes de pruebas:
    - Pruebas de interfaces.
    - Pruebas de regresión.
    - Pruebas de integración.
    - Pruebas de “stress”.
- Datos de prueba.

4.3.3.3 Procesos

Gráfico 7 Procesos de la fase de desarrollo



Fuente: el autor

4.3.3.3.1 Codificación

La codificación consiste en transformar todos los diagramas generados en pasos previos, en líneas de código fuente o lenguaje de programación que una vez compilados sean entendidos por el computador.

Existen diversos lenguajes para lograr esto. Uno de ellos es PHP que es un lenguaje de código abierto que se adapta a la filosofía del software libre, pero es solo para desarrollar aplicaciones en ambientes WEB.

Otro lenguaje muy utilizado es Java, pero actualmente en el país existe una discusión en el Ministerio de Ciencia y Tecnología, donde se delibera si este lenguaje es o no un lenguaje para código abierto, y por lo tanto si se puede o no utilizar bajo el lineamiento de software libre.

Para lograr este objetivo existen diferentes metodologías que nos ayudan a codificar los diagramas. Este tema es tan extenso que podría ser un tópico de investigación aparte.

Lo más importante a resaltar en este punto es que se debe generar todo una lista de instrucciones que reflejen los diagramas generados con anterioridad.

4.3.3.3.2 Depuración

Se utilizan tres metodologías para la detección y eliminación de los errores en la codificación:

- Depuración de errores sintácticos.
  Las especificaciones y expresiones del producto deben desarrollarse de acuerdo con la metodología definida por la organización y los requerimientos recopilados. El programador puede chequear las expresiones y sintaxis para asegurarse que ha escrito el programa de acuerdo con las reglas establecidas. Al chequear la sintaxis se hacen preguntas como:
  * ¿La función a realizar, está claramente identificada?
  * ¿Las sentencias del programa están identificadas correctamente?
  * ¿Las sentencias del programa están construidas utilizando la estructura

apropiada?

* ¿Los elementos de datos están apropiadamente identificados?

- Depuración de errores estructurales.

  Las preguntas típicas durante la depuración estructural del producto son:

  * ¿Se colocaron todas las sentencias necesarias?

  * ¿Se utilizan todas las definiciones de datos en las sentencias definidas?

  * ¿Hay una clara separación de roles en cada uno de los elementos definidos?

  * ¿Se utilizan todos los elementos de datos definidos?

  * ¿Las tablas internas y valores límite están usados de manera que cuando se excede el límite se puede continuar procesando?

- Depuración de errores funcionales.

  Las funciones son los requerimientos que el programa ejecutará. Las preguntas cuando se depura la funcionalidad son:

  * ¿El programa ejecutará la función específica en la forma indicada?

  * ¿El sistema detectará datos incorrectos o ilógicos?, es decir, ¿habrá validaciones de datos?

4.3.3.3.3 Análisis de factores de codificación

A continuación, se detallan los factores a analizar:

- Implementación del control de integridad de información.
  Es necesario tener implementados controles específicos de manera de lograr la precisión en el procesamiento deseado.
- Implementación de reglas de autorización.
  Es necesario verificar la implementación de reglas de autorización de manera de dificultar su evasión. Esto es importante ya que garantiza una mejor seguridad e integridad en el procesamiento de datos.
- Implementación de procedimientos de seguridad.
  La seguridad es la combinación de previsión y entrenamiento, más herramientas y técnicas de seguridad. Los procedimientos que aseguran que tanto las herramientas como las técnicas de seguridad estén

disponibles y que trabajen juntas, deben desarrollarse durante la fase de codificación

- Cumplimiento del programa con la metodología a adoptar.
  Los procedimientos a implementar deben asegurar conformidad con estándares, políticas, procedimientos, leyes y métodos definidos por la organización. Si se detecta la no conformidad, se deberían tomar las medidas necesarias para modificar el diseño y así lograrla.
- Programa acorde al diseño (Facilidad de uso).
  La implementación de especificaciones del sistema puede invalidar algunas de los aspectos del diseño respecto de la facilidad de uso. La programación es la traducción de especificaciones de diseño a lenguaje ejecutable por la máquina. Esto puede entorpecer el logro de la facilidad de uso. La codificación debe lograr la facilidad de uso, de igual forma en que se intenta cumplir con el resto de las especificaciones funcionales
- Programa acorde al diseño (Portabilidad).
  La portabilidad de los programas depende del lenguaje seleccionado y de cómo se usa ese lenguaje. Las especificaciones deben indicar lo que se hace y no se hace en la programación para lograr su portabilidad, y la codificación debe apegarse a esas especificaciones de diseño. Es recomendable que la portabilidad no quede en manos del programador, si no que este previamente definida.
- Programa acorde al diseño (Acoplamiento).
  Las especificaciones de diseño deben indicar parámetros a pasar desde y hacia otros módulos y aplicaciones del sistema. Normalmente es una buena práctica del programador, verificar que las especificaciones del sistema estén actualizadas antes que las funciones sean codificadas.
- Criterio de ejecución del programa (Performance).
  Una evaluación temprana del rendimiento, brindará una buena oportunidad al equipo de desarrollo para hacer ajustes si ésta no cumple con los requisitos mínimos.

4.3.3.3.4 Revisión de pares

Se utiliza la revisión por pares (peer review en inglés), para evaluar la estructura y funcionalidad del programa. La revisión por pares detecta errores sintácticos.

La revisión por pares contribuye a construir el código de un programa a través de revisiones informales, pero sin embargo efectivas, acerca de la funcionalidad del mismo.

Este tipo de revisión provee un análisis estático que evalúa la estructura y funcionalidad del programa. Puede detectar errores sintácticos en mayor medida que una simple observación visual, como resultado de un recorrido de requerimientos. (wikipedia, 2006)

4.3.3.3.5 Ejecutar pruebas

Para lograr una prueba efectiva, se debe usar un plan de pruebas creado previamente. Esta etapa es la culminación de un trabajo previo preparado para esta fase. Si esto no ocurre, la prueba podría resultar poco efectiva.

Durante esta fase, el equipo de proyecto, llevará a cabo la verificación de la ejecución de la prueba. Todo esto se realizará, tomando muestras representativas de casos y sus respectivas ejecuciones. Se verificará también, que al menos, los siguientes tipos de prueba hayan sido llevados a cabo durante esta fase:

Prueba de regresión

La prueba de regresión es la verificación de que lo que se está instalando o desarrollando no afecta a lo que ya estaba instalado si es el caso.

Prueba de autorización

En esta prueba deben verificarse las reglas de autorizaciones (permisos) para ejecutar las distintas transacciones de la aplicación.

Prueba de integridad

Debe ser verificada durante la prueba, los controles sobre la integridad de datos que maneje el sistema.

Prueba de pistas de auditoria

La función de pistas de auditoria, debe ser probada para asegurarse que para cualquier transacción, pueda realizársele un seguimiento desde su inicio hasta su fin, para lograr una correcta reconstrucción de la misma cuando sea necesario. Este punto es muy importante sobre todo en este tipo de organismos.

Prueba de " carga y stress"

Es la prueba que asegura y cuantifica el nivel de servicio del sistema ante el manejo de grandes volúmenes de datos.

Prueba de seguridad

En esta prueba lo que se intenta es violar todos los aspectos de seguridad del sistema queriendo obtener como resultado que ninguno de estos falle. Aunque es bien sabido en el mundo de la informática que ningún sistema es seguro, pues se intentará al menos asegurar las fallas más notables y que estas sean bien documentadas.

Prueba funcional

Esta prueba verifica el correcto cumplimiento de los requerimientos del usuario. Debe tener una importancia especial, pues va a asegurar la satisfacción del usuario final.

Prueba de operación

El éxito de esta etapa dependerá fuertemente de la habilidad de las personas que utilicen el sistema, además es importante que esta prueba se haga en un ambiente lo más parecido posible al de producción.

Inspecciones

En esta prueba lo que se hace es evaluar el código del sistema para verificar ciertas reglas que lo harán fácil de mantener ante supuestas modificaciones posteriores. Dichas reglas, deberán estar explicadas dentro de los estándares de la organización.

Prueba de performance

Los criterios de performance son establecidos durante la fase de requerimientos e intentan ser medidos en esta fase. De no poder realizarse porque el ambiente necesario

para llevarla a cabo sería muy costoso o difícil de reproducir, se dejará para ser medido en el cierre.

4.3.3.3.6 Verificar el registro de resultados de la prueba

Ya que documentar un problema, es el primer paso para la corrección del mismo, es necesario llevar adelante una verificación del registro de los resultados de la prueba. Esto se llevará a cabo, tomando muestras representativas de tales resultados, para su análisis y estudio posterior, ya que uno de los objetivos más importantes en la gerencia de proyectos es documentar las experiencias aprendidas para ser utilizadas más adelante en futuros proyectos y así ahorrar tiempo y dinero.

Se verificará que estén descriptos al menos, los siguientes aspectos, para todos los problemas encontrados en las pruebas:

- Declaración del problema: Establecer cuál es el problema.
- Criterio: Establecer cómo debería comportarse el sistema.
- Efecto: Es la diferencia entre el problema y el comportamiento esperado.
- Causa: ¿Qué puede haber causado el problema?

4.3.3.4 Salidas

Dos son las salidas de este proceso:

- Listado de errores encontrados y solucionados.
- Programa codificado.

4.3.3.5 Lista de Verificación

Tabla 8 Lista de verificación de la fase de desarrollo

		Respuesta			
número	Ítem	Si	No	No aplica	Observaciones
1	¿Los programadores utilizaron técnicas de debugging o depuración durante el desarrollo del producto?				

2	¿El producto fue sometido a una prueba o test completo antes de pasar a la siguiente fase?				
3	¿Se documentaron los defectos de programación encontrados en esta fase?				
4	¿Se corrigieron todos los defectos encontrados?				
5	¿El programador ejecuto la verificación y validación del programa?				
6	¿Las pruebas fueron realizadas tal como se planearon?				
7	¿Existió un ambiente de prueba?				
8	¿Estuvo correcta la estimación de tiempo para la ejecución de las pruebas?				
9	¿Se documentaron los resultados de las pruebas?				

Fuente: Diseño del investigador

4.3.3.6 Métricas

$$\text{Eficiencia en la detección de defectos}: \frac{\text{Cantidad de defectos detectados hasta instalación}}{\text{Cantidad total de defectos hasta instalación}}$$

$$\text{Eficiencia en la remoción de defectos}: \frac{\text{Cantidad de defectos removidos hasta instalación}}{\text{Cantidad total de defectos hasta instalación}}$$

4.3.3.7 Matriz de comparación de resultados para la métrica

Tabla 9 Matriz de interpretación de resultados para las métricas para la fase de Desarrollo

Eficiencia en la detección de defectos: Esta métrica determinará que tan efectivo es el proceso de desarrollo.		
Resultado	Interpretación	Responsable
>= 0	Este resultado se puede interpretar de dos maneras: o no se ha iniciado la fase de desarrollo al momento de hacer la medición, o no hubo defectos detectados en la fase. Es el resultado ideal.	

< 1	Mientras el valor se aproxime a cero, denota que el trabajo anterior, es decir, el trabajo realizado en las fases anteriores es el correcto y genera como consecuencia un correcto desarrollo. Por el contrario si se acerca a uno, indica que la mayor cantidad de defectos se encontró en esta fase lo que sugiere una revisión, por parte de los responsables, de la metodología para futuras ocasiones e incluso, una revisión de la ejecución de la fase actual.	El líder del proyecto.
= 1	Si el valor es exactamente uno, quiere decir que todos los defectos se encontraron en esta fase. Dependiendo de la cantidad de defecto encontrados en el proyecto es un resultado poco deseado y sugiere una revisión de la ejecución de esta fase por parte de las personas responsables. De ser muchos los errores encontrados aquí es aconsejable revisar la ejecución de fases anteriores.	Gerente del proyecto, líder del proyecto
Eficiencia en la remoción de defectos: ayudará a medir la calidad del desarrollo.		
Resultado	Interpretación	Responsable
0	No se removió ningún defecto. Esto puede indicar que no se ha iniciado la fase, o que se realizó una buena fase de desarrollo.	
< 1	Si el valor se aproxima a cero, indica que son pocas las remociones de defectos efectuadas en esta fase. Si por el contrario se acerca a uno, indica que la mayoría de defectos encontrados han sido removidos en esta fase y es conveniente revisar la ejecución de la misma.	Líder del proyecto y los desarrolladores.
= 1	Indica que la totalidad de remociones de defectos ha sido efectuada en esta fase. Dependiendo de la cantidad de defectos encontrados, es conveniente, por parte de los responsables, tomar la decisión de revisar toda la fase e incluso fases anteriores para determinar posibles fallas de la metodología.	Líder del proyecto,

Fuente: Diseño del investigador

4.3.4 Fase de Cierre

Una vez finalizada la fase de desarrollo y ejecutadas las pruebas internas se procede a ejecutar la fase de cierre.

4.3.4.1 Objetivos

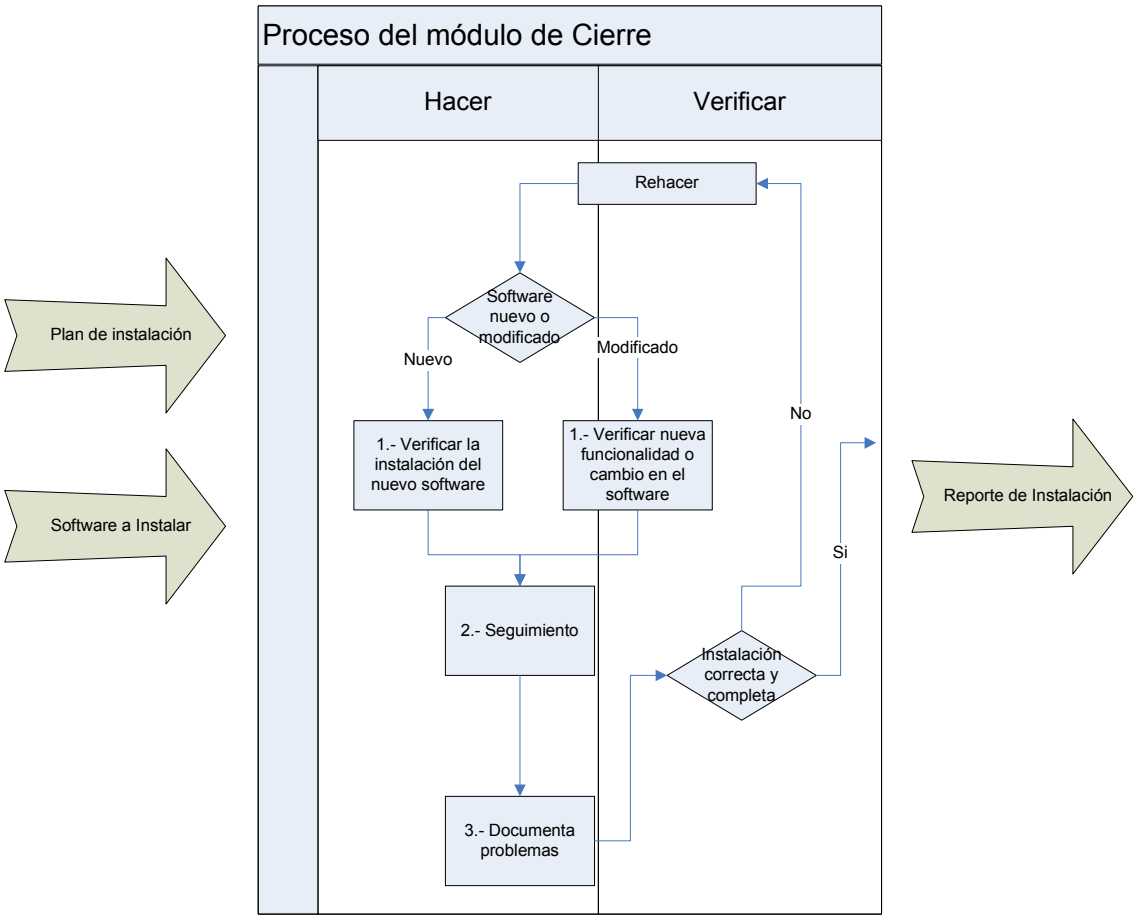
El objetivo de esta fase es la instalación del producto en el ambiente de producción y hacer entrega formal a los usuarios que van a hacer uso de la herramienta.

4.3.4.2 Entradas

- Plan de instalación (sólo si es requerida una instalación especial).
- Instrucciones y procedimientos para los nuevos usuarios.
- Resultado del proceso de instalación.

4.3.4.3 Procesos

Gráfico 8 Proceso de módulo de cierre



Fuente: el autor

4.3.4.3.1Verificar la instalación del nuevo software

La fase de instalación posee dos dificultades para el equipo de trabajo. La primera es que la instalación es un proceso separado del resto del desarrollo de la aplicación. Estas funciones no están relacionadas con las necesidades del usuario, pero sí con las de instalar en producción una aplicación completa y verificada. La segunda dificultad es que la instalación ocurre en un lapso de tiempo muy corto. Es común que la instalación dure unas horas. Por lo tanto la verificación debe ser bien planeada y ejecutada.

Es importante que los resultados de la verificación estén disponibles antes de la instalación completa del sistema. El objetivo de la verificación es determinar si la instalación fue exitosa, por lo tanto los resultados deben estar disponibles lo antes posible. Esto significa que los resultados a obtener de la verificación deben estar explícitos antes de que la verificación comience. Los factores a tener en cuenta en la verificación de la instalación son:

- Integridad y exactitud de la instalación.
- Prohibición de modificación de datos durante la instalación.
- Integridad de archivos de producción.
- Integridad del sistema/versión anterior asegurado (continuidad de procesamiento).
- Control de acceso durante la instalación.
- Puesta a disposición de instrucciones de instalación.
- Monitoreo de la performance.
- Procedimientos operativos.

4.3.4.3.2 Verificar nueva funcionalidad o cambio del software

El objetivo principal es instalar el cambio requerido en el tiempo adecuado. A continuación se detallan los objetivos específicos de la instalación de cambios:

- Poner a la aplicación modificada en producción.
- Monitorear la exactitud de los cambios.
- Mantener actualizadas con las últimas versiones disponibles, los ambientes de prueba y producción.

4.3.4.3.3 Seguimiento

Los sistemas aplicativos son propensos a tener problemas inmediatamente después de introducir una nueva versión. Por eso es necesario monitorear las salidas del producto, una vez instalada la nueva versión en producción.

El equipo de proyecto y los usuarios deberán reportar, los problemas que se presenten en el sistema. Estos problemas deben ser documentados mediante el uso de un formulario. Es lo más recomendable, para mantener una estandarización de la información.

4.3.4.3.4 Documentar problemas

Los problemas detectados durante el monitoreo de los cambios, deben ser documentados.

Además se debe evaluar el riesgo asociado a ese problema. El reporte de un problema causado por un cambio en el sistema, permite asociar tal problema a lo específico del cambio. Este reporte debe ser enviado al desarrollador o analista encargado para su análisis y posterior corrección.

4.3.4.4 Salidas

- Reporte de puestas en producción.
- Reporte de notificación de monitoreo de cambios de software.
- Reporte de problemas causados por cambios de Software.

4.3.4.5 Lista de Verificación

Tabla 10 Lista de verificación de la fase de cierre

		Respuesta			
número	Ítem	Si	No	No aplica	Observaciones
1	¿el proceso de instalación fue debidamente documentado?				

2	¿Los cambios de la nueva versión en el ambiente de producción fueron debidamente documentados?				
3	¿Se notifico a los usuarios de los cambio que se iba a efectuar en el ambiente de producción?				
4	¿Existe un formulario para documentar los problemas encontrados inmediatamente después de efectuar el cambio o instalación del producto?				
5	¿Existen procedimientos para la fase de post-instalación y el seguimiento de la utilización del sistema?				
6	¿Se verifico la funcionalidad del producto con el usuario final?				

Fuente: Diseño del investigador

4.3.4.6 Métricas

$$\text{Eficiencia en la detección de defectos}: \frac{\text{Cantidad de defectos detectados hasta instalación}}{\text{Cantidad total de defectos hasta instalación}}$$

$$\text{Eficiencia en la remoción de defectos}: \frac{\text{Cantidad de defectos removidos hasta instalación}}{\text{Cantidad total de defectos hasta instalación}}$$

4.3.4.7 Matriz de comparación de resultados para la métrica

Eficiencia en la detección de defectos: Esta métrica ayudará a determinar que tan correctos fueron los resultados de las fases anteriores.		
Resultado	Interpretación	Responsable
>= 0	Este resultado se puede interpretar de dos maneras: o no se ha iniciado la fase de cierre al momento de hacer la medición, o no hubo defectos detectados en la fase. Es el resultado ideal.	
< 1	Mientras el valor se aproxime a cero, denota que el trabajo anterior, es decir, el trabajo realizado en las fases anteriores es el correcto y genera como consecuencia un producto correcto. Por el contrario si se acerca a uno, y dependiendo de la cantidad de defectos detectados, indica que la mayor cantidad de defectos se encontró en esta fase lo que sugiere una revisión de la metodología para futuras ocasiones, e incluso considerar la opción de ejecutar nuevamente	Gerente de proyecto, líder del proyecto.

	fases anteriores.	
= 1	Si el valor es exactamente uno, quiere decir que todos los defectos se encontraron en esta fase. Dependiendo de la cantidad de defecto encontrados en el proyecto es un resultado poco deseado y se sugiere hacer una revisión de fases anteriores para determinar las causas de los defectos.	
Eficiencia en la remoción de defectos: ayudará a medir la calidad del producto.		
Resultado	Interpretación	Responsable
0	No se removió ningún defecto. Esto puede indicar que no se ha iniciado la fase, o que se realizó una buena fase de cierre.	Líder del proyecto.
< 1	Si el valor se aproxima a cero, indica que son pocas las remociones de defectos efectuadas en esta fase. Si por el contrario se acerca a uno, indica que la mayoría de defectos encontrados han sido removidos en esta fase y es conveniente revisar la ejecución de la misma.	Líder del proyecto.
= 1	Indica que la totalidad de remociones de defectos ha sido efectuada en esta fase. Dependiendo de la cantidad de defectos encontrados, es conveniente, por parte de los responsables, tomar la decisión de revisar toda la fase e incluso fases anteriores para determinar posibles fallas de la metodología.	Gerente del proyecto, líder del proyecto.

Fuente: Diseño del investigador

4.4 Factibilidad

La aplicación de esta metodología en los distintos organismos gubernamentales es totalmente factible, ya que esta desarrollada pensando en la fácil aplicación y en la generación de resultados visibles y de fácil medición.

Económicamente es totalmente factible, pues no requiere mayor inversión, más que entrenar al personal que se encargará de aplicar dicha metodología.

Tecnológicamente también es viable, pues no requiere de ninguna inversión en tecnología y con los equipos existentes se puede desarrollar a plenitud la metodología.

Análisis de los Datos

La misión de una organización o área que desarrolla proyectos de automatización de sistemas de información debe ser entregar soluciones de alta calidad dentro del

tiempo, alcance y presupuesto planificado, esta condicional no puede verificarse cuando existe ausencia de planificación derivada de la falta de aplicación de metodologías y gestiones que ocasionan que muchos de estos proyectos no tengan definidos claramente su alcance, no culminen en los tiempos previstos, o sobrepasen el presupuesto inicial, y en el peor de los casos sean cancelados.

De allí que se haga indispensable la implementación de metodologías que permitan alcanzar las metas en el marco de la calidad, oportunidad y tiempo. En Venezuela pocas organizaciones gubernamentales se ajustan al cuadro anteriormente señalado, esto es perceptible debido a la falta de indicios sobre esto, en los manuales o en cualquier otro medio bibliográfico que sirva para conocer la memoria y formas de funcionamiento de las instituciones.

Esto se debe a que las metodologías para el desarrollo de sistemas de información son muchas veces complicadas y abstractas; para su diseño e implementación inicial se requiere de expertos de alto nivel, planificación estratégica y un lapso de tiempo considerable.

En relación a las fases de desarrollo aquí planteadas, se puede notar la similitud de las fases de desarrollo de proyectos que plantea el PMI, por lo tanto es de fácil aplicación todos los conocimientos que este organismo sugiere.

Al finalizar cada fase se debe generar un producto que deje constancia del trabajo elaborado en la fase y que permita de alguna manera evaluar la calidad del trabajo realizado, para de esta manera, los responsables de la ejecución del proyecto, junto con la evaluación diaria, puedan tomar decisiones sobre los resultados de las mismas.

En relación a las métricas, existe poco desarrollo del tema hasta ahora, y es casi nula su aplicación en el país. Este hecho se puede evidenciar por la poca cantidad de trabajos investigativos desarrollados que toquen este tópico. Las organizaciones gubernamentales no elaboran este tipo de instrumentos, de lo contrario serían abundantes este tipo de datos en los informes de ejecución con indicaciones de los resultado obtenidos de la evaluación de cada fase del desarrollo de un sistema.

La métrica, responde a las características de la organización y constituye un esfuerzo intelectual a nivel abstracto y verbal, debido a lo complejo que puede resultar la cuantificación de la evolución de los sistemas de información.

El diseño y aplicación de una metodología de trabajo no es tarea sencilla, pues requiere tiempo y dedicación, además de acciones para su continuo mejoramiento y adaptabilidad a nuevas necesidades.

Recomendaciones y Conclusiones

5.1 Recomendaciones

- Profundizar en la gestión de control de calidad para proyectos y analizar futuros posibles impactos y posibles mejoras.
- Poner en práctica esta metodología y retroalimimentarla para su mejoramiento.
- Crear un equipo de trabajo que se encargue exclusivamente de la gestión de la calidad de los proyectos y de la aplicación de esta metodología.
- Desarrollar un plan estratégico de mejoramiento continuo de la metodología para que cada aplicación aporte un mejoramiento de la misma.
- Incrementar el número de métricas existentes para un control más detallado de la aplicación de la metodología.
- Desarrollar una política de gestión de documentos para mantener organizados los productos entregables al finalizar cada fase de proyecto, para de esta manera elaborar una buena bibliografía que sirva de sustento a las lecciones aprendidas en cada proyecto.
- Plantear y desarrollar nuevas investigaciones entorno al tema del control de la calidad de proyectos de desarrollo informático.

5.2 Conclusiones

El desarrollo de software del Estado Venezolano debe seguir los estándares del software libre, para cumplir con las leyes recientemente promulgadas que obligan a los organismos del Estado a adoptar software libre de calidad que ayuden a mejorar la gestión diaria de estos.

Las metodología utilizadas en este trabajo investigativo ayudan a logra éste objetivo y es imperativo que los organismos del Estado Venezolano adopten algunas de ellas para el desarrollo de sistemas automatizados de calidad, ya que en su mayoría carecen de metodologías.

Existen en el mercado muchas metodologías para el control de la calidad en el desarrollo de software. La metodología propuesta en este trabajo investigativo ayuda a lograr el control de la calidad en el proceso del desarrollo de software.

Es imprescindible desarrollar o planificar una estrategia de mejoramiento continuo de la metodología para ir afinando sus pasos. Para ello se hace necesario indudablemente el compromiso del equipo de trabajo, para que cada uno aporte ideas a este plan de mejoramiento y el manejo de las lecciones aprendidas en la ejecución de cada proyecto.

También se hace necesario incrementar las cantidades de métricas que se adapten a cada organización y a cada equipo de trabajo, para que de esta manera ayuden a un mejor control en el desarrollo de los proyectos que apliquen la metodología aquí planteada.

Bibliografía

American Society for Quality (2006). American Society for Quality. Consultado en Agosto 16, 2006 en http://www.asq.org/glossary/q.html.

Carnegie Mellon University - Software Engineering Institute (1994). The Capability Maturity Model: Guidelines for Improving the Software Process. Pittsburgh: Addison Wesley.

Castell,Manuel (2005). *La era de la Información es nuestra era*. Consultado en Mayo, 29, 2005 en http://vrpd.universidadarcis.cl/ant/index.php?option=com_content&task=v iew& id=28&Itemid=69.

Cervera Paz, Ángel (2002). Modelo de Mccall Como Aplicación de la Calidad a la Revisión del Software de Gestión Empresarial. Consultado en Mayo 29, 2005 en http://www.monografias.com/trabajos5/call/call.shtml.

Glass, R. (1998). *Defining Quality Intuitively*. IEEE Software.

Grady, R., Caswell, D. (1987). *Software Metrics: Establishing a Company-Wide Program*. USA: Prentice Hall.

Gremba, J., Myers, C. (2005). *The IDEALSM Model: A Practical Guide for Improvement*. Consultado en Junio 11, 2005 en http://www.sei.cmu.edu/ideal/ideal.bridge.html.

Haughey, D (2004). *Why Software Projects Fail and How to Make Them Succeed*. Consultado en Julio 25, 2006 en

Kaplan, C., Clark, R., Tang, V., (1995). *Secrets of Software Quality: 40 Innovations from IBM*. New York: McGraw-Hill, Inc.

Mc Call, J., Richards, P., Walters, G (1977). *Factors in Software Quality*. Rome Air Development Center: United States Air Force.

McFeeley, B. (1996). *IDEAL^{SM}: A User's Guide for Software Process Improvement*. Pittsburgh: Carnegie Mellon University.

Montes, G (2002). *Metodología PMI® complementada para gestionar proyectos de desarrollo de software y de implantación de aplicaciones de alto impacto en las organizaciones*. Santafé de Bogota: PMI Colombia.

Pressman, R. (2001). *Software Engineerin: A Practitioner's Approach*. Chicago: McGraw-Hill.

Programa de las Naciones Unidas para el desarrollo en Venezuela (PNUD) (2003). *Informe sobre Tecnologías de la Información y la Comunicación y el Desarrollo Humano en Venezuela*. Caracas: Silva Cordoliani.

Wikipedia, La enciclopedia libre (2005). *Globalización*. Consultado en Mayo, 01, 2005 en http://es.wikipedia.org/wiki/Globalizaci%C3%B3n.

Wikipedia, La enciclopedia libre (2005). *Ingeniería del software*. Consultado en Junio, 05, 2005 en http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software.