Universidad de las Ciencias Informáticas

Facultad 1

Título: Estrategia de Aseguramiento de la Calidad para el Sistema Único de Identificación Nacional de la República de Cuba

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autora: Ana de la Caridad Gómez Gregorio
Tutoras: Msc. Yudenia Ramírez Mastrapa
Ing. Brisey López Bello

Ciudad de La Habana, 4 de junio de 2010

DECLARACIÓN DE AUTORÍA

Declaro ser la única autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los _____ días del mes de _____ del año 2010.

Ana de la Caridad Gómez Gregorio

_____

Firma de la Autora

Msc. Yudenia Ramírez Mastrapa

_____

Firma de la Tutora

Ing. Brisey López Bello

_____

Firma de la Tutora

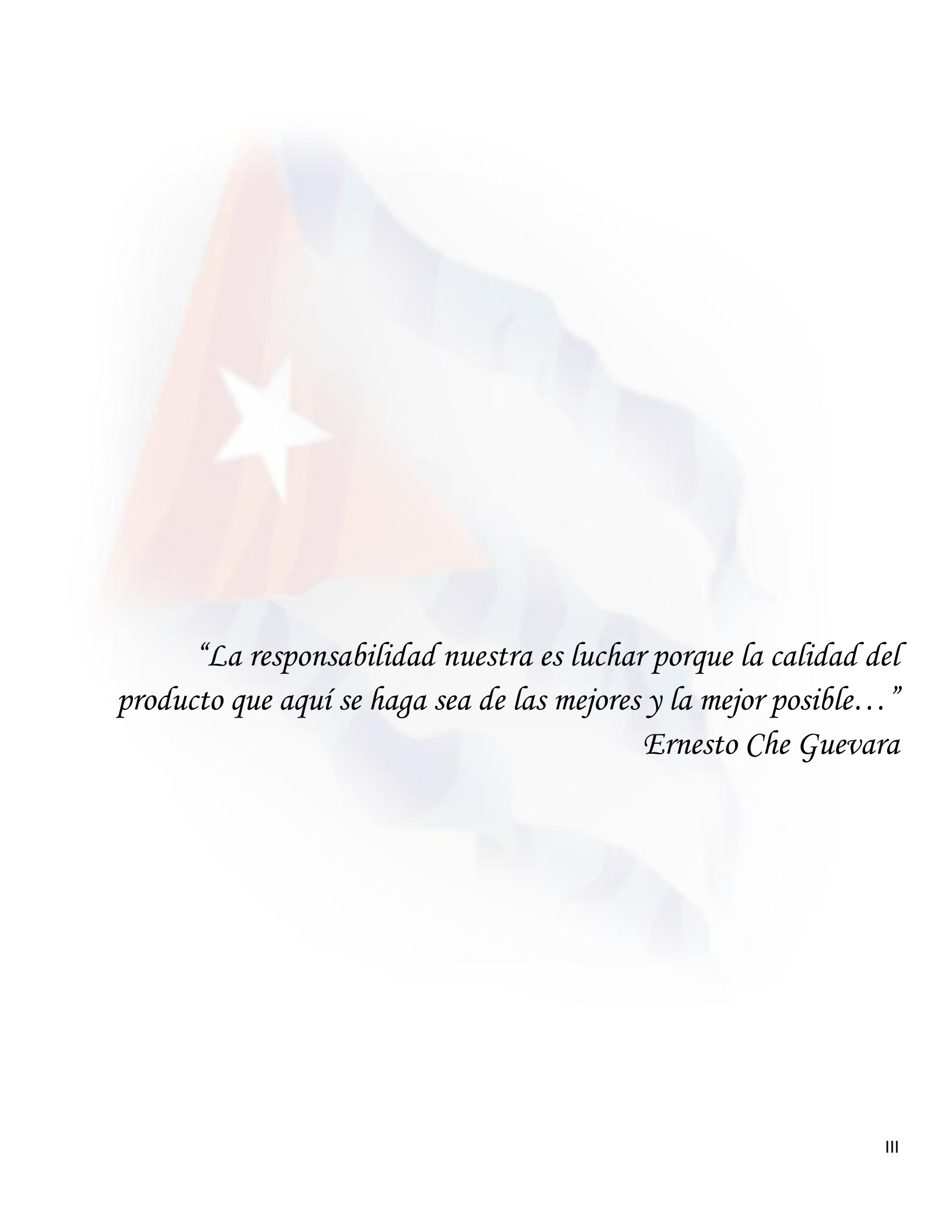
DATOS DE CONTACTO

Msc. Yudenia Ramírez Mastrapa

- Ingeniera Informática, CUJAE, 2003.
- Categoría docente: Profesora Asistente.
- Máster en Gestión de Proyectos Informáticos.
- Profesora del Departamento de Sistemas Digitales, Facultad 1, Universidad de las Ciencias Informáticas.
- Jefa de departamento de Soluciones de software del Centro de Identificación y Seguridad digital.
- Líder del proyecto Sistema Único de Identificación Nacional.
- Correo electrónico: yudenia@uci.cu

Ing. Brisey López Bello

- Ingeniera en Ciencias Informáticas, UCI 2008, Título de Oro.
- Profesora del Departamento de Sistemas Digitales, Facultad 1, Universidad de las Ciencias Informáticas.
- Analista principal del proyecto Sistema Único de Identificación Nacional.
- Correo electrónico: bbello@uci.cu



"La responsabilidad nuestra es luchar porque la calidad del producto que aquí se haga sea de las mejores y la mejor posible…"

Ernesto Che Guevara

AGRADECIMIENTOS

A la Revolución y a nuestro Comandante en Jefe por crear esta maravillosa escuela y brindarme la posibilidad de acceder a la educación.

A mi abuela Ana por ser la persona que más amor me ha brindado, por quererme tanto, por haber luchado conmigo desde pequeña.

A mi tía Cary por considerarme la hija que nunca tuvo y soportar mis malacrianzas.

A mi mamá, por su apoyo incondicional, por confiar siempre en mí, por soportar la distancia todos estos años.

A mi tatico, Yasmany, por estar conmigo desde primer año apoyándome en los buenos y malos momentos, por ser mi amigo, mi amante.

A mi papá por inculcarme el amor hacia el estudio.

A Mercedes y a Shakira por su cariño.

A mis amistades de años anteriores, a Yobi, Daynelis, Javier, Arelis, Alain, Lizy e Ismelis.

A mi gente linda de 202, esos chicos maravillosos y llenos de alegría.

A Eliska, gracias por tus consejos, por tu compañía, por tu amistad, nuca te olvidaré.

A Yadi y Diana, mis compañeras de charla, las loquitas que me contagiaron con su alegría.

Al chico de Párraga, Pacheco, gracias por escucharme cuando lo necesitaba, por tus consejos.

A los Ale (Valdés y Torres), por los momentos agradables en el lab.

Y no por estar en último lugar son menos importantes: a Yude y Bri (mis tutoras), por toda la ayuda que me brindaron, por sus revisiones oportunas, por guiarme durante todo el trabajo.

En fin, los quiero a todos y los llevo en mi corazón…

DEDICATORIA

Al hijo que nunca pude conocer y sostener en mis brazos.
A mis abuelitos Elvira y Román que se encuentran en el cielo.
A mi mamita linda.
A mi tatico.
A mi tía Cary.
Y especialmente a mi abuela Ana, por ser la persona más maravillosa del mundo.

RESUMEN

En los últimos años, el Aseguramiento de la Calidad se ha convertido en una necesidad de toda empresa productora de software que persiga implantar soluciones que contribuyan a alcanzar los beneficios económicos y sociales correspondientes, de ahí se deriva la importancia de definir y establecer procesos que ayuden a controlar la calidad del producto que se pone a consideración del usuario o cliente.

La presente investigación centra su atención en el desarrollo y aplicación de una estrategia que permita asegurar la calidad en el Sistema Único de Identificación Nacional de la República de Cuba, con el objetivo de lograr que dicho sistema cumpla con las expectativas requeridas por el cliente, que exista una documentación que persista como constancia del trabajo realizado y que a la vez pueda servir para futuras verificaciones de la calidad del software. Se realiza un estudio de los principales estándares y modelos de calidad, incluyendo metodologías de desarrollo, y a partir de este se decide elaborar la propuesta de solución tomando como base el Libro de Proceso para PPQA (IPP-3520:2009) desarrollado por la Dirección de Calidad de la Universidad de las Ciencias Informáticas y la metodología seleccionada como guía de desarrollo: *MSF for CMMI*.

PALABRAS CLAVES

Aseguramiento de la calidad, calidad, estrategia.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La producción de software se ha caracterizado tradicionalmente por presentar problemas de calidad que se hacen visibles desde el instante en que inicia el proceso de desarrollo y que perduran hasta la entrega del producto final. La calidad desempeña un papel importante dentro del desarrollo del software, influyendo positivamente en la decisión de un cliente a la hora de seleccionar el producto que necesita.

La calidad del software es un concepto que ha ido variando con los años y existe una gran variedad de formas de concebirla. Grandes estudiosos del tema, como Kaoru Ishikawa, la han definido como: “Desarrollar, diseñar, manufacturar y mantener un producto de calidad que sea el más económico, el más útil y siempre satisfactorio para el consumidor”. (1)

Pressman, adentrándose más en el ámbito del software la define como “concordancia del software con los requisitos explícitamente establecidos, con los estándares de desarrollo expresamente fijados y con los requisitos implícitos, no establecidos formalmente, que desea el usuario”. (2)

Estudios realizados por diferentes autores, como Jorge Domínguez, demuestran que un alto porcentaje del éxito o fracaso del proyecto está en la tecnología disponible y en el conocimiento que se tenga de ella, además, en la forma en que el proyecto lleva un control de calidad durante su desarrollo. Los proyectos de software tienen ahora una tasa de éxito del 32% frente al 35% del estudio realizado en el 2006 y al 16% obtenido en 1994. Por otra parte, el 44% de los proyectos ha sido impugnado, mientras que el 24% se ha cancelado antes de su finalización o desarrollado sin utilizarse. (3)

	1994	1996	1998	2000	2002	2004	2006	2009
Exitosos	16%	27%	26%	28%	34%	29%	35%	32%
Cuestionables	53%	33%	46%	49%	51%	53%	46%	44%
Fallidos	31%	40%	28%	23%	15%	18%	19%	24%

Tabla 1. Situación de los proyectos de sistemas 1994 - 2009.

Los datos antes expuestos reflejan una situación que, a pesar del transcurso de los años, está vigente en la actualidad y que continúa afectando el desarrollo de software; incluso, las grandes empresas desarrolladoras no escapan de este mal.

El aseguramiento de calidad en el software no es un lema, es una necesidad. Al igual que la Ingeniería de Software, que sugiere la integración de actividades de ingeniería en el proceso de desarrollo, también es

necesario el uso de prácticas que conduzcan hacia productos terminados con calidad. De igual manera, se requiere de procesos que contribuyan a verificar, validar y controlar que dichas prácticas sean realizadas correctamente. Se han definido varios modelos basados en las experiencias exitosas de la Ingeniería de Software que sirven de guía para las mejoras y unifican los criterios de evaluación de las empresas.

Cuba presenta actualmente un crecimiento en la producción de software, el cual trae aparejado un incremento en el interés por garantizar la calidad en el software que se produce. Teniendo en cuenta la importancia de la vinculación de todas las ramas de la economía cubana con el mundo informático es de primordial interés para el Estado Cubano la consolidación de esta vinculación, no solo por los beneficios que trae desde el punto de vista del desarrollo de sistemas para el uso interno, sino también con el objetivo de introducirse en el mercado mundial.

La Universidad de las Ciencias Informáticas (UCI) es uno de los centros surgidos a raíz de la Batalla de Ideas. La misma se encuentra inmersa en proyectos de producción de software, por lo que es lógico deducir la necesidad de llevar un control de la calidad de los mismos. Sus estudiantes tienen la peculiaridad de vincular en sus actividades el estudio con la producción, ayudando de esta forma a dar respuesta a las peticiones de desarrollo de aplicaciones que llegan continuamente a la Universidad, ya sean de origen nacional o internacional.

La necesidad de crear en Cuba un sistema de identidad seguro y su impacto en las actividades de la sociedad fueron enunciados desde inicios del siglo pasado por Fernando Ortiz, fecha que puede ser citada como punto de partida para la aparición de diferentes regulaciones relativas a esta actividad en el territorio nacional. Desde el año 1997, los procesos relativos a la identificación son gestionados por la Dirección de Identificación y Registros (DIR) del Ministerio del Interior (MININT). Durante el último período del 2008 e inicios del 2009 se ha realizado un diagnóstico del estado actual del Sistema Nacional de Identificación de la población a nivel nacional e internacional en función de proponer la modernización del mismo para garantizar la inscripción e identificación de las personas naturales, con procesos y documentos seguros que incluyan el uso de la biometría a partir del cual se genere la base de datos única de identificación de la población como premisa indispensable para el Gobierno en Línea en el país. (4)

Debido a la gran envergadura que presenta el desarrollo del Sistema Único de Identificación Nacional (SUIN), y a que el proceso de desarrollo se encuentra en la etapa inicial, aún no se han definido el control

de la documentación, procedimientos para ajustarse a estándares, técnicas como revisiones formales, auditorías internas, así como mecanismos de control de la calidad.

Todo lo anterior conlleva a buscar una solución al siguiente **problema científico**: *¿Cómo garantizar el aseguramiento de la calidad en el Sistema Único de Identificación Nacional de la República de Cuba?*

Partiendo del problema planteado se tiene que el **objeto de estudio** se enfocará en *el proceso de gestión de la calidad durante el desarrollo de proyectos de software de gestión.*

El **campo de acción** de la investigación lo constituye *el aseguramiento de la calidad en el proyecto Sistema Único Nacional de Identificación Nacional de la República de Cuba.*

Para resolver el problema planteado con anterioridad se propone como **objetivo general**: *Implementar una estrategia para el aseguramiento de la calidad en el proyecto Sistema Único Nacional de Identificación Nacional de la República de Cuba que tribute al éxito del mismo y a la satisfacción del cliente.*

A partir del análisis del objetivo general se derivan los siguientes **Objetivos específicos**:

- Realizar el marco teórico de la investigación.
- Conformar una estrategia de aseguramiento de la calidad.
- Aplicar la estrategia de aseguramiento de la calidad propuesta.

La investigación se sustenta sobre la siguiente **Hipótesis**:

Si se elabora y aplica una estrategia de aseguramiento de la calidad para el proyecto Sistema Único Nacional de Identificación Nacional de la República de Cuba, se logrará obtener un producto que satisfaga las expectativas de calidad del cliente.

Para lograr los objetivos trazados y demostrar la hipótesis, se acometieron las siguientes **tareas**:

- Realización de un estudio bibliográfico.
- Definición de actividades de aseguramiento de la calidad.
- Identificación de los principales estándares y modelos relacionados con la calidad.
- Selección de técnicas de aseguramiento de la calidad a emplear en el proyecto.
- Formalización de la estrategia de aseguramiento de la calidad.
- Aplicación de la estrategia y seguimiento a los resultados obtenidos.
- Valoración de los resultados obtenidos.

Para la realización de las tareas se emplearon los **métodos** siguientes:

Métodos teóricos:

- **Analítico-sintético:** para sintetizar y obtener los elementos más importantes de la información contenida en todos los documentos analizados.
- **Análisis histórico-lógico:** para conocer la evolución y desarrollo de los estándares relacionados al aseguramiento de la calidad. Estos métodos ayudaron a desarrollar el contenido principal de la investigación.
- **Modelación:** se utilizó para la definición de la estrategia a aplicar debido a que es necesario modelar el fenómeno una vez identificados sus componentes revelando la unidad de lo subjetivo y lo objetivo en el objeto de investigación. (5)

Métodos empíricos:

- **Observación:** permite obtener la información del comportamiento del objeto de investigación tal y como éste se da en la realidad, es decir, es una forma de obtener información directa e inmediata sobre el fenómeno. (6)
- **Entrevistas:** al entrevistar a especialistas y expertos en el tema que se quiere investigar.

Como **resultados esperados** se tienen:

- Elaborar una estrategia para el aseguramiento de la calidad en el Sistema Único de Identificación Nacional de la República de Cuba.
- Implementar la estrategia durante la ejecución del proyecto.
- Valorar los resultados obtenidos durante la ejecución y cierre del proyecto.
- Transmitir las lecciones aprendidas a la organización con el fin de llevar a escalas superiores los resultados de los equipos de proyecto.

Este trabajo de diploma estará distribuido en tres capítulos:

En el **Capítulo 1** se abordarán temas relacionados con la fundamentación teórica de la investigación, donde se enuncian definiciones relativas a la calidad del software y aseguramiento de la calidad. También se abordan conceptos fundamentales relacionados con la estrategia de aseguramiento de la calidad, los niveles de prueba, tipos de pruebas y las actividades relacionadas a los procesos de Verificación & Validación.

En el **Capítulo 2** se presenta la propuesta de la estrategia de aseguramiento de la calidad que medirá la calidad del producto software Sistema Único de Identificación Nacional. Se explican todos los aspectos concernientes a la creación de la Estrategia de Aseguramiento de la Calidad, se detallan los elementos relacionados con su estructura, el papel que juegan actividades y tareas, incluyendo los métodos y procedimientos que se aplican, así como los responsables de ejecutarlas.

En el **Capítulo 3** se presentan los resultados obtenidos después de haber aplicado la estrategia en el proyecto, así como los métodos empleados para publicar y dar seguimiento a las deficiencias encontradas en aras de evitar la repetición de los errores en posteriores etapas de pruebas. Además, para obtener una visión general respecto al trabajo desarrollado se propone la inclusión de un epígrafe denominado Lecciones aprendidas.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Una de las problemáticas más grandes que se presentan en el desarrollo de sistemas de software es el aseguramiento de la calidad. La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo.

El presente capítulo trata diferentes epígrafes que sirven como base de conocimiento para entender la propuesta de solución a la problemática que trata esta investigación. Se aborda acerca de conceptos como calidad de software y aseguramiento de la calidad. Además, se describen algunas normas y modelos de calidad de software utilizados como guía para la estandarización del proceso de desarrollo.

1.2 Calidad de software

La IEEE[1] en su estándar 610.12[2] del año 1990 define "calidad" como el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario. (7)

Por otra parte, Pressman define la calidad del software como la "concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente". (2)

La Norma ISO 8402:1994[3] define a la calidad como: "La totalidad de rasgos y características de un producto o servicio, que conllevan la aptitud de satisfacer necesidades preestablecidas o implícitas". (8)
Existen estándares o metodologías que definen un conjunto de criterios de desarrollo para guiar la forma en que se debe aplicar la Ingeniería del Software.

[1] Siglas en inglés del Instituto de Ingenieros Eléctricos y Electrónicos.

[2] Glosario de Terminologías de Ingeniería de Software.

[3] Glosario de Aseguramiento de la calidad y Gestión de la calidad.

Hay dos definiciones que aunque apuntan en direcciones diferentes son igualmente válidas para el desarrollo de esta investigación: (9)

- En el enfoque del cliente, calidad del software es el grado en que un cliente y/o usuario percibe que el producto software satisface sus necesidades.
- Enfocado en la condición industrial del producto, calidad del software es la habilidad de un producto software de satisfacer su especificación de requerimientos.

La calidad siempre va a depender de los requisitos o necesidades que se deseen satisfacer, por eso, la evaluación de la calidad de un producto siempre va a implicar una comparación entre unos requisitos preestablecidos y el producto realmente desarrollado. Teniendo esto en cuenta, en un producto de software se van a tener diferentes visiones de calidad (10):

- Necesaria o requerida: la que requiere el cliente.
- Programada: la que se ha especificado explícitamente y se intenta conseguir.
- Realizada: la que se ha conseguido.

La calidad del software es medible y varía de un sistema a otro o de un programa a otro. Un software elaborado para el control de naves espaciales debe ser confiable al nivel de cero fallas; un software hecho para ejecutarse una sola vez no requiere el mismo nivel de calidad. Existen dos enfoques principales: la calidad del proceso y la calidad del producto. Medir la calidad del producto de software es un aspecto primordial, pero: ¿el producto software puede ser de calidad si el proceso no tiene calidad? El gran problema radica en que no se puede asegurar, por eso es que se habla de aseguramiento de la calidad y no simplemente de calidad del software, pues esta última en ambientes no controlados solo podría darse en casos fortuitos.

Según estudiosos del tema, como Adolfo Guzmán, plantean que:”La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. Es posible medir los principales atributos que forman o caracterizan a un software de buena calidad. La idea es que la calidad del software se identifica por ciertos atributos: fiabilidad, flexibilidad, robustez, comprensión, adaptabilidad, modularidad, complejidad, portabilidad, usabilidad, reutilización, eficiencia, y sin lugar a dudas es posible medir cada uno de ellos, y por consiguiente, caracterizar o medir la calidad del software en cuestión.” (11)
Teniendo en cuenta que todos los atributos por muy difíciles que sean pueden ser medidos relacionándolos con otro que sea más fácil de medir, el mismo autor los define de la siguiente manera:

- La fiabilidad (software confiable, pocos errores) se mide a través del número de mensajes de error, mientras más mensajes existan, contiene menos errores el software.
- La flexibilidad (capacidad de adaptación a diferentes tipos de uso, a diferentes ambientes de uso) o adaptabilidad.
- La robustez (pocas fallas catastróficas, el sistema “no se cae”) se mide a través de pruebas y uso prolongado.
- La comprensión (capacidad de entender lo que el sistema hace) se mide viendo la cantidad de comentarios que posee el software, y la extensión de sus manuales de usuarios.
- El tamaño se mide en *bytes*, o lo que es lo mismo el espacio que ocupa en memoria.
- La eficiencia de un programa se mide en segundos, es la rapidez en su ejecución (esta medición no tiene objeción, se mide lo que se quiere medir).
- La modularidad de un programa se mide contando el número de módulos que lo conforman.
- La complejidad de un programa se mide contando el número de anidaciones en expresiones o postulados, (es lo que se le llama complejidad ciclomática).
- La portabilidad es la facilidad con que se eche a andar en otro sistema operativo distinto al que fue creado. Se mide preguntando a usuarios que han hecho estos trabajos.
- La usabilidad de un programa es alta cuando el programa aporta gran valor agregado al trabajo. “Es indispensable contar con él”. Se mide viendo qué porcentaje de las necesidades cubre ese programa.
- La reutilización de un programa se mide por la cantidad de veces que partes del mismo se han reutilizado en otros proyectos de desarrollo de software.
- La facilidad de uso (ergonomía) caracteriza a los programas que no cuesta trabajo aprender, que se amoldan al modo intuitivo de hacer las cosas. Se mide observando la cantidad de pantallas que interactúan con el usuario.

Cuando se desarrolla un software es difícil asegurar que no se detectarán errores. La calidad de un sistema de software es algo subjetivo que depende del contexto y del objetivo que se pretenda conseguir. Para determinar dicho nivel de calidad se deben efectuar medidas o pruebas que permitan comprobar el grado de cumplimiento de las especificaciones iniciales del sistema. Las pruebas de software se integran en las diferentes fases del ciclo del software dentro de la Ingeniería de software. Los errores pueden

comenzar a aparecer desde el inicio del proceso de elaboración de los requisitos y continuar surgiendo en las etapas posteriores del diseño y desarrollo. La calidad de software es la vía de comprobación que permitirá verificar el cumplimiento de los objetivos de eficiencia que se planificaron para el software, garantizando un proceso ordenado y objetivo que conlleve a conclusiones exitosas garantizando la calidad óptima del software que será probado.

1.3 Aseguramiento de la calidad

El interés por la calidad crece de forma continua a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades. Como primera aproximación es importante diferenciar entre la calidad del producto software y la calidad del proceso de desarrollo. Las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo. Es importante destacar que la calidad de un producto de software debe ser considerada en todos sus estados de evolución a medida que avanza el desarrollo de acuerdo al ciclo de vida seleccionado para su construcción (especificaciones, diseño, código, entre otros.).

La Garantía de Calidad del Software (SQA, *Software Quality Assurance*, GCS[4], Gestión de la Calidad de Software) es una actividad de protección que se aplica a cada paso del proceso del software. Comprende procedimientos para la aplicación efectiva de métodos y herramientas, revisiones técnicas formales, técnicas y estrategias de pruebas, procedimientos de garantía de ajuste a los estándares y mecanismos de medida e información. El aseguramiento de la calidad del software es un patrón de acciones planificado y sistemático que se requiere para asegurar la calidad del software. (2)

Según el estándar 730-1998a[5] de la IEEE lo define como un patrón planificado y sistemático de todas las acciones necesarias para proveer confianza adecuada de que un componente o producto cumple con los requerimientos técnicos establecidos. (12)

El aseguramiento de la calidad es la aplicación de actividades planificadas y sistemáticas relativas a la calidad, para asegurar que el proyecto emplee todos los procesos necesarios para cumplir con los requisitos. (13)

[4] Gestión de Configuración del Software.

[5] Standard for Software Quality Assurance Plans.

De acuerdo a la Norma ISO 8402:1994 se define como aseguramiento de la calidad al “conjunto de actividades preestablecidas y sistematizadas aplicadas al sistema de calidad, cuya necesidad para dar confianza adecuada de que un producto o servicio cumplirá los requisitos para la calidad ha sido demostrada”. (8)

A partir de las definiciones expuestas con anterioridad se decide que el concepto de aseguramiento de la calidad por el que se regirá esta investigación es el siguiente:

“Conjunto de actividades y acciones encaminadas a la obtención de un producto que se ajuste al empleo de estándares y que cumpla con los requisitos previamente establecidos, a fin de satisfacer las necesidades de clientes y usuarios”.

El aseguramiento de calidad de software engloba los siguientes puntos (14):

- El mejoramiento de los métodos, técnicas de análisis, diseño, codificación y prueba.
- “Revisiones técnicas formales que se aplican durante cada fase del proceso de desarrollo de software” que ayudan a detectar los defectos.
- “Utilización de estándares” durante el desarrollo.
- “Sistema de Métricas” para la retroalimentación de todas las personas.
- Definición de estrategias de prueba multiescala.
- Control de la documentación del software y de los cambios realizados.
- Un procedimiento que asegure, siempre que sea posible, un ajuste a los estándares de desarrollo del software.

La utilización de estándares constituye un aspecto fundamental a tener en cuenta en el desarrollo de sistemas de identificación debido a que brindan los medios para que todos los procesos se realicen siempre de la misma forma. Son una guía para la productividad y la calidad, es decir, constituyen una norma o patrón a seguir.

1.4 Estándares y modelos de calidad del software

Según la Organización Internacional para la Estandarización (ISO), un estándar es “un conjunto de acuerdos documentados que contienen especificaciones técnicas u otros criterios precisos para ser usados constantemente, como reglas, lineamientos o definiciones de características. Todo esto con la finalidad de asegurar que los materiales, productos, procesos y servicios son óptimos para su propósito”.

1.4.1 Estándar Internacional IEEE 830-1998

Una Especificación de Requisito de Software (ERS) forma parte de la documentación asociada al software que se está desarrollando, por tanto debe definir correctamente todos los requerimientos, pero no más de los necesarios. Esta documentación no debería describir ningún detalle de diseño, modo de implementación o gestión del proyecto, ya que los requisitos se deben describir de forma que el cliente pueda entenderlos. Al mismo tiempo, se da una mayor flexibilidad a los desarrolladores para la implementación.

El documento de Especificación de Requisitos Software supone una especie de contrato entre cliente y desarrolladores en el que los primeros indican sus necesidades, mientras que los otros se limitan a implementar lo que se indica en el documento. Principalmente por esta razón tiene tanta importancia la fase de análisis de requisitos.

La norma IEEE 830-1998 es un estándar internacional para la obtención de una buena especificación de requisitos. Las características deseables para una buena especificación de requisitos software que se indican en el IEEE son las siguientes (15) (16)

- Correcta
- No ambigua
- Completa
- Verificable
- Consistente
- Clasificada
- Modificable
- Explorable
- Utilizable durante las tareas de mantenimiento y uso

1.4.2 Estándar Internacional ISO/IEC 9126-1

La norma ISO/IEC 9126 es un estándar internacional para la evaluación del software y está enfocada a la calidad del producto. El estándar está dividido en cuatro partes, las cuales dirigen, respectivamente, lo siguiente:

- Parte 1: Modelo de Calidad

- Parte 2: Métricas Externas
- Parte 3: Métricas Internas
- Parte 4: Métricas de Calidad en el Uso

La primera parte describe un modelo en dos secciones para calidad de los productos de software:

a) calidad interna y externa

b) calidad durante el uso

La primera sección del modelo especifica seis características para la calidad interna y externa, que son divididas en subcaracterísticas que se manifiestan externamente cuando el software se usa como una parte del sistema computarizado, y son un resultado de los atributos internos del software.

El modelo de calidad establecido en el estándar ISO/IEC 9126-1 clasifica la calidad del software en un conjunto estructurado de Características y Subcaracterísticas. (Véase la figura 2). Para obtener una descripción detallada, ver Anexo 1. Definiciones de cada característica y sub-características de calidad.

.

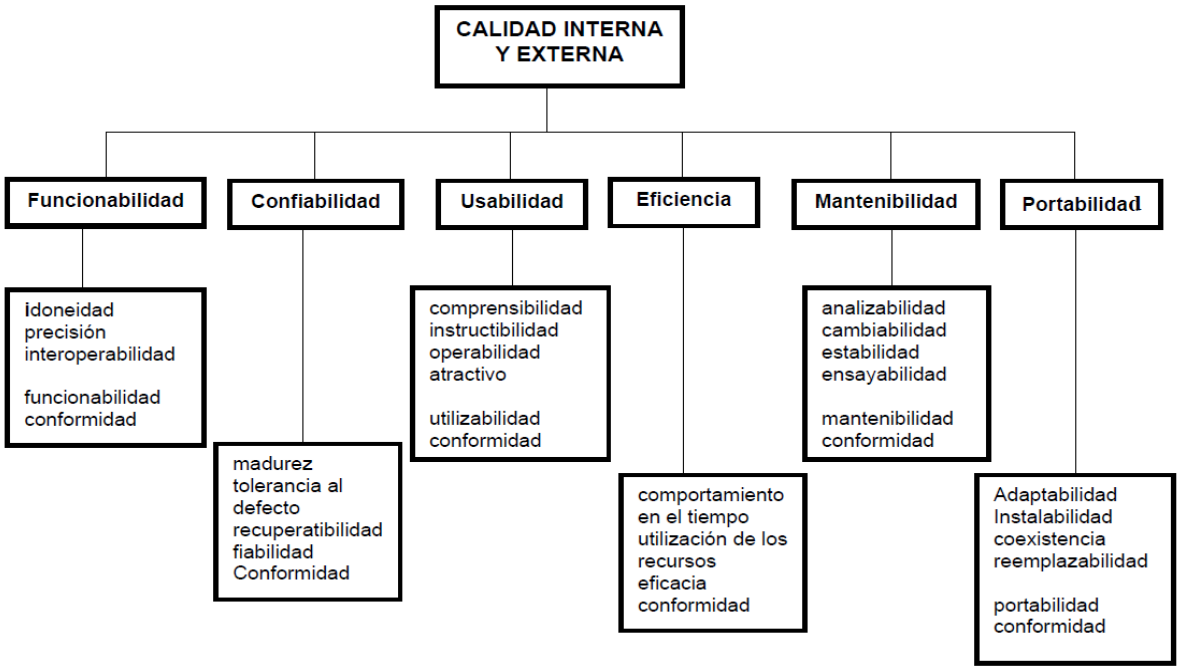


Figura 1. Modelo de calidad interna y externa.

Métricas internas

Las métricas internas pueden aplicarse a un producto de software no ejecutable (como una especificación o código fuente) durante el diseño y la programación. En el desarrollo de un producto de software los productos intermedios deben evaluarse usando las métricas internas que miden las propiedades intrínsecas, incluyendo las que pueden derivarse de un comportamiento simulado. El propósito primario de estas métricas internas es asegurar el logro de la calidad externa y la calidad durante el uso requerido. Las métricas internas constituyen una ventaja para los usuarios, evaluadores, verificadores, y diseñadores, pues le permiten evaluar la calidad de producto de software y atender los aspectos de la calidad desde las etapas más tempranas antes de que el producto de software devenga ejecutable. (17)

Métricas externas

Las métricas externas usan valores de un producto del software derivadas de las mediciones del comportamiento del sistema del que es parte, mediante el ensayo, la operación y observación del software o sistema ejecutable. Antes de adquirir o usar un producto del software, el mismo debe evaluarse usando métricas basadas en objetivos comerciales relacionados con el uso, explotación y gestión del producto en un ambiente organizativo y técnico especificado. Las métricas externas constituyen una ventaja para los usuarios, evaluadores, verificadores y diseñadores, pues le permiten evaluar la calidad del producto de software durante el ensayo o la operación. (17)

1.4.3 Estándar Internacional ISO/IEC 15504 (SPICE)

Software **P**rocess **I**mprovement **C**apability d**E**termination (SPICE) o Mejoramiento de Proceso de Software, es un modelo que surge de una iniciativa internacional y constituye una síntesis de varios modelos existentes. Tuvo dos años de prueba para ser ajustado en campo antes de ser publicado como estándar. El modelo de referencia se fundamenta en dos dimensiones bien determinadas y complementarias. Una de ellas determina los procesos a ser valorados, definiendo el proceso de vida del software, la otra dimensión presenta una escala para evaluar la capacidad. La primera dimensión, denominada dimensión del proceso, define un conjunto estándar de procesos para el ciclo de vida completo del software. Cada proceso se define desde el punto de vista de su finalidad y como un conjunto identificado de resultados.

La segunda dimensión o dimensión de la capacidad del proceso se sustenta en un conjunto de atributos que determinan el nivel. El objetivo de esta dimensión es definir la escala de medida para la capacidad del proceso.

El modelo agrupa a los procesos en cinco categorías (14):

Procesos Cliente - Proveedor (Customer - Supplier): esta categoría consiste en los procesos que directamente impactan al cliente, al soporte de desarrollo y a la transición del software al cliente.

Procesos de Ingeniería (Engineering): esta categoría consiste en los procesos que directamente especifican, implementan y mantienen un sistema, un producto de software y la documentación del usuario.

Procesos de Proyecto (Project): esta categoría consiste en los procesos establecidos dentro del proyecto, coordinación y administración de los recursos para producir un producto o proveer un servicio para satisfacer al cliente.

Procesos de Soporte (Support): esta categoría consiste en los procedimientos que establecen y soportan el desempeño de los otros procesos del proyecto.

Procesos de la Organización (Organization): esta categoría consiste en los procesos que establecen las metas de negocio de la organización, los procesos de desarrollo y recursos que ayudan a la organización a alcanzar dichas metas.

1.4.4 MOPROSOFT: Modelo de Procesos para la industria del Software

Moprosoft es el **Mo**delo de **Pro**cesos para la industria mexicana de **Soft**ware, realizado en conjunto por la Secretaría de Economía, la UNAM[6] y AMCIS[7]. Este modelo está diseñado para medir la capacidad de los procesos que siguen las empresas y para garantizar una calidad constante en el desarrollo y mantenimiento de software. Se tomaron los siguientes estándares internacionales como base para la creación de Moprosoft: ISO 9000, ISO 15504, SW-CMM y CMM-I.

Según la doctora Hanna Oktaba, el Modelo de Procesos para la Industria de Software, Moprosoft, tiene por objetivo proporcionar a la industria mexicana y a las áreas internas dedicadas al desarrollo y

[6] Universidad Nacional Autónoma de México.

[7] Asociación Mexicana para la Calidad en Ingeniería de Software.

mantenimiento de software, un conjunto integrado de las mejores prácticas basadas en los modelos y estándares reconocidos internacionalmente. (18)

Además, fomenta la estandarización de su operación a través de la incorporación de las mejores prácticas en gestión e ingeniería de software. La adopción del modelo permite elevar la capacidad de las organizaciones para ofrecer servicios con calidad y alcanzar niveles internacionales de competitividad.

Ventajas

- Fácil de entender.
- Fácil de aplicar.
- No es costoso en su adopción.

A decir de sus creadores, el modelo está orientado a pequeñas y medianas empresas, hecho favorable si se considera que aproximadamente el 80% de las empresas desarrolladoras de software del país caen en esta categoría. Su principal fortaleza es que integra varias de las prácticas propuestas por los otros modelos y corrige algunas de sus desventajas, como son el hecho de que no ha sido liberado por completo o al menos falta el modelo de evaluación; además, está en proceso de convertirse en norma compitiendo con el proyecto de norma ISO/IEC TR 15504, y aunque no ha sido probado, se planea realizar pilotos en algunas organizaciones para evaluar qué tan fácil resulta su implantación determinando los recursos necesarios. (19)

1.4.5 CMMI: Modelo Integrado de Madurez de las Capacidades

CMMI es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software. En 2002, se lanzó CMMI versión 1.1, luego en agosto de 2006 siguió la versión 1.2. El modelo para software (CMM-SW) establece 5 niveles de madurez para clasificar a las organizaciones en función de qué áreas de procesos consiguen sus objetivos y se gestionan con principios de ingeniería, es lo que se denomina un modelo escalonado o centrado en la madurez de la organización. Por otra parte, el modelo para ingeniería de sistemas (SE-CMM) establece 6 niveles de capacidad posibles para cada una de las 22 áreas de proceso implicadas en la ingeniería de sistemas. En el equipo de desarrollo de CMMI había defensores de ambos tipos de representaciones, por lo que el resultado fue la publicación del modelo con las dos representaciones: continua y escalonada.

La madurez de un proceso de software determina el grado en el cual este es explícitamente definido, administrado, medido, controlado y hecho efectivo. Además, es un indicador de la capacidad del proceso de software para lograr sus objetivos y resultados esperados. (20)

Los 5 niveles de madurez expuestos por CMMI son los siguientes:

1. **Inicial**
    - Estado inicial donde el desarrollo se basa en la heroicidad y responsabilidad de los individuos.
    - Los procedimientos son inexistentes o localizados en áreas concretas.
    - No existen plantillas definidas a nivel corporativo.
2. **Gestionado**
    - Se normalizan las buenas prácticas en el desarrollo de proyectos (en base a la experiencia y al método).
    - En este nivel consolidado, las buenas prácticas se mantienen en los momentos de estrés.
    - Están definidos los productos a realizar.
    - Se definen hitos para la revisión de los productos.
3. **Definido**
    - La organización entera participa en el proceso eficiente de proyecto de software.
    - Se conocen de antemano los procesos de construcción de software.
    - Existen métodos y plantillas bien definidas y documentados.
    - Los procesos no solo afectan a los equipos de desarrollo, sino a toda la organización relacionada.
    - Los proyectos se pueden definir cualitativamente.
4. **Cuantitativamente Gestionado**
    - Se puede seguir con indicadores numéricos (estadísticos) la evolución de los proyectos.
    - Las estadísticas son almacenadas para aprovechar su aportación en siguientes proyectos.
    - Los proyectos se pueden pedir cuantitativamente.
5. **Optimizado**
    - En base a criterios cuantitativos se pueden determinar las desviaciones más comunes y optimizar procesos.

- En los siguientes proyectos se produce una reducción de costes gracias a la anticipación de problemas y la continua revisión de procesos conflictivos.

Los 6 niveles definidos en CMMI para medir la capacidad de los procesos son:

0. **Incompleto**
    - El proceso no se realiza, o no se consiguen sus objetivos.
1. **Ejecutado**
    - El proceso se ejecuta y se logra su objetivo.
2. **Gestionado**
    - Además de ejecutarse, el proceso se planifica, se revisa y se evalúa para comprobar que cumple los requisitos.
3. **Definido**
    - Además de ser un proceso gestionado se ajusta a la política de procesos que existe en la organización, alineada con las directivas de la empresa.
4. **Cuantitativamente gestionado**
    - Además de ser un proceso definido se controla utilizando técnicas cuantitativas.
5. **Optimizando**
    - Además de ser un proceso cuantitativamente gestionado, de forma sistemática se revisa y modifica o cambia para adaptarlo a los objetivos del negocio. Mejora continua.

1.4.6 IPP-3520: 2009: Libro de Proceso para PPQA

La Universidad de las Ciencias Informáticas (UCI) es un centro productivo cuya misión es producir software y servicios informáticos a partir de la vinculación estudio – trabajo como modelo de formación. Actualmente se encuentra inmersa en un proceso de mejora con el objetivo de establecer una serie de actividades que conlleven a la obtención de la certificación para el nivel 2 de CMMI; hecho que la convertiría en la primera empresa cubana certificada con este modelo. En la actualidad, el centro está acometiendo un proyecto de mejora de sus procesos basado en el modelo CMMI y con la contratación de los servicios de consultoría del SIE *Center*(*Software Industry Excellence Center*) del Tecnológico de Monterrey.

Para dar cumplimiento a dicha meta se creó el Libro de Proceso para PPQA[8] basado en CMMI v 1.2 del SEI[9] y enfocado directamente en el área de proceso relativa a la calidad producto/proceso, situada en el nivel 2 en la categoría de Soporte. De forma general, PPQA (*Process and Product Quality Assurance*), tiene definida una guía de procesos para asegurar la calidad tanto del proceso como del producto. Su objetivo fundamental radica en proveer de una visión objetiva al personal y a la dirección de los procesos y productos asociados. (21)

CMMI es un modelo de referencia para el crecimiento de capacidades y madurez, que se enfoca tanto en procesos de Administración como de Ingeniería de Sistemas y Software. Con su instauración se espera alcanzar beneficios como:

- Calendarios y presupuestos predecibles en los proyectos.
- Mejora del ciclo de vida dentro del desarrollo de software.
- Mayor productividad.
- Mayor calidad de los productos y servicios que ofrece la universidad a sus clientes y por ende la satisfacción de los mismos.
- Mejorar la moral del personal que labora en el centro.

El servicio de consultoría que ofrece el SIE *Cente*r permite:

- Ayudar a la UCI a revisar su estrategia de mejora de procesos de software, para asegurar que su organización está basada en procesos y con un programa de mejora continua alineado con sus objetivos de negocio.
- Ayudar a la UCI a establecer las bases y fundamentos para seguir mejorando sus procesos y fortalecer su cultura de calidad en el desarrollo de software.
- Alinear los procesos de desarrollo de software con los principios y requisitos del modelo CMMI, estableciendo planes de mejora con los que la organización oriente sus procesos hacia la consecución de sus metas. (22)

[8] Process and Product Quality Assurance(Aseguramiento de la Calidad de Procesos y Productos)

[9] Software Engineering Institute

1.5 Metodologías de desarrollo de software

Para garantizar que el software sea de excelente calidad se deben tener en cuenta modelos muy bien definidos. La selección de los modelos a utilizar es de forma disímil. Un factor que influye en la toma de decisiones a la hora de seleccionar una metodología o marco de desarrollo de software, es la constante disputa que se establece entre quienes promueven el enfoque ágil versus aquellos que defienden la idea de que solamente a través de metodologías formales, como RUP[10], es que puede lograrse el éxito de un proyecto de desarrollo de software.

Según estudiosos del tema, como María A. Mendoza, existe una tendencia que se ha encargado de fusionar lo mejor de las metodologías tradicionales y ágiles, consolidando ciertas prácticas ágiles en procesos o marcos de trabajo considerados como formales. A pesar de lo anterior, lo más importante que toda organización o persona debe tener en cuenta es que no existe una metodología ideal, para cualquier escenario en la que se aplique la metodología de desarrollo que se seleccione, bien sea ágil o no, siempre dependerá directamente del equipo de trabajo, la cultura organizacional, lo cambiante del medio ambiente y la aceptación del usuario final.

En el proyecto SUIN, en acuerdo con el cliente (ver Proyecto Técnico), se definió el uso de tecnologías de Microsoft para el desarrollo, por lo que se inició el estudio de las metodologías nativas y marcos de desarrollo que se describen en los siguientes epígrafes.

1.5.1 MSF: Microsoft Solutions Framework

Microsoft Solutions Framework o Marco de Soluciones para Microsoft constituye un marco para el desarrollo de sistemas de software basado en principios, modelos, disciplinas, conceptos, prácticas y recomendaciones propias derivadas de la experiencia de Microsoft. Se autodefine como “marco” y no como metodología porque considera que no hay una única estructura de procesos válida para todos los proyectos. (23)

MSF propone una secuencia generalizada de actividades para la construcción de soluciones empresariales. Este proceso es flexible y se puede adaptar al diseño y desarrollo de una amplia gama de proyectos de una empresa. Además, está basado en fases, puntos de transición y de carga de forma

[10] Proceso Unificado de Rational

iterativa que se puede aplicar en el desarrollo de aplicaciones tradicionales, soluciones empresariales para comercio electrónico así como aplicaciones Web distribuidas. (24)

Características de MSF

- **Adaptable:** es usado en cualquier parte y no se limita a un equipo o proyecto específico.
- **Escalable:** puede organizar equipos pequeños entre 3 ó 4 personas, así como proyectos que requieren 50 personas o más.
- **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

Modelos de MSF

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

Modelo de Arquitectura del Proyecto

Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.

Modelo de Equipo

Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto.

Modelo de Proceso

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

Modelo de Gestión del Riesgo

Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir.

Modelo de Diseño de Proceso

Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo.

Modelo de Aplicación

Diseñado para mejorar el desarrollo, el mantenimiento y el soporte. Proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables y pueden ser usados en un solo ordenador o incluso en varios servidores. (25)

1.5.2 MSF Agile Software Development

MSF Agile (*Microsoft Solutions Framework Agile*) es la propuesta de Microsoft en el mundo de procesos y prácticas ágiles de desarrollo de software. “Microsoft Agile” renueva al MSF V3.0, que es un marco de trabajo en cascada y espiral, implementando las mejores prácticas del mundo de desarrollo ágil de software. MSF Agile se considera más una metodología que un marco de trabajo. Tiene como principales características el ser de planificación adaptable a los cambios y enfocado a las personas.

1.5.3 MSF for CMMI Process Improvement

Con la aparición del producto Microsoft Visual Studio Team System se ha actualizado MSF a la versión 4.0, produciendo dos variantes: *MSF for Agile Software Development* para el trabajo en entornos que emplean metodologías ágiles, y *MSF for CMMI Process Improvement* para el trabajo en entornos con el modelo CMMI.

MSF for CMMI es un proceso ágil de desarrollo de software, el cual cumple con los requerimientos para el nivel 3 de CMMI. La principal diferencia entre *MSF Agile* y *MSF for CMMI* es que este último está orientado a proyectos donde el nivel de formalidad es mucho mayor y se exige una cultura de mejoramiento continuo de procesos. Uno de los beneficios de implementar la metodología de MSF for CMMI es contar con una evaluación estándar por medio de la cual se puede validar la habilidad de desarrollar software en una organización. (26)

Dentro de los aspectos más interesantes acerca de esta metodología está el hecho de que ya viene integrada a la plataforma de desarrollo de *Microsoft Visual Studio Team System*. *Visual Studio Team System* (VSTS) es la solución Microsoft para gestionar el ciclo de vida completo de soluciones para la plataforma Windows. Por medio de VSTS es posible contar con una serie de plantillas y guías adaptadas

a cada metodología y orientadas a los roles definidos en cada una. Además, permite reducir la complejidad del proceso de desarrollo, mejorar la colaboración de todos los miembros del equipo, mejorar la productividad reduciendo los tiempos de desarrollo y pruebas, y gestionar los *workitems*[11].

En epígrafes anteriores se hacía referencia al modelo CMMI y a los niveles que lo componen. Es importante destacar que en el nivel 3 una de las áreas presente es la de Verificación y Validación, la cual desempeña un papel importante si de aseguramiento de la calidad se trata, ya que las actividades que se realizan en la misma están encaminadas a verificar y validar que se desarrolle el producto correcto y de la forma correcta.

1.6 Verificación y Validación

Pressman, citando a Barry Bohem plantea dos interrogantes que identifican claramente a la Verificación y a la Validación: (2)

- **Verificación:** ¿Estamos construyendo el producto correctamente?
- **Validación:** ¿Estamos construyendo el producto correcto?

Según el IEEE Std 729-1983[12] éstas se definen como:

- **Verificación:** Proceso de determinar si los productos de una cierta fase del desarrollo de software cumplen o no los requisitos establecidos durante la fase anterior.
- **Validación:** Proceso de evaluación del software al final del proceso de desarrollo para asegurar el cumplimiento de las necesidades del cliente. (27)

1.6.1 Objetivos del proceso de Verificación y Validación

La IEEE 1998b define que los objetivos de este proceso son (28):

1. Facilitar la temprana detección y corrección de errores en el software.
2. Mejorar la gestión de los riesgos del proceso y el producto.
3. Apoyo a los procesos del ciclo de vida del software para asegurar el cumplimiento con la ejecución del programa, cronograma, y presupuesto.

[11] Un *workitem* es un elemento de trabajo dentro de una metodología en concreto.

[12] Glosario Estándar IEEE de Terminología de Ingeniería del Software.

La norma IEEE 1012-1998[13] indica que: El propósito de V&V[14] es ayudar a la organización del desarrollo de software para construir con calidad durante el ciclo de vida de software. (28) La anterior definición es de mucha importancia para la investigación actual.

La norma 610.12[15]-1990 entre sus terminologías para la Ingeniería de Software trata la V&V como: El proceso de determinar si los requisitos para un sistema o componente son completos y correctos, los productos de cada fase de desarrollo cumplen con los requisitos o condiciones impuestas por la fase anterior, y el sistema o componente final cumple con los requisitos especificados. (7)

CMMI tiene 22 áreas de procesos clasificadas en cuatro categorías: Ingeniería, Gestión de Proyecto, Gestión de Procesos, Soporte, las cuales componen las dos representaciones que el modelo propone: la representación escalonada (madurez), donde se definen un conjunto de áreas por nivel de madurez y la representación continua (capacidad) en la cual se seleccionan las áreas a mejorar. Entre estas áreas de proceso se encuentran la Verificación y Validación.

La Validación demuestra que el producto cumplirá con el uso que tiene definido, mientras que la Verificación está encaminada a que los productos de trabajo reflejen los requisitos especificados. Las actividades de Validación usan enfoques similares a las de Verificación (pruebas, análisis, inspección). Frecuentemente los usuarios finales y otros interesados directos se involucran en las actividades de Validación. Ambas, Verificación y Validación a menudo fluyen concurrentemente y pueden usar porciones del mismo entorno.

1.7 Pruebas de software

Las pruebas constituyen uno de los métodos de evaluación presentes en el proceso de Verificación y Validación. Dentro de cada una de las etapas de desarrollo de un software las pruebas son fundamentales ya que a partir de ellas es posible controlar que los productos cumplan requisitos mínimos de operabilidad, además de garantizar la calidad de estos productos.

[13] Estándar para la Verificación y Validación de Software.

[14] Verificación y Validación.

[15] Norma IEEE 610.12 Glosario de Terminologías de Ingeniería de Software.

“Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.” (7)

De manera general las pruebas de software deben garantizar la calidad del producto desarrollado, y para lograr esto es necesario: (29)

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, creando componentes de pruebas ejecutables para automatizar las pruebas. Realizar las diferentes pruebas y manejar los resultados de cada una de forma sistemáticas. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados.

La prueba demuestra hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento. Además, los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan una buena indicación de la fiabilidad del software y, de alguna manera, indican la calidad del software como un todo.

1.7.1 Objetivo de las pruebas de software

Probar es el proceso de ejecutar un programa con el fin de encontrar errores o fallas.

Dentro de los objetivos fundamentales que se persiguen al aplicarles las pruebas a un software se encuentran los siguientes: (30)

- Un buen caso de prueba es aquel que tiene una alta probabilidad de descubrir un error no encontrado hasta entonces.
- Aumentar la calidad del producto final.

Los objetivos planteados anteriormente desacreditan la filosofía acerca de otorgarle éxito a una prueba cuando no se descubren errores, por el contrario, tiene éxito si encuentra un error no detectado hasta el momento.

Las pruebas no se realizan solamente al código también a la documentación y a la ayuda. “La prueba no puede asegurar la ausencia de defectos; solo puede demostrar que existen defectos en el software”. (2)

1.7.2 Principios de las pruebas

Las pruebas se rigen por una serie de principios que facilitarán el posterior uso de los métodos en un efectivo diseño de casos de prueba. (30)

- Hacer un seguimiento de las pruebas hasta los requisitos del cliente (trazabilidad).
- Deben ser planificadas antes de que empiecen.
- Plantear y diseñar las pruebas antes de generar ningún código.
- Empezar las pruebas en módulos individuales y avanzar hasta probar el sistema entero, empezar por “lo pequeño” y progresar hacia “lo grande”.
- No son posibles las pruebas exhaustivas.
- No deben realizarse planes de prueba suponiendo que prácticamente no hay defectos en los programas y, por tanto, dedicando pocos recursos a las pruebas. Se puede decir que los principios constituyen para las pruebas, requerimientos a seguir para certificar su éxito: detectar la mayor cantidad de errores.

1.7.3 Tareas a realizar para probar un software

Hay una serie de actividades que se deben realizar para conseguir la ejecución de las pruebas, las cuales se citan a continuación: (10)

Diseño de las pruebas: Comprende la identificación de la técnica o técnicas de pruebas que se utilizarán para probar el software. Distintas técnicas de prueba ejecutan diferentes criterios como guía para realizar las pruebas.

Generación de los casos de prueba: Consiste en la confección de los distintos casos de prueba según la técnica o técnicas identificadas previamente. La generación de cada caso de prueba debe ir acompañada del resultado que ha de producir el software al ejecutar dicho caso para detectar un posible fallo en el programa. Los casos de prueba determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba.

Definición de los procedimientos de la prueba: Conlleva a una especificación de cómo se va a llevar a cabo el proceso, quién lo va a realizar y cuándo se efectuará.
Ejecución de la prueba: Es el momento de aplicar los casos de prueba generados previamente e identificar los posibles fallos producidos al comparar los resultados esperados con los resultados obtenidos.

1.7.4 Importancia de probar

El desarrollo de software, como actividad humana al fin, no es perfecto, aún cuando el progreso sea sustancial. Las pruebas al software arrojan un panorama claro de las deficiencias detectadas en el sistema y de los riesgos que asocian a la organización, y aunque directamente no mejoran la calidad del producto, sí permiten que la administración tome medidas a la hora de la asignación de recursos para elevar la misma.

A través de las pruebas se puede comprobar que las especificaciones establecidas están siendo cumplidas con las funcionalidades que presenta el software, de igual manera que los datos que están siendo arrojados sean un indicador de la fiabilidad del sistema. La prueba de software es un elemento crítico e imprescindible para la garantía de la calidad, y de ahí la necesidad de aplicarla. (2)

Dentro de las causas que fundamentan las pruebas, las tres más sobresalientes son: (31)

Propensión a equivocarse: El ser humano es propenso a cometer equivocaciones que se manifiestan en diversos problemas contenidos en los modelos (defectos o faltas) y pueden manifestarse como fallas en tiempo de ejecución.

Fallas de hardware: La infraestructura empleada para el desarrollo de software (hardware, sistemas operativos, compiladores) no está exenta de fallas, lo que introduce defectos adicionales o permite que subsistan inadvertidos los que introdujo el desarrollador.

Creatividad del desarrollo: El desarrollo de software es una labor creativa y por ello es común que el producto desarrollado no coincida con el modelo contenido en las especificaciones.

1.7.5 Procedimiento de pruebas

Un procedimiento de prueba especifica cómo realizar uno o varios casos de prueba. Puede ser una instrucción para un individuo sobre cómo va a realizar un caso de prueba manualmente o puede ser una

especificación de cómo interaccionar manualmente con una herramienta de automatización de las pruebas para crear componentes ejecutables de pruebas. El cómo llevar a cabo un caso de prueba puede ser descrito por un procedimiento de prueba, pero es útil reutilizar a menudo un procedimiento de prueba para varios casos de prueba.

1.7.6 Estrategias de prueba

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir. Por lo tanto, cualquier estrategia de pruebas debe incorporar la planificación de la prueba, el diseño de los casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes.

Para aplicar pruebas al software se deben seguir un conjunto de estrategias para lograr que se hagan en el menor tiempo posible y con la calidad requerida, además de lograr que arrojen los resultados esperados.

Existen características que se pueden aplicar a todas las estrategias de pruebas, muchas de ellas definidas por Pressman. Dentro de las características generales de las estrategias de prueba se encuentran: (2)

- La prueba comienza en el nivel de módulo y trabaja "hacia fuera", hacia la integración completa del sistema completo.
- En diferentes puntos es adecuada la utilización de técnicas de prueba distintas.
- La prueba la lleva a cabo el que desarrolla el software y para grandes proyectos, un grupo de prueba independiente.
- La prueba y la depuración son actividades diferentes, pero la depuración puede entrar en cualquier estrategia de prueba.

Se puede concluir que en un proyecto la prueba en ocasiones requiere mayor esfuerzo que cualquier otra actividad de la Ingeniería de Software. Si se efectúa sin un plan estratégico, el tiempo se desaprovecha, el esfuerzo es consumido innecesariamente y en el peor de los casos, los errores inadvertidos quedarán sin

detectar, por tanto, puede ser muy conveniente establecer una estrategia sistemática para probar el software.

1.7.7 Niveles de prueba

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Cada una de las pruebas se realiza en determinados momentos del ciclo de vida del software. Teniendo en cuenta lo antes planteado, las pruebas se agrupan por niveles de acuerdo con las diferentes etapas del proceso de desarrollo.

Alfredo Weitzenfeld argumenta que existen tres niveles principales para aplicar las diferentes técnicas de pruebas (32):

Prueba de unidad: Mediante esta prueba solamente una unidad es probada como tal, como una clase, un paquete de servicio o un subsistema.

Prueba de integración: En ella se verifica que las unidades trabajen juntas correctamente. Ambas pueden ser realizadas mediante casos de uso de pruebas, los cuales pueden ser aplicados a clases, paquetes de servicio, subsistema o el sistema completo.

Prueba de sistema: Verifica el sistema completo o su aplicación como tal. Se toma el punto de vista del usuario final y los casos de uso de prueba ejecutan acciones típicas del usuario.

El mismo autor explica una serie de técnicas de pruebas:

Prueba de regresión: Tiene el propósito de verificar el sistema luego de haberle introducido cambios. Por ejemplo, después de corregir algún error, de manera que se mantenga la funcionalidad especificada originalmente.

Prueba de operación: Su objetivo es verificar el sistema en operación por un largo período bajo condiciones normales de uso.

Prueba de escala completa: Trata de verificar el sistema en su carga máxima mediante la asignación de los parámetros a su valor límite y la interconexión del sistema con un máximo de equipos y usuarios simultáneos. Su máxima expresión es la prueba de estrés, que significa que se prueba el sistema en los límites extremos para determinar su tolerancia si ocurre algún tipo de falla.

Prueba de rendimiento o de capacidad: Tiene como propósito medir la capacidad de procesamiento del sistema bajo diferentes cargas, incluyendo espacio de almacenamiento y utilización de la unidad de procesamiento.

Prueba de sobrecarga: Pretende observar cómo se comporta el sistema cuando se le aplica una sobrecarga, más allá de las pruebas de escala completa y rendimiento.

Prueba basada en requisitos o prueba de casos de uso: Intenta llevar a cabo pruebas basadas directamente en la especificación de requisitos. Se trata de verificar que el sistema final cumple con las especificaciones funcionales descritas para los caso de uso originales.

Prueba de documentación del usuario: Tiene como propósito probar la documentación de usuario, incluyendo la documentación de mantenimiento y servicio. Se prueba que los manuales y el comportamiento del sistema sean congruentes entre sí, que sean legibles, con una buena redacción y, en general, que sean comprensibles.

Prueba de aceptación de validación: Pretende lograr una revisión final por parte de la organización que solicitó el sistema, lo cual, a menudo significa validación del sistema. El sistema se prueba en su ambiente real por un período extenso. Cuando se termina la prueba se toma la decisión de aceptar o no el producto. Este tipo de pruebas es a veces conocido como prueba alfa. A menudo se hace una prueba beta, lo cual implica que el producto es probado por clientes seleccionados que utilizan el sistema y reportan fallas encontradas.

1.7.8 Métodos de prueba

La calidad de un sistema software es algo subjetivo que depende del contexto y del objeto que se pretenda conseguir. Para determinar dicho nivel de calidad se deben efectuar unas medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones iniciales del sistema. Al decir de Pressman, las pruebas de software son un elemento crítico para la garantía de la calidad del software, y representa una revisión final de las especificaciones, del diseño y de la codificación. (2)

Prueba de Caja Blanca

La prueba de caja blanca se centra en la estructura interna del programa para elegir los casos de prueba. Esta prueba, también llamada caja de cristal, ocupa un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para realizar sus casos de prueba. (33)

Según Pressman, mediante la prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que (2):

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

Prueba de Caja Negra

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro. (34)

La prueba de caja negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose solamente en los requisitos funcionales del sistema.

Muchos autores, como Pressman, consideran que estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

1.8 Conclusiones del capítulo

El aseguramiento de calidad es clave para el desarrollo de un software que sea capaz de satisfacer las expectativas y necesidades, tanto del cliente como de los usuarios. A partir del estudio del marco teórico

de la investigación se realizó un análisis comparativo tomando como punto de partida los distintos modelos de calidad existentes, así como las diversas metodologías de desarrollo de software que podrían servir de plataforma para el desarrollo del proyecto. (Ver Anexo 2. Comparación entre metodologías de desarrollo y Anexo 3. Estándares y modelos de calidad).

Se determinó que la Estrategia de Aseguramiento de la Calidad para el proyecto SUIN estará sustentada sobre la base de la metodología *MSF for CMMI* y el Libro de Proceso para PPQA (IPP-3520:2009). Dicha elección se fundamentó en la adopción de los subprocesos definidos por IPP-3520:2009 y a la adaptabilidad de las características que presenta la metodología al ámbito de desarrollo del proyecto, siendo la más importante de ellas la integración a la plataforma de desarrollo de Microsoft Visual Studio Team System, herramienta seleccionada para la creación de la aplicación. Se abordaron diferentes normas y modelos de calidad que guiarán el proceso de desarrollo de software, como por ejemplo la ISO/IEC 9126-1, que se selecciona ya que propone las características y subcaracterísticas de calidad a medir en un producto software, brindando la posibilidad de emplear y aplicar las métricas internas que la misma propone. Otra de las normas descrita fue la IEEE 830-1998, seleccionada debido a que constituye una guía para realizar una correcta especificación de requisitos de software. Además, uno de los epígrafes abordó acerca del proceso de pruebas, definiéndose los niveles y tipos de pruebas que existen, y se mencionaron además los principios de las pruebas.

CAPÍTULO II. SOLUCIÓN PROPUESTA

2.1 Introducción

El presente capítulo tiene como objetivo contribuir a la comprensión del entorno sobre el cual se trabaja en la investigación. Primeramente, se realiza una caracterización general del proyecto Sistema Único de Identificación Nacional, destacando la estructura organizacional del mismo. Seguidamente, se explican los aspectos que se tuvieron en consideración para adoptar como base para la elaboración de la estrategia el Libro de Procesos para PPQA y la selección de *MSF for CMMI* como metodología de desarrollo. Posteriormente, se aborda todo lo concerniente a la definición de la Estrategia de Aseguramiento de la Calidad, su estructura, actividades, tareas, herramientas y técnicas aplicadas en ella.

2.2 Características del proyecto SUIN

A partir de los estudios realizados durante el último período del 2008 e inicios del 2009 del estado actual del Sistema Nacional de Identificación de la población a nivel nacional e internacional, se propuso la modernización del sistema para garantizar la inscripción e identificación de las personas naturales, con procesos y documentos seguros. Con el objetivo de dar cumplimiento a los objetivos planteados se propone el desarrollo del Sistema Único de Identificación Nacional de la República de Cuba. Para su implementación se cuenta con la colaboración de la UCI, teniendo por tanto el apoyo de estudiantes (la mayoría cadetes insertados[16]) y profesores de la UCI.

El proyecto SUIN, para su desarrollo, está dividido en 7 módulos. El equipo está formado por 32 personas (7 profesores, 6 especialistas del MININT, 16 estudiantes de quinto año (10 cadetes) y 3 estudiantes de cuarto), los cuales están organizados en los diferentes grupos de trabajo: un grupo de analistas, un grupo de arquitectura, un grupo de bases de datos, un grupo de gestión de la configuración y un grupo de aseguramiento de la calidad. Cada rol tiene definido el flujo de trabajo que debe seguir para realizar sus responsabilidades acorde a la metodología y las actividades internas definidas. La siguiente figura representa la estructura organizativa existente en el mismo.

[16] Estudiantes miembros del MININT que se encuentran cursando estudios universitarios en universidades del país, en este caso, en la especialidad de ingeniero en ciencias informáticas en la UCI.

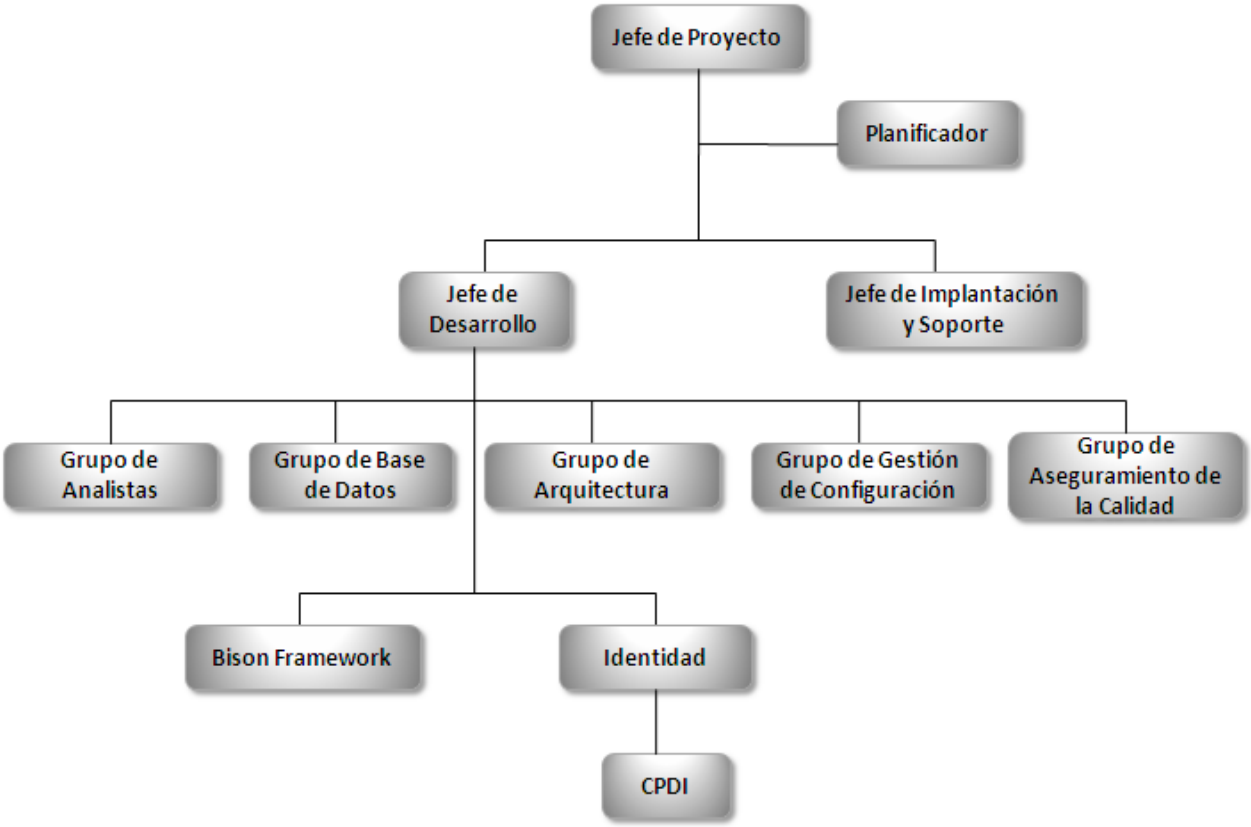


Figura 2. Organigrama del proyecto.

2.3 Adopción del Libro de Proceso para PPQA

Para adaptar el Libro de Proceso para PPQA (IPP[17]-3520:2009) a las características particulares del proyecto SUIN fue necesario tener en cuenta primeramente los procesos de desarrollo definidos en el mismo, así como las características de su esquema productivo. El proceso PPQA está compuesto a la vez por otros subprocesos que definen actividades encaminadas a asegurar la calidad del proceso y del producto, como es el caso del establecimiento de acciones correctivas destinadas a la obtención un producto capaz de satisfacer las expectativas del cliente.

[17] Integrated Product and Process.

Otro de los aspectos importantes por los cuales se decide adoptar como base de la solución propuesta lo establecido en el libro, lo constituye el hecho de estar centrado en una de las áreas de proceso de CMMI pertenecientes al nivel 2 y a que precisamente la metodología definida para guiar el desarrollo del proyecto describe el Modelo de Equipo de Microsoft para la estructuración de las personas y sus actividades en vista de alcanzar el éxito del proyecto. En la siguiente figura se presentan los grupos definidos en el modelo, donde cada uno tiene el mismo nivel de importancia dentro del desarrollo y contribución al proyecto.

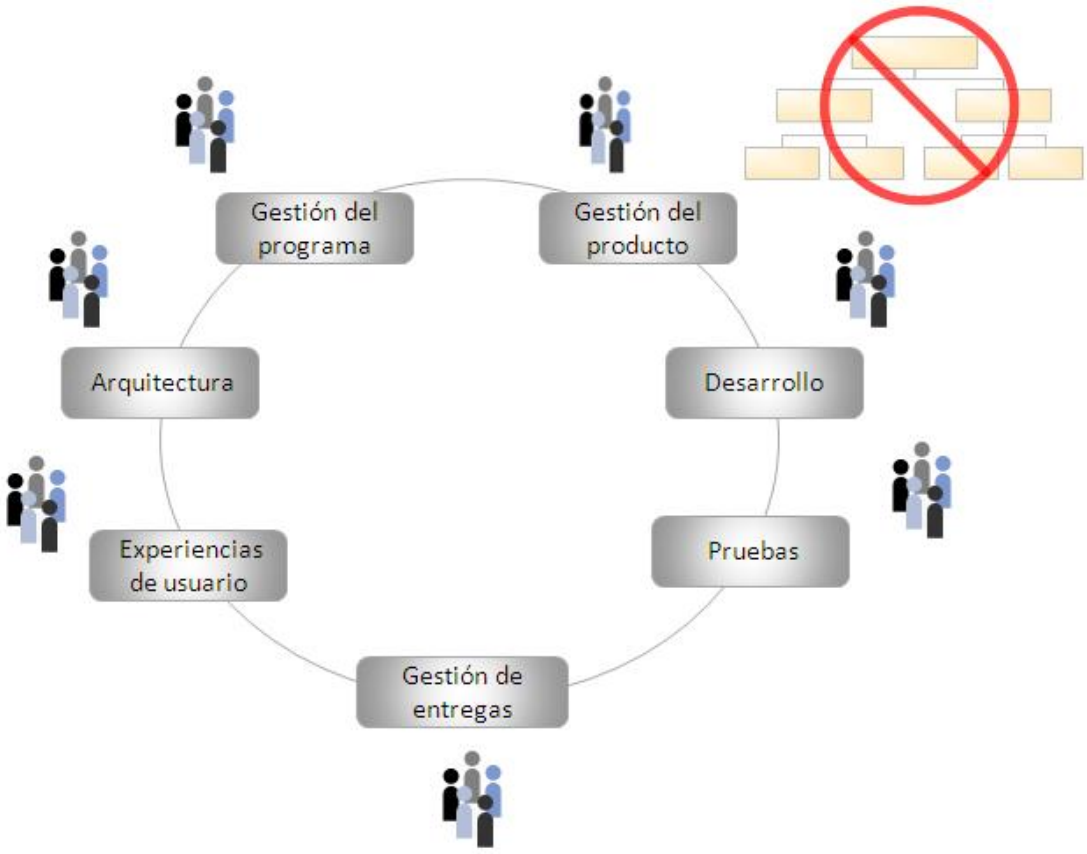


Figura 3. Modelo de Equipo según MSF for CMMI.

La utilización de este modelo permite establecer equipos de trabajo con comunicación abierta y responsabilidad compartida, donde cada rol es responsable de asegurar la calidad de una porción de la solución general. Además, la metodología propone la realización de actividades de mejora de procesos y

al igual que PPQA, plantea la realización de revisiones formales. A continuación se muestra la relación existente entre el proceso de PPQA y el ciclo de vida del software.

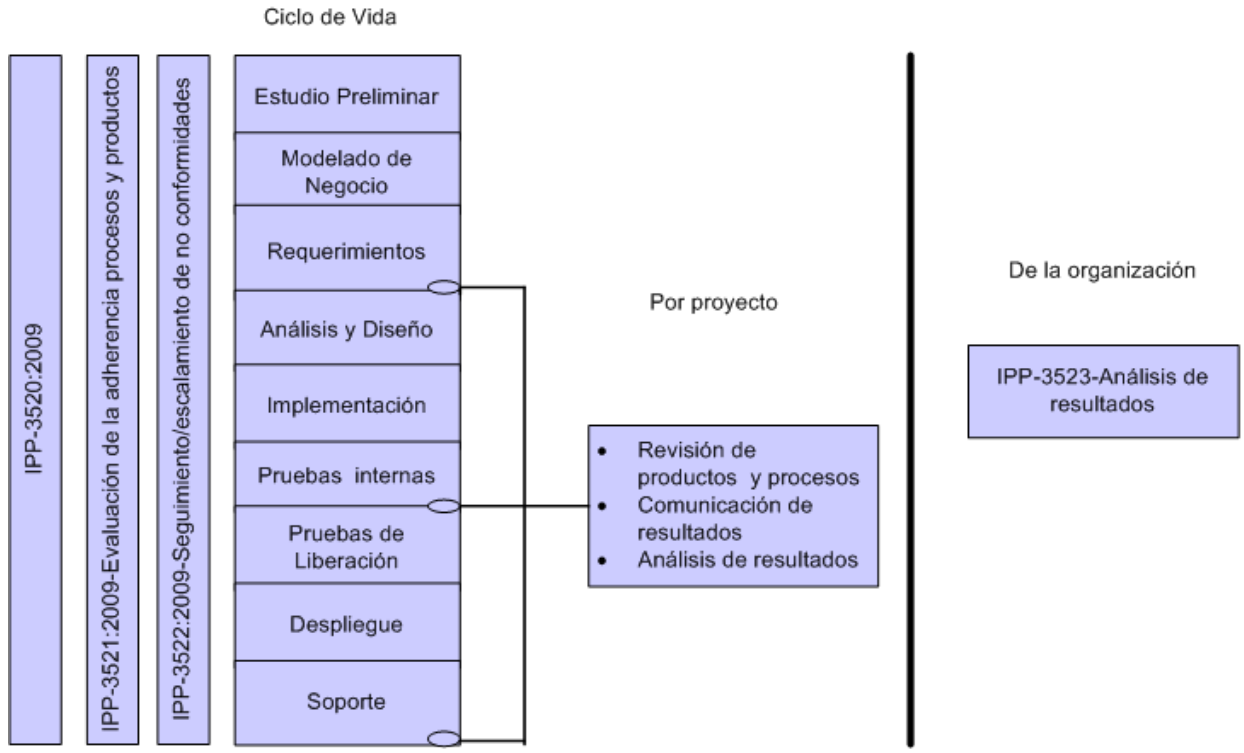


Figura 4. Relación del proceso de PPQA con el ciclo de vida del software.

Teniendo en cuenta las particularidades que presenta el ambiente de desarrollo definido para la ejecución del proyecto, se realizó una adecuación de las actividades propuestas en el Libro de Proceso para PPQA a las fases establecidas por la metodología *MSF for CMMI.* La siguiente figura expone las modificaciones acogidas para enmarcar la estrategia en el ámbito de desarrollo:

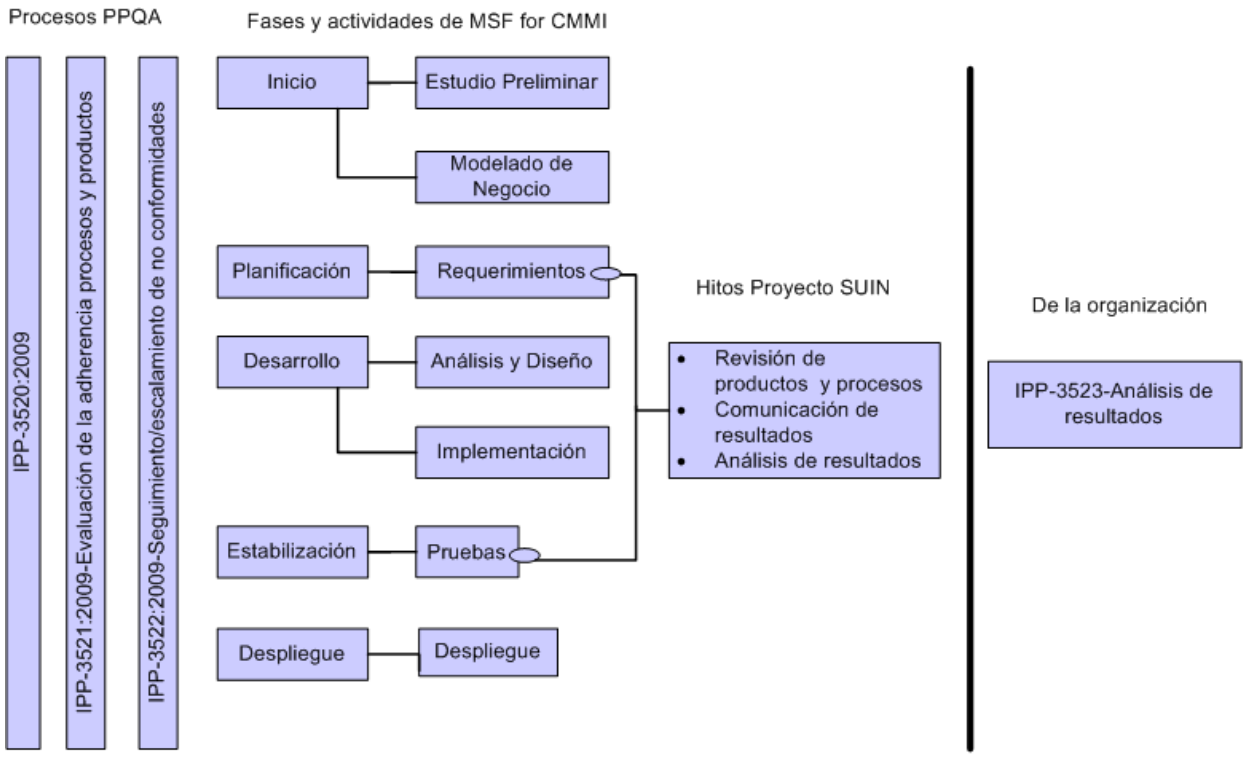


Figura 5. Adaptación del Libro de Proceso PPQA a la metodología MSF for CMMI.

Para obtener una visión más amplia del vínculo existente entre la metodología empleada, la base de la solución y la solución como tal, se realiza un mapeo entre ellas detallando los aspectos relevantes a tener presente para la obtención de una estrategia sólida. Los detalles se exponen en la siguiente tabla:

Procesos propuestos por PPQA	Fases de MSF for CMMI				
	Inicio	Planificación	Desarrollo	Estabilización	Despliegue
	Tareas propuestas				
Evaluación adherencia proceso/producto	- Revisar el Marco Legal. - Definir, revisar y aprobar el Proyecto Técnico.	- Revisar los procesos del negocio. - Analizar la trazabilidad de los requisitos. - Seleccionar proveedores válidos de requisitos. - Definir y revisar la lista de chequeo de requisitos. - Evaluar los requisitos de software. - Aplicar el Plan de Gestión de la Configuración. - Definir el Plan de Mediciones. - Identificar y seleccionar los riesgos críticos.	- Analizar la trazabilidad entre requisitos y diseño. - Definir y revisar lista de chequeo de interfaces. - Definir y revisar lista de chequeo de arquitectura. - Evaluar el diseño del software. - Revisar el diseño del flujo de procesos. - Revisar las interfaces. - Definir y revisar el Plan de pruebas de integración. - Definir y revisar el Plan de pruebas de	- Ejecutar y evaluar pruebas unitarias. - Ejecutar y evaluar pruebas de rendimiento. - Ejecutar y evaluar pruebas de estrés. - Ejecutar y evaluar pruebas de carga. - Ejecutar y evaluar pruebas de seguridad.	- Ejecutar pruebas unitarias a la base de datos. - Evaluar la documentación. - Ejecutar y evaluar pruebas de liberación. - Ejecutar y evaluar pruebas de aceptación.

		- Definir el Plan de Mitigación de Riesgos.	sistema. - Definir y revisar lista de chequeo de codificación. - Revisar el código fuente. - Definir los diseños de casos de prueba. - Revisar los diseños de casos de prueba. - Ejecutar y evaluar las pruebas unitarias. - Ejecutar y evaluar pruebas de integración. - Ejecutar y evaluar pruebas de sistema. - Ejecutar y evaluar pruebas de aceptación.		
		- Revisar los procesos de gestión de configuración del proyecto			
Seguimiento y	- Definir el Registro	- Registrar y actualizar las no conformidades en el Registro de Evaluaciones del proyecto.			

escalamiento de no conformidades	de Evaluaciones del proyecto.	- Proponer y registrar acciones correctivas.	
		- Verificar ejecución de acciones correctivas.	
		- Identificar no conformidades a escalar.	
Análisis de resultados	- Se realizan revisiones e inspecciones a los entregables obtenidos al finalizar cada iteración.		- Analizar los resultados de las pruebas de aceptación.
	- Publicar resultados de las revisiones.		

Tabla 2. Adaptación de las actividades propuestas por IPP-3520:2009 a las fases del proyecto SUIN.

2.4 Estrategia para el aseguramiento de la calidad en el SUIN

El aseguramiento de la calidad está comprendido dentro de la garantía de la calidad y está encaminado a obtener productos que sean capaces de satisfacer las necesidades del cliente. Para alcanzar dicha meta es preciso garantizar primero la calidad del proceso de desarrollo de software, ya que el mismo determina la obtención de un buen producto. Dentro del entorno de desarrollo del proyecto SUIN existen varios factores determinantes para el establecimiento de la estrategia, siendo el principal la participación de la Comunidad Técnica[18] en las actividades que se definan para asegurar el control tanto de los entregables que se generan durante las diferentes etapas de desarrollo, como de los procesos necesarios para la obtención de los mismos. Si bien es cierto que el empleo de metodologías contribuye a obtener la clave del éxito, no se puede pasar por alto que es necesario adaptarlas en función de las características del proyecto.

Se considera que una estrategia bien formulada ayuda a ordenar y asignar los recursos de una organización de una forma viable, basada en sus capacidades y carencias internas así como en la posible anticipación a los cambios del entorno. (35)

En el caso del proyecto SUIN, debido a la gran envergadura que presenta y a la importancia del mismo para la Dirección de Identificación y Registro (DIR), se hace preciso la definición de una estrategia de aseguramiento de la calidad. Con ella se pretende establecer qué hay que hacer, quién o quiénes deben ser responsables de ejecutar tareas que tributen a lograr y preservar la calidad, a través de qué métodos, procedimientos o herramientas se llevarán a cabo las tareas y cuándo se ejecutarán las mismas.

La estrategia estará compuesta por una serie de aspectos contemplados en cada una de las fases que propone la metodología *MSF for CMMI*[19] y en los procesos y subprocesos definidos en el Libro de Proceso para PPQA[20]. En la siguiente figura se representan las etapas que propone la metodología y los hitos a alcanzar en cada una de ellas:

[18] Es la encargada de construir el software, está integrada por analistas, programadores, entre otros.

[19] La utilización de MSF for CMMI se basó fundamentalmente en su integración con la herramienta de desarrollo seleccionada, Visual Studio Team System, y al hecho de proveer una serie de métodos y plantillas bien definidas y documentados.

[20] La aplicación del Libro de Proceso de PPQA se basó fundamentalmente por la definición de procesos que miden la adherencia tanto del producto como del proceso.

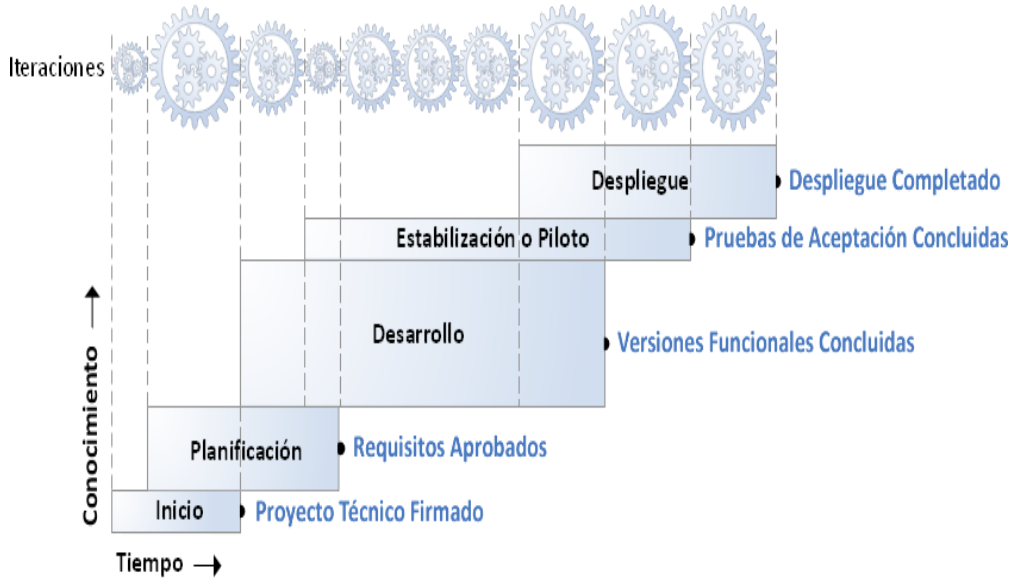


Figura 6. Relación entre fases de la metodología e hitos del Sistema Único de Identificación Nacional

Dicha metodología fue seleccionada debido a que presenta características relacionadas al enfoque iterativo y de trabajo en equipo, que a la vez la hacen compatible con la herramienta de desarrollo que se emplea en el proyecto, *Visual Studio Team System*. A continuación se citan algunas de ellas:

- Es posible contar con una serie de plantillas y guías adaptadas a ella y orientadas a los roles definidos en cada una.
- Organiza el proceso de desarrollo definiendo flujos de trabajo, roles y responsabilidades.
- Incrementa la transparencia en proyectos.

A partir del estudio teórico realizado y las características del proyecto SUIN, así como la definición de adoptar el Libro de Proceso PPQA, se crea la Estrategia de Aseguramiento de la Calidad, (Ver Anexo 4. Estrategia de Aseguramiento de la calidad SUIN) cuya vista general se muestra a continuación:

Figura 7. Vista general de la Estrategia de Aseguramiento de la Calidad SUIN

Es importante destacar que las tareas se enfocarán en la evaluación de los entregables del proyecto mediante el empleo de parámetros que miden consistencia, corrección, completitud, legibilidad, funcionalidad y exactitud. Estos términos se definen como:

- Consistencia:

El grado de uniformidad, estandarización y libertad de contradicciones entre la documentación o parte de un sistema o componente.

- Corrección:

El grado en que un sistema o componente está libre de fallas en su especificación, diseño e implementación.

El grado en que el software, documentación u otro producto cumplen con sus requerimientos específicos.

- Completitud:

Nivel o grado en que algo está completo.

- Legibilidad:

Calidad de la escritura que hace que sea fácil de leer y entender.

- Funcionalidad:

Es la capacidad del software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas cuando es usado bajo las condiciones especificadas.

- Exactitud

Capacidad del software para proporcionar efectos o resultados correctos convenidos con el grado de exactitud necesario.

La selección de los criterios antes expuestos estuvo dada a partir de las características presentadas en el estándar ISO/IEC 9126-1 relacionadas a las cualidades que debe poseer un producto de software, así como a las definidas por la norma IEEE 830-1998 en función de aplicarlas a la revisión de la documentación.

2.4.1 Estructura

A continuación se detallan los principales elementos que integran la Estrategia de Aseguramiento de la Calidad SUIN:

2.4.1.1 Propósito

En el propósito se especifican los objetivos para los cuales se crea la estrategia. (Anexo 4, Sección 1.1)

2.4.1.2 Alcance

Se especifican los proyectos con los cuales se involucra la estrategia. (Anexo 4, Sección 1.2)

2.4.1.3 Definiciones y acrónimos

Se definen los términos que pueden constituir un obstáculo para la comprensión de la estrategia. (Anexo 4, Sección 1.3)

2.4.1.4 Referencias

Se especifican los documentos a los cuales se hace referencia. (Anexo 4, Sección 1.4)

2.4.1.5 Objetivos de calidad

Se detallan los objetivos a cumplir en el proyecto en función de lograr asegurar la calidad. (Anexo 4, Sección 2)

2.4.1.6 Organización

Se especifica el capital humano con que cuenta el proyecto para su desarrollo. (Anexo 4, Sección 3.1)

2.4.1.7 Roles y responsabilidades

Acorde a la metodología establecida en el proyecto, los miembros del equipo de trabajo pueden asumir uno o más roles que a la vez ejecutan las tareas definidas en las actividades. (Anexo 4, Sección 3.2)
Estas actividades generan productos de trabajo (documentos, planes, código, cualquier salida tangible) que transitan por diferentes estados mientras se desarrollan.

2.4.1.8 Procesos de PPQA

Por cada fase propuesta por MSF for CMMI se establecieron una serie de tareas atendiendo a las que define dicha metodología y a las necesidades de calidad del proyecto. (Anexo 4, Sección 4)

Los miembros del equipo de desarrollo poseen usuarios que pertenecen a grupos con privilegios de seguridad acorde a los roles implementados. Debido a las características que presenta el proyecto y a la cantidad de personal involucrado, un mismo rol puede asumir diferentes responsabilidades dependiendo de la tarea en la cual participa. A continuación se muestra un ejemplo que avala el anterior planteamiento:

Tarea 1: Revisión de los procesos del negocio

Analista: lleva a cabo la comprensión de todos los procesos de negocio y los describe detalladamente especificando el flujo de actividades de cada uno de ellos, obteniendo los Diagramas del proceso y la DTA[21].

Tarea 2: Evaluación de requisitos del software

Analista: identifica y describe las funcionalidades de su módulo, verificando de forma constante su trabajo.

2.4.2 Actividades

Las actividades están agrupadas dentro de *workstreams*[22] y están asociadas a determinados roles. Las mismas están dirigidas a diferentes fases del desarrollo del software y para su culminación exitosa requieren del cumplimiento de una serie de tareas que las componen. Teniendo en cuenta la adopción del

[21] Descripción Textual de Actividades.

[22] Flujos de trabajo definidos por la metodología.

Libro de Proceso para PPQA, la estrategia contendrá algunas de las actividades definidas para PPQA, con los ajustes pertinentes para adaptarlas a las particularidades del proyecto. De esta manera se tienen:

Estudio preliminar

Se sientan las bases de todo el respaldo legal que debe tener el software. Además, se revisan los sistemas actuales y los marcos de referencia; se define tanto el modelo de desarrollo como el Proyecto Técnico.

Modelado del negocio

Se identifican y describen textualmente los procesos de negocio de la organización para obtener una mejor comprensión. El aseguramiento de la calidad está encaminado fundamentalmente a verificar y validar que los procesos de negocio identificados reflejan correctamente el funcionamiento de la organización y que fueron documentados adecuadamente.

Requerimientos

Se define qué debe hacer el sistema, por lo que se identifican los requerimientos funcionales y no funcionales del software y se desarrollan las interfaces. El aseguramiento de la calidad se enfoca en verificar la completitud, corrección y consistencia de los requisitos. La descripción de los requisitos de software se realiza teniendo en cuenta los criterios establecidos en el estándar IEEE 830-1998.

Análisis y Diseño

Se describe cómo el sistema será realizado a partir de las funcionalidades previstas y las restricciones impuestas (requerimientos), por lo que se indica con precisión lo que se debe programar. Los requerimientos son transformados en arquitectura. El diseño incluye base de datos (Diccionario de datos), interfaces de usuario, servicios. Es la actividad que guía el análisis y diseño de la solución. La estrategia está encaminada en esta etapa a verificar la ausencia de errores en la transformación de los requisitos en diseño.

Implementación

Constituye la transformación del diseño en código fuente y estructura de base de datos. Se implementan las clases significativas para la arquitectura. Involucra la adición de nuevas funcionalidades a la forma arquitectónica del sistema. Una vez concluida la tarea de desarrollo, debe ser probada a nivel de unidad, el código debe ser analizado e integrado al código base existente.

Pruebas

El objetivo de esta actividad es verificar que los requerimientos de software han sido satisfechos a través de pruebas de unidad, integración, sistema, carga y aceptación. Teniendo en cuenta las características de la aplicación a desarrollar, la cual es un sistema de uso a nivel nacional, se hace necesaria la realización de pruebas:

- **Unitarias:** las pruebas unitarias son una herramienta muy importante para el personal de pruebas pero, sobre todo, para los desarrolladores. Aunque el tamaño de la "unidad" que se prueba puede variar, Visual Studio Team System Test incluye un conjunto de herramientas de prueba perfectamente integradas con Visual Studio que permite generar pruebas unitarias específicas para métodos, incluidos los métodos privados.
- **Web:** una prueba Web, también llamada prueba Web declarativa, está compuesta por una serie de solicitudes HTTP. Funcionan en la capa de protocolo emitiendo solicitudes HTTP. Las pruebas Web sirven para probar la funcionalidad de aplicaciones Web y para probar las aplicaciones Web bajo carga. Se utilizan en pruebas de rendimiento y en pruebas de carga excesiva.
- **De carga:** las pruebas de carga pueden contener tanto pruebas Web como pruebas unitarias. El propósito principal es simular el acceso de muchos usuarios al mismo tiempo a un servidor. Las pruebas de carga contienen uno o varios escenarios que se utilizan para modelar cómo interactúa un grupo de usuarios con una aplicación de servidor.
- **De sistema:** es la prueba que se ejecuta cuando el sistema está funcionando como un todo. Su objetivo radica en detectar la ausencia de funcionalidades o errores presentes que pueden comprometer la calidad del software.
- **De integración:** se realizan con el objetivo de combinar y probar múltiples componentes juntos. construyendo una estructura de programa que esté de acuerdo con lo que dicta el diseño.
- **De aceptación:** es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Administración de la configuración del software

Está dirigida a la configuración necesaria para el proceso de desarrollo de software. Se describen las políticas, estrategias y tareas que deben ejecutarse durante el ciclo de vida del proyecto. Se identifican y definen los elementos en el sistema, controlando los cambios de estos a lo largo de su ciclo de vida, registrando y reportando el estado de los elementos y las solicitudes de cambio, y verificando que estén completos y que sean los correctos.

Gestión operacional

La planificación del proyecto es el proceso de definir las tareas secuenciales:

- ¿Qué es lo que se va a hacer?
- ¿Cómo y cuándo se va a hacer?
- ¿Quién lo va a hacer?

En esta disciplina se conoce suficiente acerca de los riesgos y posible negocio como resultado del proyecto, lo que posibilita una decisión con conocimiento acerca de si se continúa con el proyecto o se abandona.

2.4.1 La tarea SQA

La tarea constituye el componente más importante de la Estrategia de Aseguramiento de la Calidad. Está orientada a evaluar un producto de trabajo determinado dentro del desarrollo. A continuación se propone una estructura que cumple con lo establecido por *MSF for CMMI* y que presenta otros elementos incorporados a raíz del trabajo que se desempeña en el proyecto.

Figura 8. Estructura de la Tarea SQA

En la *descripción* se especifica claramente la manera en que se evaluará el producto de trabajo, atendiendo a los parámetros de corrección, completitud, consistencia, exactitud y legibilidad para los cuales son definidos los criterios a tener en cuenta.

En las *herramientas y técnicas* se especifica la (las) herramienta(s) que se utilizará(n) para darle cumplimiento a la tarea, así como el modo de aplicación (técnica) dentro del proyecto. Para la selección de las mismas se tuvieron en cuenta las características y necesidades del ambiente productivo en el que se llevan a cabo.

En las *entradas* se detallan los artefactos que son necesarios para la ejecución de la tarea y en las *salidas* se citan los elementos obtenidos después de llevar a cabo la tarea.

La selección de las tareas a incluir en la estrategia se realizó teniendo en cuenta las que propone la metodología *MSF for CMMI* con las variaciones requeridas para lograr una correcta adaptación al entorno del SUIN. La fase de Inicio estuvo enfocada en la realización de un estudio preliminar de los sistemas actuales, procesos del negocio y del Marco Legal del software, motivo por el cual las tareas de

aseguramiento de la calidad estuvieron dirigidas principalmente a la aprobación del Proyecto Técnico y comprensión de las leyes y normativas establecidas por Cuba relacionadas a la identificación y registro de la población. Las siguientes tareas constituyen un ejemplo de las planteadas en la estrategia:

- **Tarea: Revisión del Marco Legal.**

Descripción

Se evalúa el ajuste al Marco Legal del software en cuanto a corrección y completitud.

Corrección:

Verificar que se exprese correctamente la correspondencia entre los procesos definidos y los artículos de las leyes, normativas y reglamentos establecidos para el registro de la población.

Completitud:

Verificar que están definidos todos los procesos de acuerdo a las restricciones definidas en las leyes de la República de Cuba.

Herramientas y técnicas

Se aplica un *walkthrough*[23] que será desarrollado en una o varias reuniones de trabajo entre el equipo de analistas, un miembro del equipo de calidad y el Líder de proyecto, donde realizarán la evaluación del Marco Legal en cuanto a corrección y completitud.

Roles y responsabilidades

Analista: realiza la exposición en el *walkthrough.*

Líder de proyecto: participa en el *walkthrough.*

Responsable de calidad: participa en el *walkthrough.*

Entradas

Leyes, normativas y reglamentos vigentes del Sistema de Identificación y Registro de la población de la República de Cuba.

Marco Legal

DTA

Salidas

Minuta de cada reunión realizada.

[23] Es una forma de revisión organizada y dirigida por el responsable del elemento a analizar. Su objetivo es familiarizar a la audiencia con el contexto que se analiza.

- **Tarea: Definición, revisión y aprobación del Proyecto Técnico**

Descripción

Se define el Proyecto Técnico con el propósito de establecer el alcance desde el punto de vista técnico y conceptual para la ejecución del proyecto. Se revisa atendiendo a completitud.

Completitud: verificar que estén definidos los elementos a tener en cuenta para la formulación de la propuesta, que se identifiquen los problemas actuales, los criterios de éxito, los posibles riesgos, las responsabilidades de las partes involucradas en el proyecto(MININT y UCI), así como la arquitectura base propuesta.

Herramientas y técnicas

Se desarrollaron reuniones con la participación del cliente y los responsables del proyecto en las cuales se definieron las metas que se persiguen con la creación del sistema a partir del marco de referencia brindado y se planteó la solución propuesta. Las revisiones fueron llevadas a cabo por miembros del Centro de Identificación y Seguridad Digital.

Roles y responsabilidades

Líder de proyecto: presenta una propuesta del Proyecto Técnico y corrige los señalamientos detectados por el cliente.

Especialista del Centro de Identificación y Seguridad Digital: revisa el Proyecto Técnico.

Cliente: aprueba el Proyecto Técnico.

Entradas

Versión de Proyecto Técnico

Salidas

Minuta de Reunión Chequeo Proyecto Técnico

Minuta de Reunión Discusión del Proyecto Técnico

Minuta de Reunión con la DTS[24]

Proyecto Técnico aprobado

Aspectos como la revisión del basamento legal requieren de revisiones minuciosas, y teniendo en consideración que se trata de un proyecto como el SUIN, que automatiza procesos vitales referentes a la identificación y registro de la población y que a la vez tiene un respaldo legal importante, aumenta el valor

[24] Dirección de Tecnología y Sistemas del MININT

de la revisión. De no lograrse el total cumplimiento de las leyes no se estará construyendo el sistema idóneo.

Se definen tareas que son realizadas en distintas actividades teniendo en cuenta la necesidad de verificar el mantenimiento de la relación existente entre los elementos de una actividad en particular y sus predecesoras. Tal es el caso de la tarea Análisis de la trazabilidad en la actividad Requerimientos, la cual verifica la relación existente los procesos del negocio definidos en el Modelo de negocio y los requisitos funcionales identificados en la ERS[25]. A continuación se profundiza en la descripción de la misma.

- **Tarea: Análisis de la trazabilidad**

Descripción

Se verifica la relación existente entre los procesos de negocio definidos, los requisitos funcionales identificados y el modelo conceptual en cuanto a corrección, completitud y consistencia.

Corrección:

Verificar que las relaciones existentes entre los procesos, requisitos y modelo conceptual sean correctas.

Completitud:

Verificar que cada requisito funcional descrito es traceable con alguno de los procesos de negocio identificados.

Consistente:

Verificar que las relaciones existentes entre los procesos de negocio y los requisitos funcionales están especificadas correctamente.

Herramientas y técnicas

Se aplican revisiones técnicas de acuerdo a la descripción de las tareas.

Roles y responsabilidades

Administrador de la calidad: realiza la revisión.

Entradas

ERS

DTA

Salidas

Análisis de la trazabilidad

[25] Especificación de Requisitos de Software

Cuando se refiere a la implementación se plantean varias tareas a desarrollar, siendo la revisión del código una de las fundamentales. Seguidamente se especifica en qué consiste la misma.

- **Tarea: Revisión del código**

Descripción

La revisión del código se usa para asegurar que el código nuevo o modificado se adapta a las consideraciones de calidad establecidas, entre ellas estándares de codificación, conformidad a la arquitectura y el diseño, legibilidad y seguridad. Se harán revisiones en base a:

Corrección

El código no debe contener errores. Los errores de corrección se pueden referir a dos aspectos. Defectos de "escritura", es decir, lo que habitualmente se conoce por "programa que no funciona". Por ejemplo, bucles infinitos, variable definida de un tipo pero utilizada de otro, contador que se sale de las dimensiones de un array[26]. Defectos con respecto al diseño: el diseño no realiza lo que el diseño establece.

Se tendrán en cuenta aspectos como:

- Verificar que los nombres de las clases y los métodos estén propiamente definidos para representar la funcionalidad del segmento del código.
- Verificar que el código cumple con los estándares definidos en el proyecto.

Herramientas y técnicas

Se realiza un *peer review*[27] por parte del implementador del código que se revisa junto a otro programador de su equipo de desarrollo para evaluar los criterios de corrección, consistencia y completitud. Se aplica una inspección al código por parte del arquitecto de software.

Roles y responsabilidades

Desarrolladores: generan y efectúan la revisión del código atendiendo a lo establecido en la lista de chequeo de codificación.

Entradas

Código fuente

Lista de chequeo de codificación

[26] En programación se considera un medio para guardar un conjunto de objetos de la misma clase.

[27] Es una técnica en la cual un trabajo se somete al escrutinio de uno o más expertos en el área y se emite una evaluación del trabajo que incluye sugerencias sobre cómo mejorarlo.

Salidas

Lista de chequeo de codificación (llena).

2.4.2 Herramientas y técnicas aplicadas

La aplicación de herramientas y técnicas en el proyecto está relacionada con las características de las tareas del desarrollo SUIN. Tanto para la realización de verificaciones como de validaciones se pueden utilizar distintos tipos de técnicas. En general, estas técnicas se agrupan en dos categorías:

- ***Técnicas de Evaluación Estática:*** buscan faltas sobre el sistema en reposo, es decir, estudian los distintos modelos que componen el sistema software buscando posibles faltas en los mismos. Así pues, estas técnicas se pueden aplicar tanto a requisitos como a modelos de diseño y código.
- ***Técnicas de Evaluación Dinámica:*** generan entradas al sistema con el objetivo de detectar fallos, cuando el sistema ejecuta dichas entradas. Los fallos se observan cuando se detectan incongruencias entre la salida esperada y la salida real. La aplicación de técnicas dinámicas es también conocida como pruebas de software y se aplican generalmente sobre código.

Las revisiones, como parte de la evaluación estática, pretenden detectar manualmente defectos en cualquier producto del desarrollo y son aplicadas en toda su totalidad.

En las **revisiones formales** los participantes son responsables de la fiabilidad de la evaluación, y generan un informe que refleja el acto de la revisión. Además, se realiza ***peer review*** por parte de los analistas e implementadores que intercambian el trabajo en busca de errores. Los ***walkthrough****,* guiados por analistas, diseñadores y probadores, son aplicados para la obtención y revisión de artefactos fundamentales en actividades como Inicio, Diseño y Pruebas. A continuación se muestra una tarea que aplica la técnica de *peer review:*

- **Tarea: Definición y revisión del Plan de pruebas de sistema.**

Descripción

Se genera el Plan de pruebas de sistema y se valida que el plan satisface los siguientes criterios:

- Las pruebas programadas cubren los requerimientos del software.
- El uso apropiado de estándares y métodos de prueba.
- Conformidad con los resultados esperados.
- Capacidad de ser implementado y mantenido.

Herramientas y técnicas

El Administrador de la calidad, en función de responsable de pruebas de sistema, junto al Líder de proyecto, realiza un *peer review* durante la generación del Plan de pruebas de sistema. Este trabajo permite que en paralelo con la obtención del Plan se ejecute la revisión del mismo.

Roles y responsabilidades

Administrador de calidad: participa en el *peer review.*

Líder de proyecto: aprueba el Plan de pruebas de sistema.

Entradas

ERS

Salidas

Plan de pruebas de sistema

Las pruebas, como técnica dinámica, tienen un peso importante dentro de la estrategia, por lo que se dedica una actividad completa. La generación y verificación de los planes, diseño, procedimientos y casos de prueba se hace en diferentes momentos del desarrollo anteriores a la actividad de Pruebas, que es donde se ponen en práctica, con la excepción de las pruebas unitarias que se ejecutan en la actividad de Implementación.

La estrategia incluye pruebas de unidad para la capa de acceso a datos y de negocio, pruebas de integración, pruebas de sistema, pruebas de aceptación y pruebas de liberación. Para las pruebas de sistema se propone la realización de pruebas de funcionalidad, rendimiento, carga y estrés. Para la prueba de integración se especifica el enfoque que tiene su ejecución en el SUIN, que al ser un sistema en el cual el diseño de la arquitectura obliga al desarrollo integrado, esta tiene el objetivo de verificar que al ejecutar una determinada acción se actualicen consistentemente todas sus dependencias dentro del sistema. A continuación la especificación de la tarea correspondiente:

- **Tarea: Definición y revisión del Plan de pruebas de integración.**

Descripción

Se define el Plan de pruebas de integración con el objetivo de evaluar el comportamiento del software como un todo, teniendo en cuenta que de la correcta integración entre los módulos depende que no existan inconsistencias en los datos registrados en el sistema. Se verifican los resultados contra los criterios de aceptación para la prueba. Se valida que el plan satisface los siguientes criterios:

- Traceable a los requerimientos del software.
- Consistencia externa con los requerimientos del software.
- Consistencia interna.
- Las pruebas programadas cubren los requerimientos del software.
- Conformidad con los resultados esperados.
- Capacidad para ser implementado y mantenido acorde a las necesidades del usuario.

Herramientas y técnicas

Se realizan un *peer review* entre los responsables de calidad mientras se genera el plan, permitiendo al unísono su revisión.

Roles y responsabilidades

Administrador de calidad: elabora el plan.

Arquitecto de software: revisa el plan verificando que se han incluido los componentes a tener en cuenta para ejecutar la prueba.

Entradas

ERS

Salidas

Plan de pruebas de integración

2.5 Conclusiones del capítulo

El presente capítulo inicialmente presentan las particularidades del proyecto SUIN. Seguidamente se explican los elementos a tener en cuenta para adoptar como base para la estrategia el Libro de Proceso para PPQA, así como la metodología de desarrollo *MSF for CMMI* adaptándolas a las características de desarrollo del proyecto SUIN. Se realizó una descripción de la estrategia propuesta, desglosándose la estructura de la misma y detallando minuciosamente cada acápite que la compone. En base a lo expuesto anteriormente se explicó en qué consiste la Actividad y la Tarea. De esta última se realizó un esquema que representa los aspectos a tomar en consideración para la ejecución de la misma, enfatizando en las herramientas y técnicas a emplear para su cumplimiento.

CAPÍTULO III. APLICACIÓN DE LA PROPUESTA

3.1 Introducción

En los capítulos precedentes de este trabajo investigativo se explicaron las definiciones que sirven de base para la comprensión de los objetivos que se persiguen con el mismo. Además, se detalló el diseño de una estrategia para asegurar la calidad del proyecto Sistema Único de Identificación Nacional de la República de Cuba en aras de lograr su ejecución exitosa. El objetivo de este capítulo es describir el modo de aplicación de la estrategia diseñada con anterioridad y exponer los resultados obtenidos a partir de la ejecución de las tareas desempeñadas. Se tendrán en cuenta los procesos de prueba y revisión efectuados durante las diversas fases por la que ha transitado el desarrollo del software. Para la realización de dichas pruebas se hará uso de *Microsoft Visual Studio Team System Test*, el cual facilita la realización de pruebas unitarias, de carga, manuales y web.

3.2 Aplicación de la estrategia propuesta

La elaboración de una estrategia implica un proceso de planificación que culmina en un plan general con misiones organizativas, metas, objetivos básicos a desarrollar en determinado plazo con recursos mínimos y los métodos que aseguren el cumplimiento de dichas metas. Tomando como base el planteamiento anterior se puede decir que en el proyecto Sistema Único de Identificación Nacional de la República de Cuba (SUIN) se acometieron una serie de actividades planteadas en la Estrategia de Aseguramiento de la Calidad propuesta con anterioridad, las cuales estaban encaminadas a mejorar tanto la calidad del proceso como la del producto. Es importante aclarar que la aplicación de dicha estrategia se realizó inicialmente hasta la primera iteración de desarrollo del software, permitiendo el refinamiento de los procesos existentes y la toma decisiones con un enfoque de mejora continua que permita limar defectos y limitantes para posteriores iteraciones.

Por cada fase de desarrollo fueron establecidos hitos cuyo cumplimiento fue evaluado a través de la ejecución de tareas de aseguramiento de la calidad en los distintos módulos que integran el SUIN. Los resultados obtenidos fueron analizados y los errores detectados fueron objeto de seguimiento hasta su total eliminación.

A inicios del proyecto se confeccionó el Plan de proyecto con el propósito de definir la planificación para el desarrollo y manejo de los cambios que surjan durante el ciclo de vida del software. Como parte del plan se elaboraron una serie de cronogramas relativos a la Fase de Inicio y a la Captura de Requisitos con el fin de tener definidas y documentadas las tareas a realizar, los responsables de llevarlas a cabo y las fechas de inicio y fin de las mismas para verificar su cumplimiento. Para obtener una información profunda acerca de las tareas efectuadas en la Captura de Requisitos, ver Anexo 5. Cronograma perteneciente a la captura de requisitos. A continuación se muestra una figura que representa una vista parcial del cronograma correspondiente a la Fase Inicio:

	Nombre de tarea	Duration	Start	Finish
1	**Proyecto**	**120 days**	**Mon 5/18/09**	**Fri 10/30/09**
2	**Fase Estudio Preliminar**	**120 days**	**Mon 5/18/09**	**Fri 10/30/09**
3	Reunión de inicio	1 day	Mon 5/18/09	Mon 5/18/09
4	Revisión de Marco Legal	1 day	Tue 5/19/09	Tue 5/19/09
5	Revisión de Manuales de Procesos/Trabajo	2 days	Wed 5/20/09	Thu 5/21/09
6	Revisión de Sistemas Actuales	2 days	Wed 5/20/09	Thu 5/21/09
7	**Elaborar Proyecto Técnico**	**33 days**	**Mon 5/18/09**	**Wed 7/1/09**
8	Establecer marco de referencia	1 day	Mon 5/18/09	Mon 5/18/09
9	Caracterizar entorno de aplicación	2 days	Mon 5/18/09	Tue 5/19/09
10	Definir la solución y requisitos técnicos	15 days	Wed 5/20/09	Tue 6/9/09
11	Definir arquitectura de la solución (Aproximación Técnic	7 days	Wed 6/10/09	Thu 6/18/09
12	Definir modelo de desarrollo	5 days	Fri 6/19/09	Thu 6/25/09
13	Identificación de riesgos(de cara al cliente)	3 days	Fri 6/26/09	Tue 6/30/09
14	Definir organización del Proyecto	1 day	Wed 7/1/09	Wed 7/1/09
15	Actualizar estimaciones de tamaño, tiempo, esfuerzo y costo	1 day	Thu 7/2/09	Thu 7/2/09
16	Revisión y Aprobación del Proyecto Técnico por la UCI	5 days	Fri 7/3/09	Thu 7/9/09
17	Corregir Proyecto Técnico	2 days	Fri 7/10/09	Mon 7/13/09
18	Revisión y Aprobación del Proyecto Técnico con el cliente	79 days	Tue 7/14/09	Fri 10/30/09
19	Planear y Asignar recursos-roles al proyecto (formación del equip	1 day	Mon 5/18/09	Mon 5/18/09

Figura 9. Cronograma perteneciente a la Fase Inicio.

En los siguientes epígrafes se detalla cada acción ejecutada en las fases comprendidas en la primera iteración, especificando la actividad a la cual se está haciendo alusión.

3.2.1 Fase Inicio

Durante este período las actividades de aseguramiento de la calidad se encuentran encaminadas fundamentalmente a la revisión, corrección y aprobación del Proyecto Técnico tanto por la UCI como por el cliente, así como a la identificación de los riesgos desde el punto de vista de los interesados. Es importante resaltar que el objetivo de dicho documento es establecer el alcance desde el punto de vista técnico y conceptual para la ejecución del proyecto y que está dirigido a las direcciones del MININT involucradas en el desarrollo del proyecto propuesto (DTS y DIR[28]). Además, contiene la recopilación de la información que influirá directamente en la solución a desarrollar.

La aplicación de la estrategia se inicia con la realización de un estudio preliminar de los sistemas actuales y del marco legal, así como la definición del Proyecto Técnico, el cual fue sometido a un proceso de revisión por parte de la comisión revisora de la Dirección General de Producción de la UCI, hasta su aprobación final. A partir de esta se inicia el mismo proceso de revisión y corrección de cara al cliente hasta su firma. Una vez validado dicho documento por el cliente y establecidos los elementos relativos a la organización del proyecto, se procede a efectuar las actividades pertenecientes a la Fase de Planificación e ir transitando por las restantes fases a lo largo del ciclo de vida del proyecto hasta su culminación. El Sistema Único de Identificación Nacional de la República de Cuba estará enfocado en los procesos que se realizan en las oficinas de trámites de Carné de Identidad y Registro de Población, así como aquellos que son desempeñados en el Centro de Personalización de Documentos de Identificación. Para una mejor comprensión se recomienda ver el Anexo 6. Módulos que integran el Sistema Único de Identificación Nacional.

3.2.2 Fase Planificación

En esta fase el hito fundamental radica en la aprobación de los requisitos de software por parte del cliente. En ella el analista desempeña un papel fundamental, ya que siempre debe llegar a conocer la temática y el problema a resolver y dominarlo hasta el ámbito que el futuro sistema a desarrollar lo abarque. Por ello, el analista debe tener alta capacidad para comprender problemas de muy diversas áreas o disciplinas de trabajo. Se identifican los requisitos funcionales y no funcionales pertenecientes a cada uno de los módulos que integran el SUIN, y en base a ellos se elaboran las Especificaciones de Requisitos de Software (ERS). Durante este período se definen también las actividades relacionadas a la Gestión de la

[28] Dirección de Identificación y Registros del MININT.

configuración de software, haciendo énfasis, principalmente, en la forma de identificar los artefactos, entregables y versiones de los documentos, siendo los responsables de calidad los encargados de velar por la correcta denominación y estructura de los mismos. No obstante, como parte de la configuración se tienen en cuenta la definición de las líneas base para cada fase. Además, en esta etapa se planifican las pruebas a ejecutar y se diseñan los casos de pruebas que después serán empleados para verificar el cumplimiento con los requisitos.

Actividad: Requerimientos

Si la evaluación en general es difícil, la de los requisitos en particular lo es más debido a que lo que se evalúa es la definición del problema. En el proyecto SUIN se definió un Plan de administración de requisitos (Ver Expediente de Proyecto, ID-REQ101-2009 Plan de administración de requisitos) donde se establecen los elementos base a tener en cuenta para su gestión, como son los atributos (estado, beneficio, esfuerzo, estabilidad) y aspectos de trazabilidad para el seguimiento a los mismos durante todo el ciclo de vida del proyecto.

Para la definición de los requerimientos funcionales se contó con la información proporcionada por proveedores que fueron seleccionados tomando en cuenta criterios de aceptación como conocimiento del mercado, conocimiento del negocio y su interacción con el sistema. (Ver Expediente de Proyecto DIR-REQ102-2009- Proveedores de Requisitos DIR.) Además, se analizó la relación de cada requisito con los procesos de negocio identificados a fin de garantizar el cumplimiento de los objetivos del sistema. En función de la adopción de elementos del proceso de mejora y del enfoque a procesos, se ajustó la forma de realizar la especificación de los requisitos, la cual fue validada por el Grupo de normalización y métricas de Calisoft. También es importante citar que con el propósito de establecer la trazabilidad de los requisitos de forma general, la cual incluye el análisis de requisitos versus requisitos, requisitos versus responsable, requisitos versus implementación y requisitos versus pruebas, se empleó el *Team System*, componente que viene integrado a *Visual Studio* y a través del cual se puede conocer el impacto que tendría un cambio en un requerimiento en el resto de las actividades vinculadas a él, así como el encargado de actualizarlo.

Como se indica en la tarea “Evaluación de requisitos del software”, se procedió a efectuar una revisión profunda de la documentación correspondiente a cada requerimiento empleando como técnica de evaluación una lista de chequeo confeccionada tomando en cuenta los aspectos definidos por la

metodología, la semántica y la estructura del documento (Ver Anexo 7. Lista de chequeo de Requisitos de Software) Al final del proceso de revisión fue detectado un número considerable de no conformidades, 223 para ser más exactos, pertenecientes a cada uno de los distintos módulos. En el siguiente gráfico se desglosa dicha cifra atendiendo a los módulos en que fueron encontradas:

Figura 10. Tabulación de revisión interna.

Una vez corregidos los errores detectados se integraron cada una de las ERS en un documento general que fue revisado por Calisoft y el cual fue devuelto con 0 no conformidades. Posteriormente, se procedió a la liberación y validación de los requisitos con el cliente mediante varias sesiones de trabajo en las que se presentaron los prototipos no funcionales del sistema. La aplicación de dicha técnica contribuyó al intercambio de ideas entre analistas e interesados, logrando obtener una visión más clara de las necesidades reales del cliente.

3.2.3 Fase Desarrollo

El hito a lograr en esta etapa lo constituye la obtención de las versiones funcionales del sistema. En función de ello cada uno de los módulos fue sometido a un proceso de revisión individual en el cual se enfatizó en cada una de las diversas funcionalidades, así como en el ajuste a las pautas de diseño y a los

estándares de codificación. Además, es importante destacar que la arquitectura fue otro aspecto a medir sobre la cual se tuvieron en cuenta detalles concernientes a cada una de las capas (Procesos y Servicios, Negocio, Base de Datos y Acceso a Datos).

En el proceso de revisión estuvieron presente el líder de proyecto, arquitecto, responsables de calidad, analista principal y desarrolladores, los cuales, mediante el empleo de listas de chequeo encontraron deficiencias que fueron registradas en el documento Registro de no conformidades de Diseño en vista a su posterior solución (Ver Anexo 8. Registro de no conformidades de Diseño). En la siguiente figura se exponen los resultados generales obtenidos a partir de la revisión realizada a los módulos Administración, Supervisión, Captura de Datos, Recepción, Entrega y Reportes.

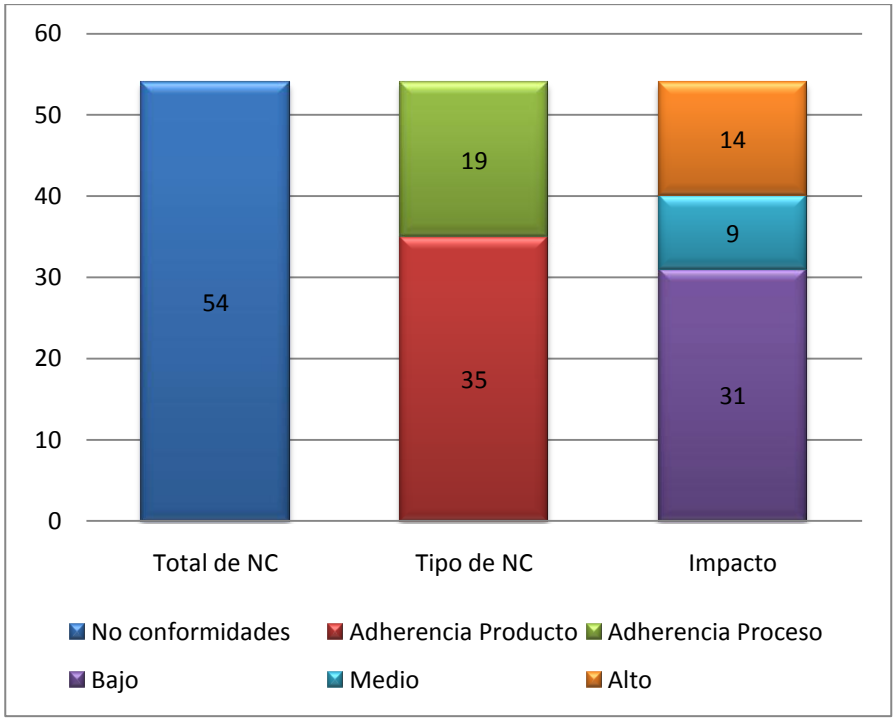


Figura 11. Reporte de no conformidades emitido por el proyecto.

A partir de los errores detectados se realizó una tabulación de los mismos indicando el porcentaje de no conformidades encontradas por módulos. Ver la siguiente figura:

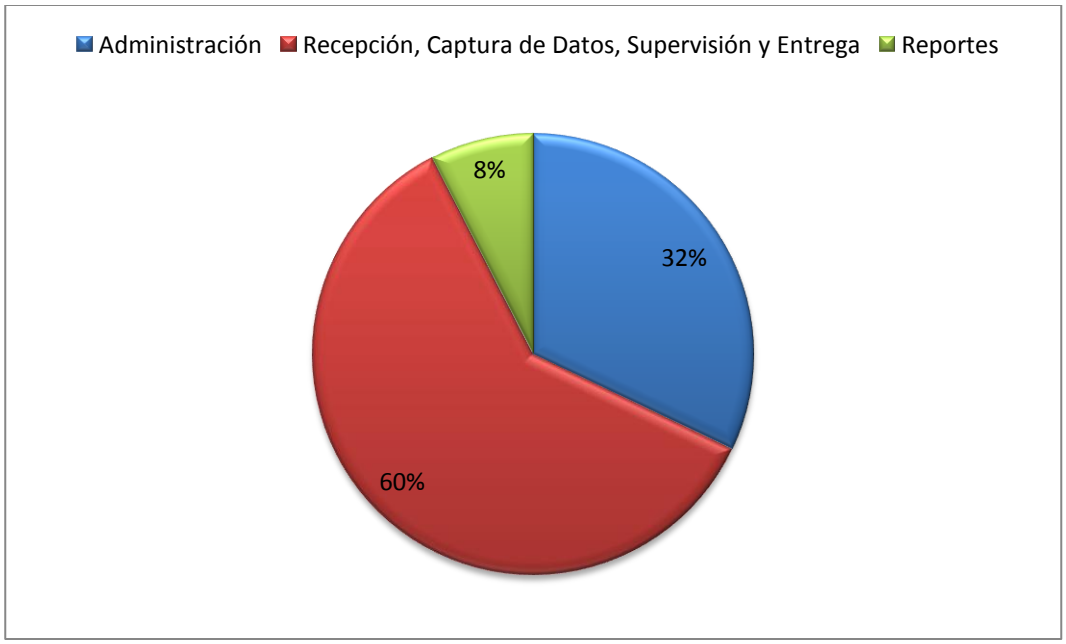


Figura 12. Porcentaje de las no conformidades detectadas.

Actividad: Análisis y diseño

Al igual que la evaluación de requisitos, la evaluación de diseño es crucial, ya que los defectos de diseño que queden y sean transmitidos al código, cuando sean detectados en fases más avanzadas del desarrollo, o incluso durante el uso, implicará un rediseño del sistema, con la subsiguiente recodificación.

Con respecto al diseño, los objetivos principales de su evaluación estática son:

- Determinar si la solución elegida es la mejor de todas las opciones; es decir, si es la óptima y la forma más fácil de realizar el trabajo.
- Determinar si la solución satisface todos los requisitos descritos en la especificación; es decir, si la solución elegida realizará la función encomendada al software.

Se procedió a efectuar una revisión de las interfaces de usuario, las cuales constituyen un elemento determinante para la aceptación por parte del cliente y usuarios respecto al sistema, de ahí el énfasis que se hizo en dicho elemento. Es importante destacar que inicialmente se definió en el proyecto un manual de pautas de diseño, al igual que uno de codificación. Para la verificación del ajuste a los mismos se elaboraron listas de chequeo, entre ellas la de interfaces (Ver Anexo 9. Lista de chequeo de Interfaces) que permitió revisar tanto las dimensiones así como el posicionamiento de diversos componentes (cajas

de texto, botones).Tras la aplicación de la lista de chequeo se obtuvieron resultados que indicaron que existían ciertos problemas vinculados principalmente al desaprovechamiento del área de trabajo, es decir, varias interfaces con amplios espacios brindaban la posibilidad de reubicar otros controles.

También se revisó todo lo concerniente al diseño la arquitectura para el desarrollo del proyecto. Cada módulo que integra el SUIN está diseñado siguiendo un patrón de capas bien definidas y diseñadas para reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas, por lo que se realizaron en verificaciones de los permisos, de la correspondencia entre los procesos y requerimientos, así como de la visualización de la información. Para obtener más detalles de los elementos analizados, ver Anexo 10. Lista de chequeo de Arquitectura.

Actividad: Implementación

Durante esta actividad se lleva a código fuente, en el lenguaje de programación elegido, todo lo diseñado en la fase anterior. Esta tarea la realiza el desarrollador siguiendo por completo los lineamientos impuestos en el diseño y en consideración siempre a los requisitos funcionales y no funcionales (ERS) especificados con anterioridad.

Mientras se programa la aplicación, sistema, o software en general, se realizan también tareas de refinación, esto es la labor de ir liberando al código de los errores factibles de ser hallados en esta fase, formando parte de las técnicas de aseguramiento de la calidad. Es necesario realizar pruebas unitarias, normalmente con datos de prueba, aunque es obvio que todos los errores no serán detectados en una sola actividad. La aparición de algún error funcional (mala respuesta a los requerimientos) eventualmente puede llevar a retornar a la fase de diseño antes de continuar la implementación.

De manera similar a la revisión de las interfaces, se elaboró una lista de chequeo que permitió revisar el código y verificar el ajuste a los estándares de codificación previamente definidos en el proyecto (Ver Anexo 11. Lista de chequeo de Codificación). El aseguramiento de la calidad estuvo enfocado en la obtención de uniformidad en el código respecto a la declaración de las clases, las interfaces, métodos, atributos y otros elementos.

- **Actividad: Pruebas**

Las pruebas se pueden definir como un elemento primordial en la búsqueda de sistemas que cumplan con lo establecido en sus especificaciones de requisitos. Para su ejecución deben ser planificadas

primeramente, pero se hace mucho más necesaria la especificación de definiciones de varios aspectos dentro de los cuales se encuentra la configuración del ambiente de pruebas en el caso de las pruebas de sistema, de rendimiento o de carga. En actividades anteriores se definieron los tipos de prueba a realizar y se diseñaron los casos de prueba basados en requisitos que serían usados durante la ejecución de pruebas de sistema. Con el objetivo de asegurar la aplicación correcta de los casos de prueba se definió una lista de chequeo para revisar la inclusión de los aspectos relevantes a tener en cuenta. (Ver Anexo 12. Lista de chequeo de Diseño de Caso de Prueba). Inicialmente se realizaron pruebas unitarias por parte de los desarrolladores con el objetivo de detectar los errores existentes en los elementos testeables más pequeños.

En el proyecto SUIN fue creado un ambiente de pruebas con características semejantes al ambiente real de funcionamiento para efectuar las revisiones a cada versión funcional generada. Con vista a la obtención de las versiones funcionales fue precisa la ejecución de pruebas de integración para asegurar que los componentes en el modelo de implementación operaran correctamente cuando se combinaran para ejecutar un requisito. Dichas pruebas fueron aplicadas para verificar el correcto funcionamiento del sistema a partir de la unión de los módulos Recepción, Captura de datos, Supervisión y Entrega en el momento de efectuar procesos que requieren que la información o datos utilizados por unos sean los mismos a manejar o devolver por otros; por ejemplo: cuando se realiza la captura de datos para cierto trámite, el nombre, número de carné de identidad y demás elementos, deben mantenerse invariables para la supervisión y entrega del documento final.

La siguiente imagen muestra una de las interfaces de la aplicación:

Figura 13. Interfaz de una versión funcional del sistema

Como parte de las pruebas se revisaron los módulos Administración, Recepción, Captura de Datos, Reportes, Supervisión y Entrega, siendo sometidos a 3 revisiones como parte del proceso de pruebas. Posteriormente, fueron sumados al proceso de revisión el Centro de Personalización de Documentos y el módulo para el control de insumos. A partir de las no conformidades detectadas, a cada implementador del requisito en cuestión se le asignaron tareas de resolución de defectos a través del *Team System*,

siendo posible establecer el seguimiento y verificación del cumplimiento de las mismas. Ver la siguiente figura en la que se visualiza el entorno de asignación de tareas:

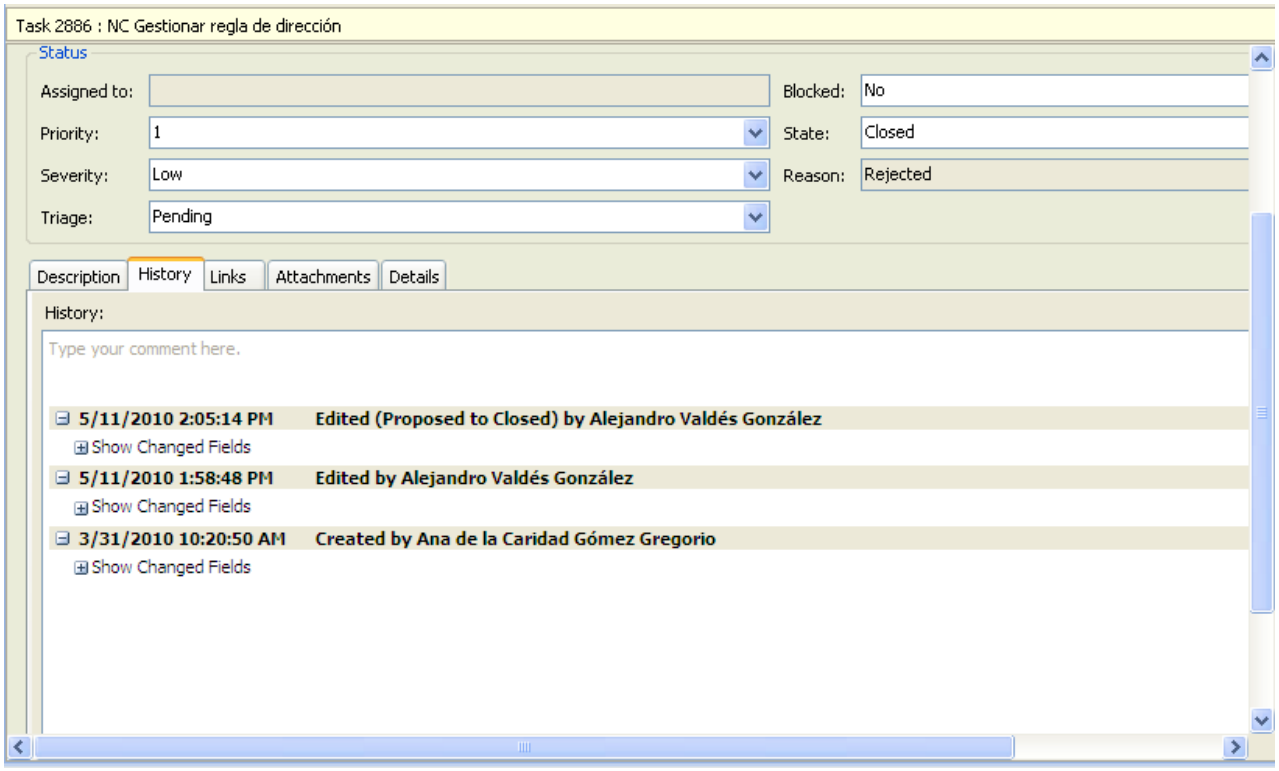


Figura 14. Asignación y seguimiento de no conformidades.

Las pruebas de carga se realizan con el objetivo de probar el rendimiento de las funcionalidades de una solución y comparar valores respecto a diferentes entradas buscando diferencias que no deben existir. Fueron aplicadas al *framework*[29] creado para el desarrollo de la aplicación. Se emplearon 25 usuarios concurrentes como variables de entrada para la realización de las pruebas de carga. Para la aplicación del procedimiento de pruebas de rendimiento diseñado al software se identificó un entorno con las siguientes características:

[29] Conjunto de clases modeladas de forma general para resolver problemas relacionados en un contexto específico.

Hardware	Datos
Microprocesador	Intel(R) Xeon(R) CPU 3.00 GHz (Servidores) Intel(R) Core(TM) 2 Duo CPU E7300 a 2.66 GHz(PC Clientes)
Memoria RAM	2 GB (PC Clientes) 3 GB (Servidores)
Ancho de Banda	100 Mbps
Disco Duro	320 GB(PC Clientes) 80 GB (Servidores)

Figura 15. Especificaciones de hardware

En el siguiente epígrafe se exponen los resultados arrojados a partir de las pruebas realizadas. Es importante destacar que las pruebas sirvieron también para determinar cuáles eran los módulos que se encontraban más atrasados en el cumplimiento del plan establecido para la iteración.

3.3 Publicación y análisis de los resultados

Hoy en día el conocimiento es el activo más valioso en una empresa. Su gestión implica en gran medida catalogar, consultar, acceder, buscar e integrar información y documentación que en la mayoría de las empresas se encuentra dispersa y en soportes no informatizados. Por este motivo la gestión de la documentación se ha convertido en una necesidad para las organizaciones. (36)

En el proyecto se empleó como herramienta para la gestión documental el SharePoint, creándose un portal que permitió a los miembros del equipo de proyecto, en dependencia de los permisos otorgados, acceder a la documentación almacenada en él. Dentro de las facilidades que aporta dicho portal se pueden citar:

- Establecimiento de la comunicación y colaboración entre los miembros del proyecto permitiendo compartir anuncios, tareas, calendarios y planes de proyectos, además de la creación de documentos en grupo.
- Su estructura de portal electrónico garantiza una mejor organización del contenido.

- Las áreas del portal son editables y es posible limitar el acceso de usuarios a áreas determinadas.
- La publicación de los reportes logra un mayor control de lo que pueden ver los usuarios y da la oportunidad de hacer accesible la información a un mayor número de personas en el proyecto para mejorar la toma de decisiones.

Uno de los elementos expuestos en el portal fue el Expediente de Proyecto, cuya estructura se muestra en el Anexo 13. Portal creado para la gestión documental del proyecto.

Mediante el SharePoint fue posible la publicación de los resultados obtenidos durante las revisiones y auditorías realizadas en la primera iteración de desarrollo.

De forma general, las pruebas realizadas permitieron la detección y solución de no conformidades. Los resultados obtenidos durante la primera iteración fueron tabulados en las gráficas que se muestran a continuación:

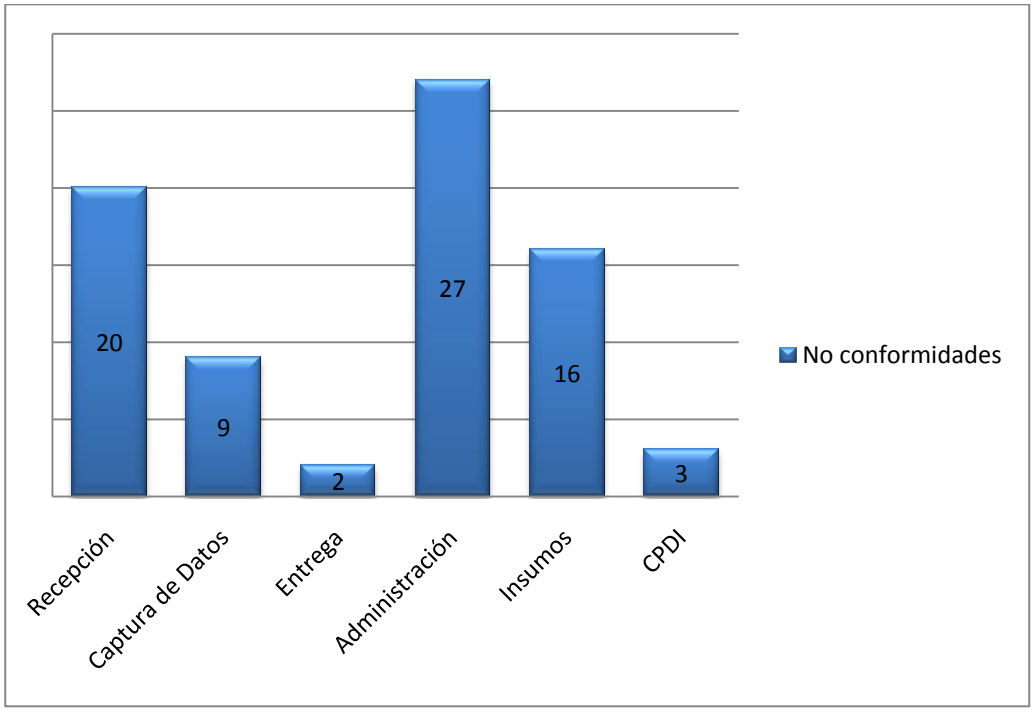


Figura 16. Resultados por módulos de la primera revisión del sistema (Iteración 1)

En la segunda iteración se tuvo en cuenta la corrección de los errores identificados en la anterior, aquellos a los cuales no se les había dado solución, así como los que surgieron. Como resultado de la misma se

llevó a cabo un análisis del total de no conformidades presente en la aplicación, resultado que permitió determinar que a pesar de darle solución a un número considerable de NC[30] todavía existían algunas que debían ser limadas. En la siguiente figura se puede apreciar la diferencia entre cada una de las revisiones efectuadas demostrándose el avance alcanzado hasta la última:

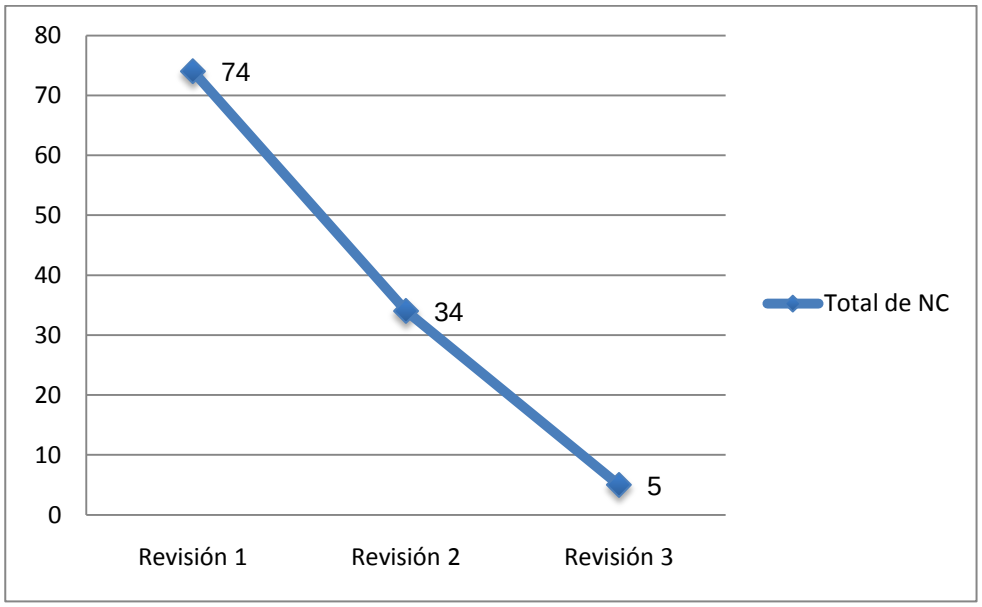


Figura 17. Resultados de las revisiones hechas a la versión funcional del sistema.

3.4 Seguimiento y escalamiento de no conformidades

Tomando como punto de referencia las actividades planteadas en el subproceso IPP-3522:2009 Seguimiento y escalamiento de no conformidades, se definió en el Registro de Evaluaciones del proyecto (REP) una sección destinada a controlar el seguimiento de las NC. De forma general, cuando una no conformidad es detectada se registra en el REP especificando su tipo (adherencia a proceso/producto[31]) y el impacto que tiene en el proyecto. Posteriormente se asigna la resolución de la misma al responsable del proceso o producto donde fue encontrada, quien propone las acciones correctivas (AC) a ser

[30] No conformidades/conformidad.

[31] Adherencia a proceso se refiere al cumplimiento con los lineamientos establecidos. Adherencia a producto se enfoca en la satisfacción de las necesidades de clientes y usuarios.

ejecutadas. Existen 3 tipos de AC[32] (capacitación, formación y toma de medidas), las cuales se clasifican en correctiva, preventiva y de mejora. Las correctivas son aquellas en las que se corrige solamente la no conformidad, preventiva cuando se procura que no se repita la no conformidad y de mejora cuando se modifica algún proceso. Para dar seguimiento a las NC se realizan varias revisiones para verificar que los errores señalados con anterioridad hayan sido corregidos. Cuando una NC es detectada se dice que se encuentra "abierta". Si en revisiones posteriores aún quedan NC pendientes porque no pudieron ser resueltas por la persona a la cual fue asignada, se propone reasignar la no conformidad definiendo el nuevo responsable de solucionarla y el cargo que ocupa en su nivel de escalabilidad; de lo contrario se cambia su estado a "cerrada".

Visual Studio constituye una vía alterna para la asignación de AC. Al seleccionar la opción correspondiente se genera una plantilla que permite introducir el estado, prioridad, responsable y título de la acción correctiva. (Ver Anexo 14. Asignación de acciones correctivas a través de Visual Studio Team System)

3.5 Auditorías y revisiones externas

La detección temprana de defectos trae consigo grandes beneficios debido a que contribuye a evitar errores en el producto final. Las revisiones constituyen una vía para la detección y corrección de los mismos ya que permite (10):

- Evaluación del progreso del proyecto
- Potencia las capacidades de los participantes
- Mejoran la comunicación entre el equipo de desarrollo, aumentando su motivación
- Proporciona aprendizaje, retroalimentación y prevención
- Forma y educa a los participantes

Con el objetivo de garantizar la calidad del producto final se acometieron una serie de revisiones internas y externas encaminadas a detectar aquellos problemas que pueden atentar contra la aceptación del cliente. En noviembre, tanto el Centro de Identificación y Seguridad Digital (CISED) como Calisoft realizaron revisiones y auditorías que sirvieron para mejorar de cierta forma el proceso de desarrollo del proyecto. Los principales elementos a medir por ambas entidades se centraron en la adherencia a procesos y productos. Las no conformidades encontradas sirvieron como base para establecer

[32] Acciones correctivas.

mecanismos que contribuyeran a asegurar la calidad del software. En el Anexo 15. Documento de no conformidades emitido por CISED, se especifican en qué consistieron los problemas hallados por dicha organización.

Como resultado de la auditoría realizada por Calisoft fueron arrojados datos que indican que a pesar de ser un proyecto que para esa fecha estaba prácticamente surgiendo, contaba con determinada madurez en cuanto a organización y planificación, motivos por los cuales se obtuvo una evaluación satisfactoria, que es la máxima categoría que otorga dicha entidad en este tipo de evaluaciones. (Ver Anexo 16. Minuta de Reunión de Cierre de auditoría de Calisoft). A continuación se exponen los resultados antes mencionados:

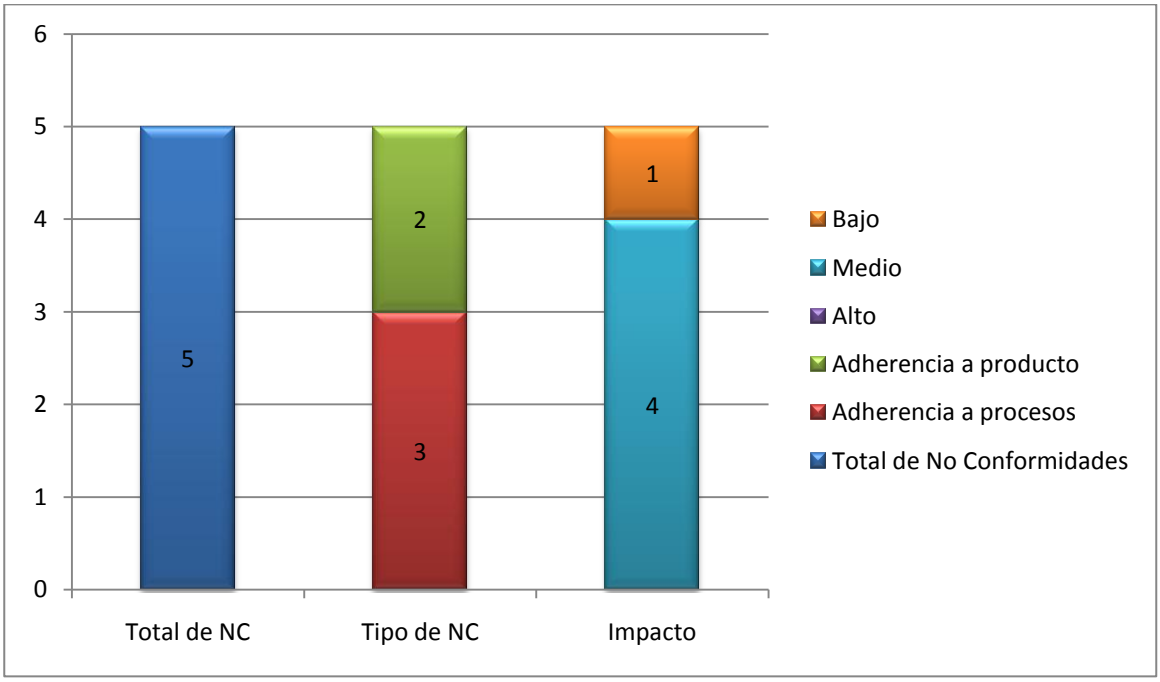


Figura 18. Reporte de No Conformidades emitido por Calisoft.

También se verificó el cumplimiento con los lineamientos de calidad establecidos por Calisoft para la Universidad de las Ciencias Informáticas, siendo registrado el mismo en la siguiente figura:

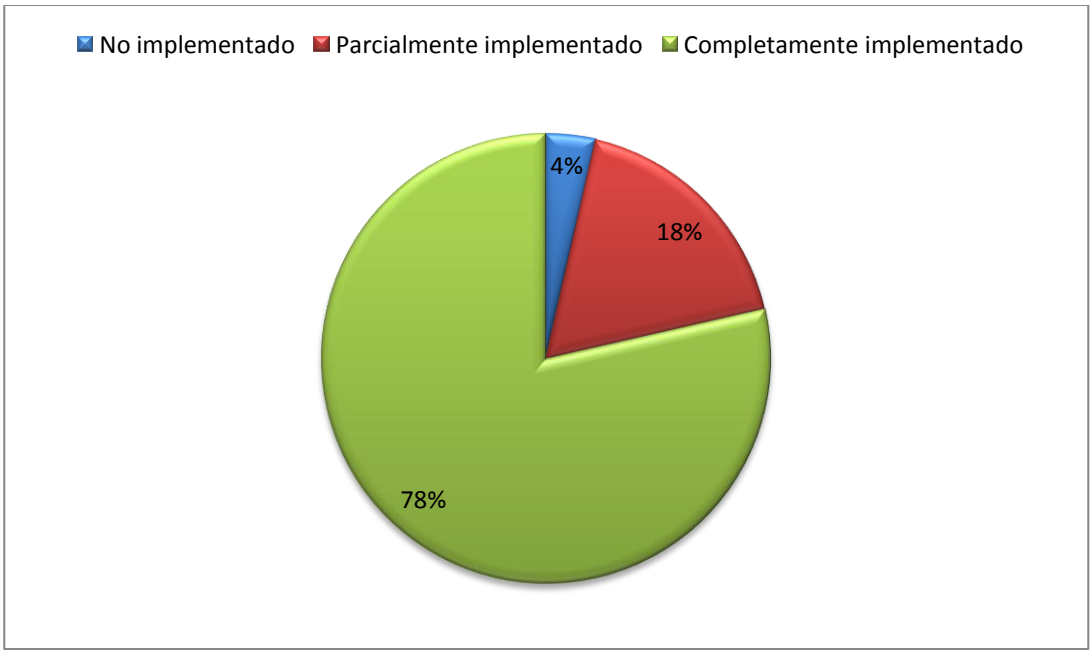


Figura 19. Porcentaje del cumplimiento con los lineamientos de calidad.

Como principales fortalezas salieron a relucir las siguientes:

- El proyecto, a pesar de no estar incluido dentro del alcance del programa de mejora, mantiene interfaces con el mismo y se han utilizado procesos y documentos definidos en dicho programa obtenido excelentes resultados.
- El ambiente de desarrollo es muy favorable para el equipo de proyecto.
- Los líderes, analistas, aseguradores de la calidad y demás roles directivos están muy capacitados y dominan correctamente los procesos.

3.6 Resumen de cumplimiento de las tareas

Partiendo de las tareas propuestas para cada fase del proceso de desarrollo del software y el estado de ejecución en que se encuentra cada una, se confeccionó un resumen en el cual se indica el nivel de cumplimiento hasta la primera iteración del proyecto. (Ver Anexo 17. Resumen del cumplimiento de las tareas propuestas).

3.7 Lecciones aprendidas con la aplicación de la estrategia

Como parte del análisis de la estrategia propuesta se realizaron entrevistas a varios de los integrantes del equipo de desarrollo. En la mayoría de los casos las respuestas estuvieron enfocadas en los aportes que brindó la misma, los cuales son citados a continuación:

- **La implicación de todo el equipo de desarrollo en las tareas SQA** tributa a la optimización del tiempo planificado en los cronogramas de trabajo para el cumplimiento de cada actividad SQA. Además, contribuye a que las tareas se asignen, ejecuten y supervisen por los especialistas del tema que se revisa.
- **La existencia de proveedores de requisitos** facilita la comprensión de las necesidades del cliente y permite a los analistas la obtención de requerimientos con un mayor nivel de especificidad.
- **La estandarización de la documentación** constituye un paso de avance en lo que se refiere a organización.
- **La vinculación del cliente a las revisiones, tanto estáticas como dinámicas, y el uso de listas de chequeo**, influyen en la detección temprana de errores que pueden comprometer el adecuado funcionamiento del software.
- **El trabajo en equipo** es factible y favorece a la adecuada gestión de los conflictos debido a que cada uno aporta sus conocimientos.

3.8 Conclusiones del capítulo

En este capítulo se mostraron los resultados de la aplicación de la estrategia propuesta para lograr el aseguramiento de la calidad en el Sistema Único de Identificación Nacional y se presentaron los resultados obtenidos, logrando demostrar la validez de la misma a través del desarrollo de los diferentes procesos establecidos en cada una de las fases de desarrollo.

CONCLUSIONES

El aseguramiento de la calidad de un producto software es un tema importante y polémico debido a que un software sin la calidad requerida hace perder prestigio a la organización que lo desarrolla. En el proyecto Sistema Único de Identificación Nacional de la República de Cuba no se tenía establecido cómo llevar a cabo este proceso, por lo que se propuso realizar una estrategia para asegurar la calidad de un producto software durante todo el ciclo de vida.

Como conclusiones generales de la investigación se muestran alcanzados los objetivos propuestos satisfactoriamente:

- *Realizar el marco teórico de la investigación.* La tarea efectuada para dar cumplimiento a este objetivo fue la realización de un estudio bibliográfico de los principales conceptos asociados al tema, así como la definición de actividades de aseguramiento de la calidad y la identificación de los principales estándares y modelos de calidad que podrían ser empleados en la estrategia.
- *Conformar una estrategia de aseguramiento de la calidad.* Para darle cumplimiento a este objetivo, en el Capítulo I del trabajo investigativo se estudiaron a profundidad las principales definiciones relacionadas con el aseguramiento de la calidad dadas por diferentes autores, se tuvo en cuenta también el proceso de mejora en que está inmerso la UCI, y la relación que presenta con la metodología de desarrollo empleada: *MSF for CMMI*. Además, se seleccionaron las técnicas de aseguramiento de la calidad a emplear en el proyecto sobre la base de estos conocimientos, en el Capítulo II se explicó la confección de la estrategia y el basamento que tuvo en el Libro de Proceso para PPQA (IPP-3520-2009).
- *Aplicar la estrategia de aseguramiento de la calidad propuesta*. Se aplicó la estrategia y se le dio seguimiento a los resultados obtenidos hasta el momento actual del proyecto.

RECOMENDACIONES

Teniendo en cuenta que el principal objetivo de todo proyecto de software es entregar un producto con la calidad requerida para ser capaz de satisfacer las necesidades de los clientes y usuarios; y que el presente trabajo recoge solamente los resultados obtenidos a partir de las revisiones efectuadas durante la primera iteración, se recomienda:

- Completar la Estrategia para las etapas de Estabilización y Despliegue.
- Realizar otras revisiones en posteriores iteraciones de la aplicación que permitan la verificación y rectificación de los errores.
- Aplicar los principios de la mejora continua en función del refinamiento de la estrategia.

REFERENCIAS BIBLIOGRÁFICAS

1. **ISHIKAWA, K.** *"¿Qué es el Control Total de la Calidad?: Modalidad Japonesa ".* 1998.

2. **Pressman, R.S.** *Ingeniería de Software.Un enfoque práctico.* 2002. pág. 614p. Vol. 5ta.

3. **Domínguez, Jorge.** The Curious Case of the CHAOS Report 2009. [En línea] 2009. http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-2009.html.

4. **Ramírez, Yudenia, de la Vega, Erick y López, Brisey.** *Proyecto Técnico.* Ciudad de La Habana : s.n., 2009.

5. **Serrano, Gloria Peréz.** *Investigación Cualitativa: Retos e Interrogantes.* Ciudad de la Habana:CEPES : s.n., 1996.

6. **Álvarez, Carlos.** *Metodología de la Investigación Científica.* Centro de Estudios de Educación Superior "Manuel Grain". Santiago de Cuba : s.n., 1995.

7. **IEEE Std 610.12-1990.** *IEEE Standard Glossary of Software Engineering Terminology.*

8. **ISO 8402:1994.** *Quality management and quality assurance - Vocabulary .* 1994.

9. **Sánchez, P. F.** Aseguramiento de la Calidad del Software. [En línea] 2006. http://pfsanchez.blogspot.com/2006/08/aseguramiento-de-la-calidad-del.html.

10. **Vegas, Sira, Juristo, Natalia y Moreno, Ana M.** *Técnicas de Evaluación de Software.* [En línea] [Citado el: 10 de diciembre de 2009.] http://is.ls.fi.upm.es/docencia/erdsi/Documentacion_Evaluacion_7.pdf.

11. **Guzmán Arenas, Adolfo.** *Mitos, creencias y supersticiones sobre la calidad del software y de su enseñanza.* 2003.

12. **IEEE Std 730, 1998a.** *Standard for Software Quality Assurance Plans.*

13. **PMBOK.** *Guía de los Fundamentos de la Dirección de Proyectos .* 2005.

14. **Lovelle, Juan Manuel.** *Calidad del Software.* 1999. pág. 12.

15. **R., Chalmeta.** *"ADSI II. 2º Boletín de transparencias".* 2000.

16. **G., Piattini Mario.** *Análisis y diseño detallado de aplicaciones.* [ed.] RA-MA Editorial. 1ª ed. Madrid : s.n., 1996.

17. **NC-ISO/IEC Std.9126-1.** *Software engineering—Product quality—Part 1: Quality Model.* 2005.

18. **Boletín IIE.** *Moprosoft: el nuevo modelo que impondrá una norma mexicana para la calidad en la industria del software.* México : s.n., 2003.

19. **Alarcón, Armando Silva.** Modelos de calidad.La industria del software en México. *Entérate en línea.* 27 de noviembre de 2008, Vol. 74.

20. **Romero, Claudia Ivette García.** El Modelo de Capacidad de Madurez y su Aplicación en Empresas Mexicana de Software. 2001.

21. **INSTITUTE, S. E.** *CMMI for Developmen tVersion 1.2 " Improving processes for better products".* 2006. pág. 573.

22. **Calisoft.** *Proceso de Mejora.* [En línea] UCI, 2010. http://calisoft.uci.cu.

23. **Turner, Michael S. V.** Microsoft Solutions Framework Essentials: Building Successful Technology Solutions. [En línea] 2006. http://www.microsoft.com/mspress/books/10250.aspx.als: Building Successful Technology Solutions. ISBN 0-7356-2353-8.

24. **Cabrera, Armando.** *Procesos de Ingeniería del Software.* Loja : s.n. pág. 16.

25. **estudio, Centros de.** [En línea] 7 de 6 de 2004. www.informatizate.net.

26. **Keeton, Marlys.** *Microsoft Solutions Framework (MSF): A Pocket Guide.* s.l. : Van Haren Publishing, 2006.

27. **IEEE Std 729 - 1983.** *Glosario Estándar IEEE de Terminología de Ingeniería del Software.*

28. **IEEE Std.1012.** *Standard for Software Verification and Validation , 1998b.* Std 1012.

29. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El proceso Unificado de Desarrollo de Software. [En línea] 2004.

30. **López Quesada, Juan Antonio.** *Fundamentos de Ingeniería del Software.* [En línea] 2005. [Citado el: 23 de noviembre de 2009.] http://dis.um.es/~lopezquesada/FISw.htm.

31. **Fernández, Luis.** *Presentación del Grupo de Calidad de Software.* [En línea] 2002. [Citado el: 5 de diciembre de 2009.] http://www.ati.es/gt/calidad-software/presentacion.htm.

32. **Weitzenfeld, A.** *Ingeniería de software orientada a objetos con Java e Internet.* [ed.] Thomson Paraninfo. 2005. pág. 678.

33. **León, F.** *Ingeniería del Software.* 2006.

34. **Mañas, José A.** *Prueba de Programas.* 1994.

35. **Ramírez, Yudenia.** *Estrategia de Integración para el Sistema de Identificación, Migración y Control de Extranjeros de la República Bolivariana de Venezuela.* Ciudad de La Habana : s.n., 2008. pág. 198.

36. **Integradores de tecnologías de la información y las comunicaciones.** CESSER Informática y Organización. [En línea] 2006. http://www.cesser.com/es/solucion.aspx?prod=111.

GLOSARIO

A

AC: Acciones correctivas, 69

AMCIS: Asociación Mexicana para la Calidad en Ingeniería de Software, 14

Aseguramiento de la calidad: El aseguramiento de la calidad es la aplicación de actividades planificadas y sistemáticas relativas a la calidad para asegurar que el proyecto emplee todos los procesos necesarios para cumplir con los requisitos., 9

B

Beneficio: Está sujeto a las necesidades y satisfacción del cliente., 59

Branches: Ramas de desarrollo, 101

Bug: Es un elemento de trabajo que se comunica que un posible problema existe o ha existido en el sistema., 119

C

Calidad: La capacidad de un grupo de características inherentes a un producto, componente o proceso de satisfacer los requerimientos del cliente., VI

Calisoft: Entidad perteneciente a la UCI que brinda servicios para la realización de revisiones y pruebas de calidad a productos de software ., 70

Casos de prueba: Conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular., 25

Check-in: Se refiere a la acción de adicionar un elemento al repositorio., 97

CISED: Centro de Identificación y Seguridad Digital, 70

CMMI: Modelo Integrado de Madurez de la Capacidad., 15

Completitud: Nivel o grado en que algo está completo., 42

***Corrección*:** El grado en que un sistema o componente está libre de fallas en su especificación, diseño e implementación., 42

D

DCP: Diseño de Casos de Prueba, 110

Diseño de las pruebas: Comprende la identificación de la técnica o técnicas de pruebas que se utilizarán para probar el software., 25

DNC: Documento de No Conformidades, 86

***DTA*:** Descripción Textual de Actividades, 44

E

ERS: Especificación de Requisito de Software., 11

Esfuerzo: Establece la duración estimada de tiene el desarrollo de un determinado elemento

en función de la complejidad que le aporta a la solución., 59

Estabilidad: Establece el porcentaje admisible de cambios que puede tener un elemento., 59

Estado: Se utiliza para controlar el estado de los elementos de trazabilidad a lo largo del proceso de desarrollo del software., 59

Estrategia: La estrategia se concibe como una manera de planificar y dirigir las acciones para alcanzar determinados objetivos. Sus elementos claves son la determinación de metas y objetivos a largo, mediano y corto plazo y la adaptación de acciones y recursos necesarios., VI

Estrategias de prueba: Integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software., 27

F

Framework: Conjunto de clases modeladas de forma general para resolver problemas relacionados en un contexto específico., 66

I

IEEE: Instituto de Ingenieros Eléctricos y Electrónicos., 9

IEEE 1012-1998: Estándar para la Verificación y Validación de Software., 23

IEEE Std 729-1983: Glosario Estándar IEEE de Terminología de Ingeniería del Software., 22

IEEE-830-1998: Estándar para la Especificación de Requisito de Software, 11

ISO: Organización Internacional para la Estandarización., 10

ISO/IEC 9126: Estándar Internacional para la Evaluación del software., 11

L

Legibilidad: Calidad de la escritura que hace que sea fácil de leer y entender., 42

M

MSF: Microsoft Solutions Framework., 19

N

NC: No conformidades, 69

No conformidad: Problema detectado en un artefacto según error con respecto a lo definido en artefactos anteriores y/o en lo pactado con el cliente. No concordancia con normas internacionales que deben ser cumplidas por el artefacto. Insatisfacción del cliente con el resultado final de un elemento de configuración según lo pactado con anterioridad en el proyecto, 69

P

Peer review: Es una técnica en la cual un trabajo se somete al escrutinio de uno o más expertos

en el área y se emite una evaluación del trabajo que incluye sugerencias sobre cómo mejorarlo, 52

PPQA: Aseguramiento de la Calidad de Procesos y Productos, 18

Procedimiento de pruebas: especifica cómo realizar uno o varios casos de prueba o partes de estos., 26

Pruebas de software: Constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas., 23

R

Release: Versión funcional del software, 122

REP: Registro de Evaluaciones del proyecto, 69

Runtime: Son aquellos servicios que van a estarse ejecutando constantemente para todas las instancias de workflow., 104

S

SEI: Software Engineering Institute, 18

SIE Center: Software Industry Excellence Center, 17

SPICE: Software Process Improvement Capability dEtermination., 13

SUIN: Sistema Único de Identificación Nacional., 2

U

UNAM: Universidad Nacional Autónoma de México, 14

V

Validación: Proceso de evaluación del software al final del proceso de desarrollo para asegurar el cumplimiento de las necesidades del cliente., 22

Verificación: Proceso de determinar si los productos de una cierta fase del desarrollo de software cumplen o no los requisitos establecidos durante la fase anterior., 22

Verificación y Validación: El proceso de determinar si los requisitos para un sistema o componente son completos y correctos, los productos de cada fase de desarrollo cumplen con los requisitos o condiciones impuestas por la fase anterior y el sistema o componente final cumple con los requisitos especificados., 22

VSTS: Visual Studio Team System, 21

W

Walkthrough: es una forma de peer review en la que un diseñador o programador guía a miembros del equipo de desarrollo y otras partes interesadas a través del producto de software. Los participantes hacen preguntas y comentarios sobre posibles errores, violación de estándares de desarrollo y otros problemas., 53

***Workitems*:** Elementos de trabajo., 22