View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Universidad Nacional de Lanús: Portal de revistas científicas de la UNLa "Arturo Peña...

Aseguramiento de la Calidad en la Construcción de Sistemas Basados en el Conocimiento: Un Enfoque Práctico

Eduardo Diez
Laboratorio de Investigación y Desarrollo en Aseguramiento de Calidad de Software
Grupo Investigación en Sistemas de Información
Departamento Desarrollo Productivo y Tecnológico. Universidad Nacional de Lanús.
Remedios de Escalada, Buenos Aires, Argentina.
ediez@unla.edu.ar

***Resumen*—La función de aseguramiento de la calidad del software (SQA) se debe basar en un planificado y sistemático diseño de acciones y métodos, requeridos para garantizar la calidad del mismo. En el presente trabajo, se presenta un diseño de acciones y métodos que constituyen un enfoque práctico para el desempeño de la función de SQA en una organización, adaptado especialmente a la metodología IDEAL para el desarrollo de sistemas basados en el conocimiento. Este enfoque no pretende ser exclusivo y en ningún caso limita o inhibe la aplicación de otras acciones, métodos o modelos, sino que podrá ser su complemento, adaptándolo convenientemente. El enfoque o modelo de aseguramiento de la calidad del software que tiene las siguientes características: el modelo de aseguramiento de calidad de software sugerido es una interfaz a una metodología, ideal en este caso, de desarrollo de software, el modelo que aquí se presenta no es una metodología en sí misma, sino que debe acoplarse a una metodología de desarrollo para poder implementarse, esta interfaz está compuesta por módulos independientes, donde cada uno de ellos se asocia a ciertos procesos de la metodología ideal.**

***Palabras Claves*—Aseguramiento de la calidad del software, sistemas basados en el conocimiento, metodología ideal.**

I. Introduccion

La calidad es una cualidad esencial de cualquier producto, generado por una organización, que va a ser usado por otros. Antes del siglo veinte, las actividades relacionadas con el aseguramiento de la calidad era responsabilidad única de la persona que construía el producto. La primera función de control y de aseguramiento de la calidad formal fue introducida por los laboratorios Bell en 1916 y se extendió rápidamente por todo el mundo de las manufacturas. Hoy en día, cada empresa tiene un mecanismo que asegura la calidad de sus productos, de hecho, durante la pasada década se ha usado ampliamente como táctica de mercado, la declaración explícita de mensajes que ponían de manifiesto la calidad ofrecida por las empresas.

La evolución del aseguramiento de la calidad en el desarrollo de software ha sido paralela a la evolución de la calidad en la fabricación de hardware. Durante los primeros años de la informática (los años 50 y 60), la calidad era responsabilidad únicamente del programador. Durante los años 70 se introdujeron estándares de aseguramiento de la calidad para el software en los contratos militares de desarrollo de software y se extendieron rápidamente en los desarrollos de software del mundo comercial.

La función de aseguramiento de la calidad del software (SQA) se debe basar en un planificado y sistemático diseño de acciones y métodos, requeridos para garantizar la calidad del mismo. El alcance de la responsabilidad del aseguramiento de la calidad, en el desarrollo de software, abarca a muchos constituyentes de una organización, tales como ingenieros de software, líderes de proyecto, clientes, comerciales y personas que trabajan dentro del equipo de SQA (una conformación del mismo se presentará en capítulos sucesivos).

El equipo de SQA debe analizar el software desde diversos puntos de vista, respondiendo a algunas de estas preguntas:

- ¿Satisface el software, de forma adecuada los principales factores de calidad?
- ¿Se ha realizado el desarrollo del software de acuerdo con estándares preestablecidos?
- ¿Se han aplicado las técnicas y métodos apropiados para el desarrollo del software?

Para responder a éstas y otras cuestiones, en el presente trabajo, se presenta un diseño de acciones y métodos que constituyen un enfoque práctico para el desempeño de la función de SQA en una organización, adaptado especialmente a la metodología IDEAL para el desarrollo de sistemas basados en el conocimiento. Este enfoque no pretende ser exclusivo y en ningún caso limita o inhibe la aplicación de otras acciones, métodos o modelos, sino que podrá ser su complemento, adaptándolo convenientemente.

En la sección II se establecen los Conceptos Básicos, describiendo brevemente los conceptos de calidad, control y aseguramiento de la misma y presentando el alcance de la función de SQA en una organización. En la sección III se esboza el Modelo General, es decir, el diseño de acciones y métodos que constituyen un enfoque práctico para el desempeño de la función de SQA en una organización. También de presenta el modelo de mejora continua del mismo. En la sección IV se describen los Módulos del Modelo, se presentan para cada uno de ellos las acciones a desempeñar, su objetivo, sus entradas, salidas, lista de verificación, métricas y participantes involucrados. En la sección V se describe las principales Técnicas y Herramientas, a utilizar en los módulos del modelo descrito. En la sección VI se describe el Nivel de Servicio Esperado, a través de un acuerdo de nivel de servicio recomendado. En la sección VII se detalla el Equipo de SQA sugerido, indicando su estructura y responsabilidades. En la sección VIII se establecen las Conclusiones del presente

trabajo, señalando los beneficios y problemas esperados en la aplicación de la función de SQA en una organización. También se establecen las bases para un análisis costo-beneficio. Por último se detalla la Bibliografía utilizada para la confección del presente trabajo, ya sea referenciada o consultada.

II. Conceptos Básicos

En la presente sección se establecen los Conceptos Básicos, describiendo brevemente los conceptos de calidad, control y aseguramiento de la misma y presentando el alcance de la función de SQA en una organización.

A. Paradigma de la calidad

El paradigma de la calidad es aplicable a todas las actividades que se llevan a cabo en una organización. Se puede definir en términos generales a la calidad como:

La calidad es la suma de todos aquellos aspectos o características de un producto o servicio, que influyen en su capacidad para satisfacer las necesidades de los usuarios.

Por otro lado, con respecto a la satisfacción del usuario, se puede decir que:

La satisfacción del usuario está determinada por la diferencia entre la calidad percibida y la calidad esperada, cuando éste hace uso de un producto o servicio.

Los principales elementos del paradigma de la calidad son los siguientes:

- La naturaleza de la calidad: Orientación a los aspectos o características de un producto o servicio que influyan en su capacidad para satisfacer necesidades dadas, más que a la adecuación a estándares o a especificaciones preestablecidas.
- La perspectiva del proceso: Focalización en el proceso más que en el producto.
- Orientado a datos: Basado en la recolección, análisis y comparación de datos.
- Focalización en el cliente o usuario: La obtención de la satisfacción del cliente o usuario es el objetivo final de todo proceso.
- Eliminación de defectos: Priorización de técnicas de prevención de defectos sobre técnicas de detección y corrección.
- Gestión para la calidad: La adopción del paradigma requiere del compromiso de la alta dirección.

En primer término, se debe diferenciar entre la calidad del producto y la calidad del proceso que lo genera.

Las primeras aproximaciones a la calidad estaban basadas solamente en el control de la calidad del producto terminado, es decir en actividades que sólo desataban acciones correctivas para eliminar los defectos del producto. A estas actividades se las suele catalogar como correctivas, tardías y relacionadas con el producto.

Con el tiempo y tal cual se establece en uno de los principios del paradigma de la calidad, las aproximaciones de calidad se fueron trasladando al aseguramiento de la misma sobre el proceso que genera el producto, es decir en actividades preventivas, antes que el producto esté terminado, y que desatan acciones para evitar que el producto terminado tenga defectos. A estas actividades se las suele catalogar como preventivas, tempranas y relacionadas con el proceso.

Ahora bien, la realidad muestra que toda aproximación eficaz a la calidad, contiene una combinación de actividades de aseguramiento de la calidad y de actividades de control de la misma y que estas se complementan fácilmente.

B. Calidad del Software

Particularmente, en el caso del software, existen muchas definiciones distintas en la bibliografía, sin embargo, en lo que a este trabajo respecta, tomaremos la definición de R. Pressman [1]:

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

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

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

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

B.1. Puntos de vista de la calidad del software

En la ingeniería del software o del conocimiento, la visión de la calidad no es única, dependiendo del punto de vista desde el cual se la analice, ver figura 1.

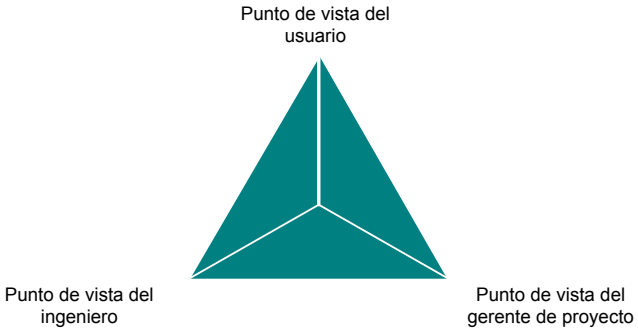

Fig. 1. Puntos de vista de la calidad del software

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

- Punto de vista del usuario: El punto de vista del usuario, con respecto a la calidad, estará basado en los factores externos del producto, tales como funcionalidad y facilidad de operación.

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

B.2. Factores que determinan la calidad del software

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

McCall y sus colegas [2] han propuesto los siguientes:

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

Es difícil, y en algunos casos imposible, desarrollar medidas directas de los anteriores factores de calidad. Por tanto, cada factor se descompone en atributos o criterios, más fácilmente medibles. Cabe aclarar que cada uno de estos atributos puede corresponder a más de un factor de calidad.

- Facilidad de auditoría. La facilidad con que se puede comprobar la conformidad con los estándares.
- Exactitud. La precisión de los cálculos y del control.
- Normalización de las comunicaciones. El grado en que se usan el ancho de banda, los protocolos y las interfaces estándar.
- Completitud. El grado en que se ha conseguido la total implementación de las funciones requeridas.
- Concisión. Lo compacto que es el programa en términos de líneas de código.
- Consistencia. El uso de un diseño uniforme y de técnicas de documentación a lo largo del proyecto de desarrollo del software.
- Estandarización en los datos. El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.
- Tolerancia de errores. El daño que se produce cuando el programa encuentra un error.
- Eficiencia en la ejecución. El rendimiento en tiempo de ejecución de un programa.
- Facilidad de expansión. El grado en que se puede ampliar el diseño arquitectónico, de datos o procedimental.
- Generalidad. La amplitud de aplicación potencial de los componentes del programa.
- Independencia del hardware. El grado en que el software es independiente del hardware sobre el que opera.
- Instrumentación. El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen.
- Modularidad. La independencia funcional de los componentes del programa.
- Facilidad de operación. La facilidad de operación de un programa.
- Seguridad. La disponibilidad de mecanismos que controlen o protejan los programas o los datos.
- Autodocumentación. El grado en que el código fuente proporciona documentación significativa.
- Simplicidad. El grado en que un programa puede ser entendido sin dificultad.
- Independencia del sistema de software. El grado en que el programa es independiente de características no estándar del lenguaje de programación, de las características del sistema operativo y de otras restricciones del entorno.
- Facilidad de traza. La posibilidad de seguir la pista a la representación del diseño o de los componentes reales del programa hacia atrás, hacia los requisitos.
- Formación. El grado en que el software ayuda para permitir que nuevos usuarios apliquen el sistema.

Otra lista de factores de calidad es la desarrollada por Grady y sus colegas [3]. Los factores y sus atributos correspondientes son los siguientes:

- La funcionalidad se obtiene mediante la evaluación del conjunto de características y de posibilidades del programa, la generalidad de las funciones que se entregan y la seguridad de todo el sistema.

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

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

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

C. Alcance de la función de SQA

La calidad del software no es algo en lo que se empieza a pensar una vez que se ha generado el código. Según R Pressman [1] el aseguramiento de la calidad del software (SQA) es una "actividad de protección" que se aplica a lo largo de todo el proceso de ingeniería software y engloba:

1) Un enfoque de gestión de la calidad.
2) Tecnología de ingeniería del software o del conocimiento efectiva (métodos y herramientas).
3) Revisiones técnicas formales que se aplican durante cada paso de la ingeniería del software o del conocimiento.
4) Una estrategia de prueba en múltiples niveles.
5) El control de la documentación del software y de los cambios realizados.
6) Un procedimiento que asegure un ajuste a los estándares de desarrollo del software (cuando sea posible).
7) Mecanismos de medición y de información.

Las anteriores involucran tanto a los ingenieros de software como al equipo de SQA. Los ingenieros de software deben aplicar métodos técnicos y mediciones sólidas, conducir revisiones técnicas formales y ejecutar pruebas de software bien planificadas. Por otro lado, de acuerdo al Software Engineering Institute (SEI) [4], el equipo de SQA tiene como propósito proveer a la gerencia la visibilidad apropiada del proceso que está siendo usado y de los productos siendo desarrollados. El propósito involucra:

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

Las actividades del equipo de SQA, acorde al SEI [4], comprenden:

Preparar un plan de SQA para el proyecto: El plan es desarrollado durante la planificación del proyecto y es revisado por todas las partes interesadas. Las actividades de aseguramiento de la calidad a desempeñar por el equipo de ingenieros de software y el equipo de SQA son regidas por este plan. El plan identifica:

- Evaluaciones a realizar.
- Auditorías y revisiones a realizar.
- Estándares aplicables al proyecto.
- Procedimientos para reporte y seguimiento de errores.
- Documentos a ser producidos por el equipo de SQA.
- Retro-alimentación a proveer al equipo del proyecto de software.
- Participar en el desarrollo de la descripción del proceso de software del proyecto: El equipo de software selecciona un proceso de software para el proyecto, el equipo de SQA revisa la descripción del proceso, verificando que cumpla con las políticas de la organización, estándares de software internos y estándares impuestos externamente, entre otros.
- Revisar las actividades de ingeniería del software o de conocimiento para verificar que cumplan con el proceso de software definido: El equipo de SQA identifica, documenta y hace el seguimiento de cualquier desviación del proceso y verifica las correcciones realizadas.
- Auditar productos de trabajo de software designados, para verificar que cumplan con aquellos definidos como parte del proceso de software: El equipo de SQA revisa productos seleccionados; identifica, documenta y hace el seguimiento de las desviaciones; verifica las correcciones realizadas y periódicamente informa los resultados de su trabajo al gerente del proyecto.

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

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

III. Modelo General

En la presente sección se esboza el Modelo General, es decir, el diseño de acciones y métodos que constituyen un enfoque práctico para el desempeño de la función de SQA en una organización. También se presenta el modelo de mejora continua del mismo.

A. Modelo de aseguramiento de calidad sugerido

Una metodología de desarrollo de software (ya sea para software convencional o para sistemas basados en el conocimiento) establece una forma consistente de ejecutar las actividades relacionadas con el software dentro de una organización. Toda metodología describe:

- Elementos: Cubren un conjunto de actividades bien definidas, relacionadas y agrupadas.
- Arquitectura: Establece el orden, las interfaces, las interdependencias y otras relaciones entre los elementos.
- Adicionalmente, la metodología podría describir estándares, métodos y herramientas.

La metodología de desarrollo es la que provee continuidad en el proceso de software de una organización y es una referencia para métricas y mejoras de dicho proceso.

Sobre la base del concepto de metodología y de los conceptos básicos ya definidos en el capítulo anterior, se propone un enfoque o modelo de aseguramiento de la calidad del software que tiene las siguientes características:

- El modelo de aseguramiento de calidad de software sugerido es una interfaz a una metodología de desarrollo de software. En particular, para el presente trabajo, se utilizará la metodología IDEAL.
- El modelo que aquí se presenta no es una metodología en sí misma, sino que debe acoplarse a una metodología de desarrollo para poder implementarse.
- Esta interfaz está compuesta por módulos independientes, donde cada uno de ellos se asocia a ciertos procesos de la metodología IDEAL.

La interfaz cuenta también con un módulo de ejecución recurrente, en donde, para todas y cada uno de las entradas a cada uno de los otros módulos, se verifica la conformidad de las mismas con la metodología, estándares y normas aplicables que estén vigentes en la organización.

El modelo de aseguramiento de la calidad del software sugerido no es dependiente de la metodología de desarrollo utilizada en una organización, sino que se adapta a ella. Sobre la base de los procesos de la metodología de desarrollo, sus precedencias, sus ciclos de ejecución, y por supuesto de las características del proyecto en curso, se seleccionan aquellas porciones de la interfaz que resulten necesarias para obtener los niveles de calidad deseados y el orden de ejecución de las mismas. En la figura 2 se muestra la relación entre el modelo general de aseguramiento de la calidad del software y la metodología IDEAL.

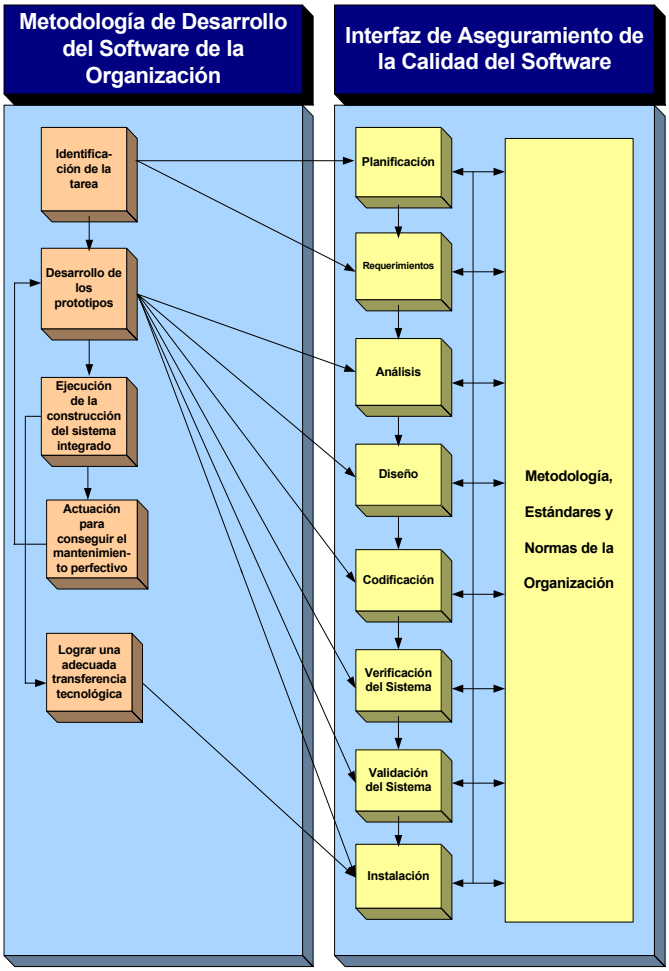

Fig. 2. Relación entre el modelo general y la metodología de desarrollo

B. Aplicación del Modelo

El modelo aquí presentado, es un modelo genérico cuya flexibilidad permite su adaptación a cada proyecto en particular. De esta forma, se podrán seleccionar los segmentos del mismo que cubran las necesidades de cada proyecto, evitando la realización de actividades innecesarias.

El modelo general de aseguramiento de la calidad, se documenta en un plan general de aseguramiento de la calidad. La flexibilidad, que permite la adaptación del modelo genérico o plan general de aseguramiento de la calidad del software, a todo tipo de proyectos, se formaliza a través de los planes específicos de aseguramiento de la calidad del software para cada proyecto.

El plan específico de aseguramiento de la calidad, para un proyecto en particular, será entonces, la adaptación del plan general a las características particulares de ese proyecto. En la figura 3 se muestra la relación entre el plan general de aseguramiento de la calidad del software y cada uno de los planes específicos.

C. Módulos del modelo

En el capítulo siguiente se describirá cada uno de los módulos del modelo general de aseguramiento de la calidad propuesto, indicando para cada uno de ellos los siguientes:

- Objetivo: Descripción del propósito del módulo, para usar como una regla contra la que medir el progreso del mismo.

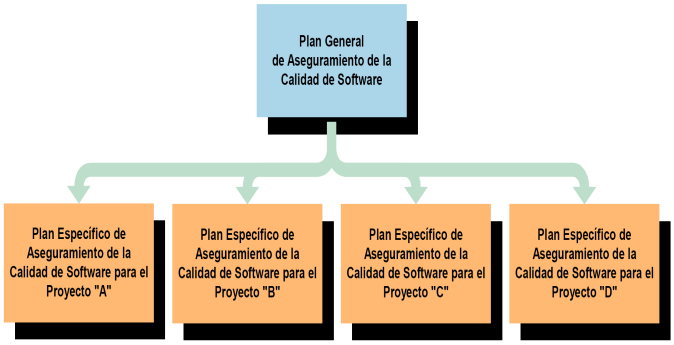

Fig. 3. Relación entre el plan general y los planes específicos

- Entrada: Documentos y/o información necesaria para completar el módulo. El nombre de los documentos es genérico, sin embargo este nombre debe ser adaptado al nombre del producto resultante, del proceso lógico correspondiente, en la metodología utilizada.
- Proceso: Descripción de las tareas y acciones que debe ejecutar el integrante del equipo de SQA para dar por cumplido el módulo.
- Salida: Productos que deben ser generados al final del módulo.
- Lista de verificación: Checklist que utilizan los integrantes del equipo de SQA para verificar que ha cumplido el módulo correctamente. Las lista de verificación se definen en tablas, los campos de esas tablas se explican en la tabla I.

TABLA I. EXPLICACIÓN DE LOS CAMPOS QUE FORMAN PARTE DE LAS TABLAS QUE DEFINEN LAS LISTAS DE VERIFICACIÓN

CAMPOS	EXPLICACIÓN
Número	Un número que identifica secuencialmente los ítems de control de calidad, el resultado positivo indicaría que ese paso ha sido realizado correctamente.
Ítem	Ítem específico de control de calidad que es usado para medir la efectividad de ejecución de este paso.
Respuesta	El analista de calidad deberá indicar en esta columna si ha realizado el ítem referenciado. La respuesta puede ser: SI, NO o N/A si la misma no es aplicable al proyecto en cuestión.
Comentarios	Esta columna es utilizada para clarificar la respuesta de SI, NO o N/A para los ítems indicados. Generalmente la columna de comentarios sólo se completa en las respuestas por NO; las respuestas por NO deberán ser investigadas y se deberá tomar una determinación respecto a si este ítem deberá ser completado antes de considerar este paso completo.

D. Actualización del plan

D.1. Modelo de mejora continua del plan

El modelo o plan general de aseguramiento de la calidad del software aquí presentado no mantiene un estado estacionario o estático, sino que es dinámico y sujeto a permanentes mejoras y actualizaciones de acuerdo a la experiencia y a las necesidades que surjan como consecuencia de la aplicación del mismo.

En la figura 4 se muestra el modelo de actualización y mejora continua del plan general de aseguramiento de la calidad del software y de los planes específicos de aseguramiento de la calidad del software de cada proyecto, así como la adaptación del plan general, y la ejecución y verificación de los planes específicos.

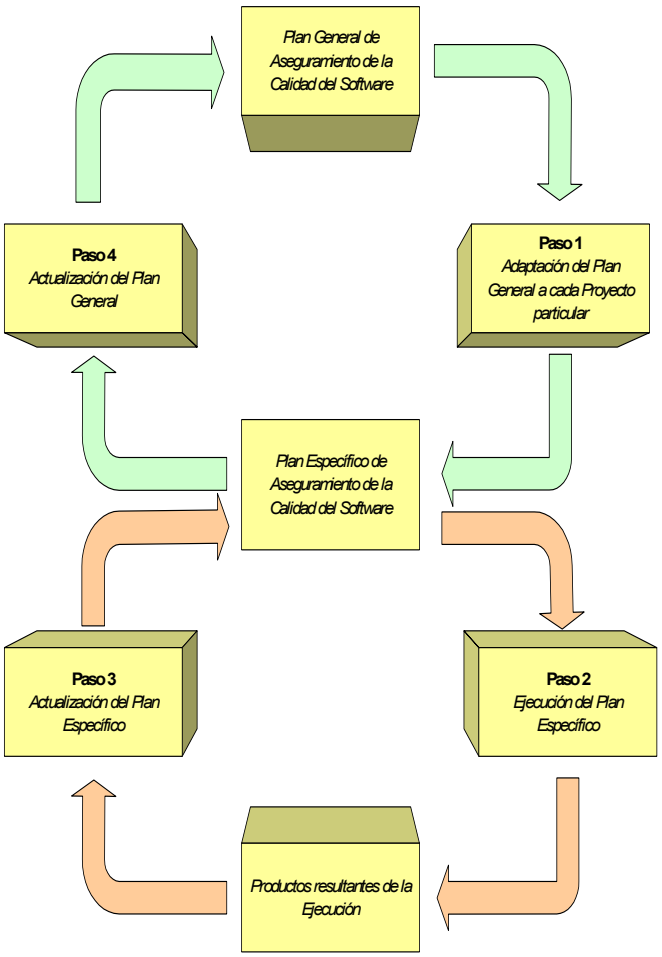

Fig. 4. Modelo de actualización y mejora continua

D.2. Detalle del modelo de mejora continua del plan

A continuación se detalla el modelo de mejora continua del plan general de aseguramiento de la calidad como así también de los planes específicos correspondientes a cada proyecto particular.

Adicionalmente, y sólo para contextualizar los pasos correspondientes a la mejora continua, se detallan brevemente los pasos correspondientes a la adaptación del plan general, y la ejecución y verificación de los planes específicos:

Paso 1:

- Entrada: Plan general de aseguramiento de la calidad de software.
- Acción: Adaptación del plan general de aseguramiento de la calidad de software a cada proyecto particular.
- Salida: Plan específico de aseguramiento de la calidad de software para un proyecto particular

Paso 2:

- Entrada: Plan específico de aseguramiento de la calidad de software, para un proyecto particular.
- Acción: Ejecución del plan específico de aseguramiento de la calidad de software.
- Salida: Productos resultantes de la ejecución del plan específico de aseguramiento de la calidad de software.

Paso 3:

- Entrada: Productos resultantes de la ejecución del plan específico de aseguramiento de la calidad de software.
- Acción: Análisis de la calidad de los productos resultantes de la ejecución del plan específico. Generación de recomendaciones y acciones a realizar, con su correspondiente justificación, para mejorar el plan específico de aseguramiento de la calidad de software.
- Salida: Plan específico de aseguramiento de la calidad de software, para un proyecto particular, actualizado.

Paso 4:

- Entrada: Actualizaciones realizadas al plan específico de aseguramiento de la calidad de software, para un proyecto particular.
- Acción: Análisis de las actualizaciones realizadas al plan específico de aseguramiento de la calidad del software, con sus correspondientes justificaciones. Actualización del plan general de aseguramiento de la calidad del software.
- Salida: Plan general de aseguramiento de la calidad del software actualizado.

IV. Módulos del Modelo

En el presente capítulo se describen los Módulos del Modelo, se presentan para cada uno de ellos las acciones a desempeñar, su objetivo, sus entradas, salidas, lista de verificación, métricas y participantes involucrados.

A. Planificación

A.1. Objetivo

- Determinar qué recursos estarán disponibles para producir y mantener el software asociado al proyecto.
- Determinar cuándo y cómo se incurrirá en costos de personal, tiempo de procesamiento, etc.
- Medir el avance del desarrollo del proyecto.

A.2. Entrada

- Plan de desarrollo (de software) del proyecto.
- Estimación del proyecto y método utilizado para realizarla.
- Descripción del proceso de desarrollo a utilizar en el proyecto.

A.3. Proceso

En la figura 5 se detalla gráficamente, el proceso a desarrollar durante este módulo:

A.3.1.Verificación de la Estimación del Proyecto

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

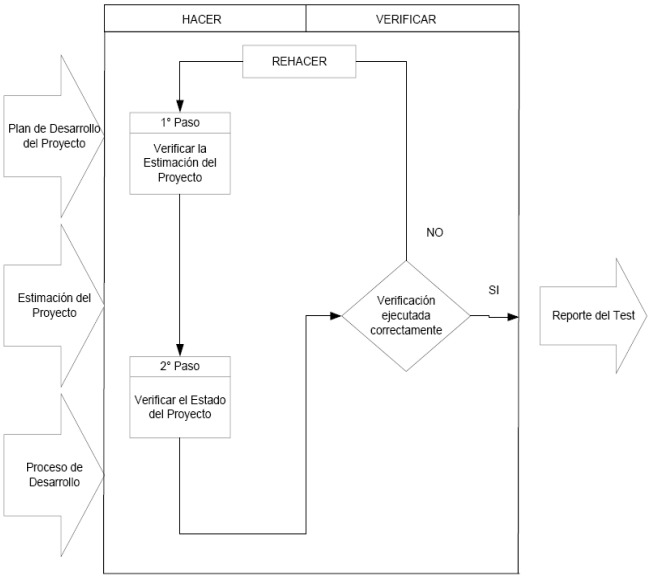

Fig. 5. Proceso del módulo de planificación

Una estimación inapropiada de costos puede dañar más a la calidad del proyecto de software que a cualquier otro factor.

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

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

El detalle de cada uno de estos puntos, se encuentra en el capítulo de Técnicas y Herramientas.

A.3.2. Verificación del Estado del Proyecto

Para conocer el estado del proyecto se sugiere un sistema simple de acumulación de puntos para medir el progreso. Luego este sistema puede compararse con el reporte gerencial del proyecto. Si hay una diferencia significativa entre ambos sistemas de medición del progreso, el analista de calidad puede cuestionar la validez de los resultados producidos por la gerencia de proyecto y/ o sistema de conteo.

El sistema de puntos para realizar la medición durante el desarrollo del software, provee un método objetivo, preciso y eficiente de recolección y reporte del rendimiento de la información en el campo de la ingeniería que a menudo carece de visibilidad. El método utiliza información basada en ítems de entregables de software y es obtenida como parte del proceso de desarrollo. Los resultados son fáciles de interpretar y pueden presentarse en una variedad de formatos. El esquema es flexible y puede modificarse para proyectos grandes y chicos. El detalle del mismo, se encuentra en el capítulo de Técnicas y Herramientas.

A.4. Salidas

- Informe de estimación del proyecto
- Informe de estado del proyecto

A.5. Lista de Verificación (Checklist)

En la tabla I se presenta la lista de verificación:

TABLA II. LISTA DE VERIFICACIÓN DEL MÓDULO DE PLANIFICACIÓN

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
1	¿La gerencia del proyecto, esta de acuerdo en tener un equipo de aseguramiento de la calidad y evaluación de la estimación y estado del plan de desarrollo?				
2	¿El equipo de aseguramiento de la calidad conoce la estimación del progreso utilizada para el proyecto?				
3	¿El equipo de aseguramiento de la calidad conoce el método para realizar los informes de estado del proyecto?				
4	¿El equipo de aseguramiento de la calidad entiende el método utilizado para estimar y calcular?				
5	¿El equipo de aseguramiento de la calidad ha realizado una buena verificación para determinar la validez de las estimaciones realizadas?				
6	Si el equipo de aseguramiento de la calidad está en desacuerdo con la validez de las estimaciones realizadas. ¿Existe un procedimiento razonable para manejar tales diferencias?				
7	¿El equipo del proyecto posee un sistema de reportes razonable para informar el estado del mismo?				
8	¿El equipo de aseguramiento de la calidad ha establecido que pueden utilizarse los informes de estado de proyecto como guía para la toma de decisiones?				
9	¿Existe algún procedimiento a seguir, si los informes de estado indican que el proyecto está por encima o por debajo de las estimaciones?				
10	¿El equipo de aseguramiento de la calidad ha tenido en cuenta los factores más importantes en la evaluación de la estimación realizada (Ejemplo: tamaño del software, etc.)?				
11	¿El equipo de proyecto ha recibido una copia del estado del mismo?				
12	¿El equipo de aseguramiento de la calidad tiene conocimiento de cómo se efectúa la planificación?				
13	¿El equipo de aseguramiento de la calidad tiene conocimiento de cómo se efectúa la estimación del proyecto?				
14	¿El equipo del proyecto tiene conocimiento del proceso de desarrollo utilizado para construir el software especificado por el proyecto?				
15	¿El plan de proyecto está completo?				
16	¿La estimación del proyecto está totalmente documentada?				
17	¿El proceso de desarrollo está totalmente documentado?				
18	¿El método de estimación utilizado para el proyecto, es razonable respecto de las características del mismo?				
19	¿La estimación efectuada es razonable como para completar el proyecto según lo especificado en el plan?				
20	¿El equipo del proyecto tiene un método definido para determinar e informar el estado del mismo				
21	¿El equipo de aseguramiento de la calidad, está de acuerdo con que el estado informado coincide con el estado actual del proyecto?				

A.6. Métricas

- Eficacia de la estimación de la duración: $\frac{\text{Duración actual del proyecto}}{\text{Duración estimada del proyecto}}$
- Eficacia de la estimación del esfuerzo $\frac{\text{Esfuerzo actual del proyecto}}{\text{Esfuerzo estimado del proyecto}}$

A.7. Involucrados

A.7.1. Equipo de Ingeniería

- Gerente de proyecto.

A.7.2. Equipo de Aseguramiento de la Calidad del Software

- Líder de aseguramiento de la calidad del software.

B. Requerimientos

B.1. Objetivo

- Determinar si los requerimientos expresan las necesidades del usuario.
- Verificar la separación de los requerimientos entre obligatorios y opcionales.
- Verificar que cada requerimiento:
- Esté debidamente documentado, reflejando las necesidades del usuario.
- Se encuentre dentro del alcance definido del sistema en cuestión.
- Se encuentre dentro de los límites del presupuesto para el sistema en cuestión.
- Se encuentre bien definido (en forma completa), de manera de evitar cambios posteriores.
- Determinar que los requerimientos documentados puedan ser "implementables" (pasibles de ser analizados, diseñados, codificados) y "verificables" (pasibles de ser verificados y validados) en fases posteriores.

B.2. Entradas

- Minutas de reunión con el usuario.
- Documento de especificación de requerimientos del usuario.

B.3. Proceso

En la figura 6 se detalla gráficamente, el proceso a desarrollar durante este módulo.

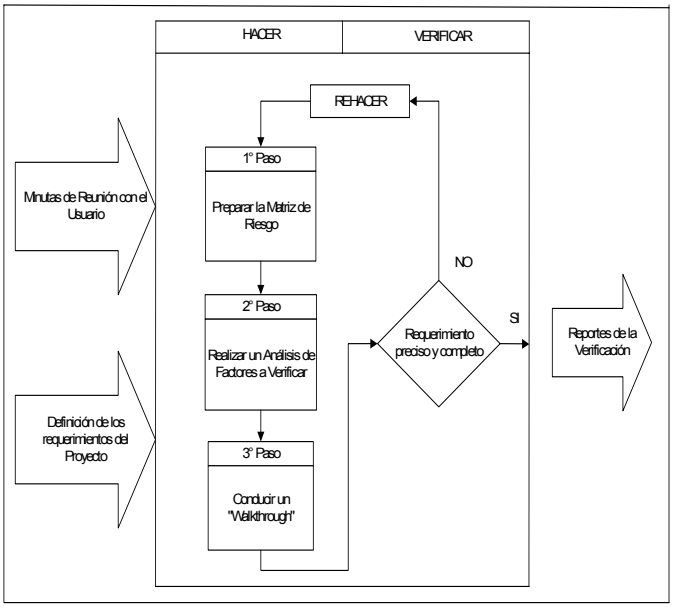

Fig. 6. Proceso del módulo de requerimientos

B.3.1. Preparar la Matriz de Riesgo

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

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

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

en la fase de diseño. La ejecución de la matriz de riesgo es un proceso de cinco pasos, los cuales se encuentran detallados en el capítulo de Técnicas y Herramientas.

B.3.2. Realizar un Análisis de Factores a Verificar

Debe llevarse a cabo un proceso para estimar las incumbencias ("concerns") asociadas a la fase de requerimientos del desarrollo del sistema. Debe incluirse un programa de verificación para cada factor a considerar (son 15), cubriendo cada fase del proceso de desarrollo. Para cada factor hay un programa de verificación que tiene en cuenta ciertas consideraciones las cuales se encuentran detalladas en el capítulo de Técnicas y Herramientas.

El programa de verificación enumera aquellos criterios, que aseguran al equipo de aseguramiento de la calidad, que la magnitud de esa preocupación es mínima. A estos criterios debe responder el equipo de aseguramiento de la calidad. También se debe realizar una verificación suficiente para evaluar si el equipo de proyecto ha manejado adecuadamente cada criterio de verificación.

Los factores a verificar en la etapa de requerimientos son:

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

B.3.3. Conducir un "Walkthrough" de Requerimientos

De los procesos de revisión, la inspección o recorrido (walkthrough) de requerimientos es el menos estructurado y el más propenso a la creatividad. Por lo tanto éste se convierte en un proceso de revisión que complementa los objetivos de la fase de requerimientos. El objeto de este proceso de revisión es crear una situación en la cual un grupo de gente con las habilidades adecuadas pueda ayudar al equipo en el desarrollo de las soluciones del proyecto. El "walkthrough" intenta usar la experiencia y juicio del equipo de revisión como un soporte en el proceso de desarrollo.

Este método se implementa como un proceso de cinco pasos ejecutados en secuencia, los cuales se encuentran detallados en el capítulo de Técnicas y Herramientas, que son:

- Establecer reglas básicas
- Seleccionar el equipo / Notificar a los participantes
- Presentación del proyecto
- Preguntas / recomendaciones
- Reporte final

El tiempo destinado a cada paso dependerá del tamaño de la aplicación que se esté revisando y el grado de asistencia deseado del equipo de "walkthrough".

B.4. Salidas

- Matriz de riesgo
- Análisis de factores de verificación.
- Informe de la inspección ("walkthrough") confeccionado.
- Resumen de la inspección ("walkthrough") confeccionado.

B.5. Lista de Verificación (Checklist)

En la tabla III se presenta la lista de verificación.

TABLA III. LISTA DE VERIFICACIÓN DEL MÓDULO DE REQUERIMIENTOS

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
1	¿Los requerimientos definidos son verificables?				
2	¿El usuario está de acuerdo con el requerimiento definido?				
3	¿Los desarrolladores entienden los requerimientos?				
4	¿El requerimiento definido coincide con los objetivos del proyecto?				
5	¿Se identificaron los riesgos del proyecto?				
6	¿Se siguió un proceso razonable en la definición del requerimiento?				
7	¿El proceso de control de requerimientos, es adecuado para minimizar los riesgos del proyecto?				
8	¿Durante el proceso de control de requerimientos, se ha llevado a cabo un "walktrough"?				

B.6. Métricas

- Estabilidad de los requerimientos: $\frac{\text{Cantidad de requerimientos iniciales}}{\text{Cantidad total de requerimientos}}$
- Completitud de los requerimientos: $\frac{\text{Cantidad de nuevos requerimientos agregados}}{\text{Cantidad total de requerimientos}}$
- Tasa de efectividad de la verificación: $\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$

B.7. Involucrados

B.7.1. Equipo de Ingeniería

- Gerente de proyecto.
- Grupo de análisis de requerimientos.

B.7.2. Equipo de Aseguramiento de la Calidad del Software

- Líder de aseguramiento de la calidad del software.

- Analistas de aseguramiento de la calidad del software Senior y/o Semi-senior.

C. Análisis

C.1. Objetivo

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

C.2. Entradas

- Documento de especificación de requerimientos revisado
- Documento de especificación funcional.

C.3. Proceso

En la figura C se detalla gráficamente, el proceso a desarrollar durante este módulo.

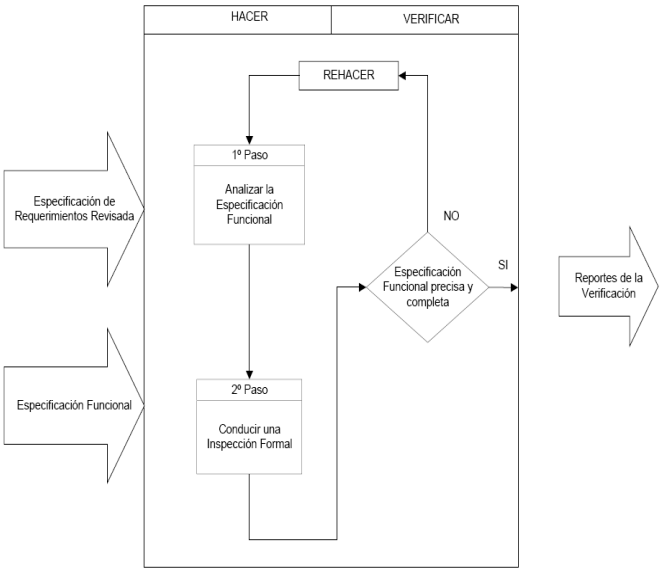

Fig. 7. Proceso del módulo de análisis

C.3.1. Analizar la Especificación Funcional

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

- Separar requerimientos funcionales de implementación.
- Desarrollar un modelo del comportamiento deseado del sistema, que comprenda datos y la respuesta funcional del sistema a los estímulos del ambiente.
- Establecer un contexto en el cual el software opera especificando la forma en la cual otros componentes del sistema interactúan con el software.
- Desarrollar un ambiente en el cual el sistema opere y reaccione a los estímulos del mismo.
- Crear un modelo cognitivo en lugar de un diseño o un modelo de implementación. El modelo cognitivo describe el sistema como es percibido por los usuarios de esa comunidad.
- Reconocer que una especificación tendrá varios niveles de detalle.

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

- El formato de la representación y contenido debería ser relevante para el problema. Debe llevarse a cabo, un desarrollo general para los contenidos de las especificaciones.
- La información contenida en la especificación debe ir de lo general a lo particular (estar anidada). Las representaciones deberían revelar capas de información a fin de permitir al lector que pueda moverse al nivel de detalle que sea requerido. Se deben indicar esquemas de numeración dentro de los diagramas utilizados, indicando el nivel de detalle que se presenta. Es importante presentar la misma información a diferentes niveles de abstracción para ayudar a su entendimiento.
- Los formatos diferentes de diagramas a utilizar y otras formas de notación, deberán ser restringidas al mínimo en número y consistentes en su uso. La información confusa e inconsistente, sea tanto simbólica o gráfica, perjudica el entendimiento y propicia errores.
- La especificación funcional, debe ser verificable.

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

I. Introducción
   a. Resumen del Sistema
   b. Descripción del producto
   c. Entregables
II. Ambiente
   a. Usuarios
   b. Servicios
   c. Físico
   d. Seguridad
III. Descripción de la información
   a. Representación del contenido de la información
   b. Representación del flujo de la información
      i. Flujo de datos
      ii. Flujo de control
IV. Descripción funcional
   a. Partición funcional
   b. Descripción funcional
      i. Narrativa de proceso general
      ii. Restricciones / Limitaciones
      iii. Requerimientos de rendimiento
      iv. Restricciones de diseño
      v. Diagramas de soporte
   c. Descripción de control
      i. Especificación de control
      ii. Restricciones de para el diseño
V. Descripción de situación
   a. Estado del sistema
   b. Eventos y acciones
VI. Validación y criterios
   a. Límites y restricciones
   b. Clases de pruebas requeridas
   c. Respuesta esperada del software
   d. Consideraciones especiales
VII. Cronograma
VIII. Apéndices

C.3.2. Conducir una Inspección Formal

La revisión de una especificación funcional es conducida por el desarrollador y el cliente. Dado que las especificaciones forman el punto fundamental y a partir del cual se realizará el diseño y subsecuentemente las actividades de la ingeniería del software o de conocimiento, se deberán extremar los cuidados para conducir esta revisión.

La revisión en una primera instancia es conducida en un nivel macro. En este nivel, los inspectores intentan garantizar que la especificación esté completa, consistente y exacta.

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

- Detectar y remplazar términos vagos.
- En el caso de listas de elementos, asegurar que hayan sido incluidos todos.
- Cuando se especifiquen rangos (numéricos, etc.) asegurarse que no contengan valores asumidos sin ser definidos expresamente
- Estar alerta de verbos y pronombres ambiguos.
- Identificar afirmaciones que no incluyan certeza.
- Cuando una estructura es descripta en palabras, asegurar que se realice un diagrama para ayudar a comprenderla.
- Cuando sea necesario realizar un cálculo especificado, asegurar que se presente al menos dos ejemplos del mismo.

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

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

C.4. Salidas

- Informe de la Inspección
- Resumen de la Inspección

C.5. Lista de Verificación (Checklist)

En la tabla IV se presenta la lista de verificación.

C.6. Métricas

- Estabilidad de los requerimientos: $\frac{\text{Cantidad de requerimientos modificados}}{\text{Cantidad total de requerimientos establecidos}}$
- Estabilidad de la funcionalidad: $\frac{\text{Cantidad de funciones modificadas}}{\text{Cantidad total de funciones establecidas}}$
- Correctitud de la funcionalidad: $\frac{\text{Cantidad de funciones corregidas}}{\text{Cantidad total de funciones establecidas}}$
- Completitud de la funcionalidad: $\frac{\text{Cantidad de nuevas funciones agregadas}}{\text{Cantidad total de funciones establecidas}}$
- Eficiencia en la detección de defectos: $\frac{\text{Cantidad de defectos detectados hasta análisis}}{\text{Cantidad total de defectos}}$
- Eficiencia en la remoción de defectos: $\frac{\text{Cantidad de defectos removidos hasta análisis}}{\text{Cantidad total de defectos}}$
- Tasa de efectividad de la verificación: $\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$

TABLA IV. LISTA DE VERIFICACIÓN DEL MÓDULO DE ANÁLISIS

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
1	¿Los objetivos y metas del sistema se mantienen consistentes con las políticas de software de la organización?				
2	¿La estructura y el flujo de la información, está adecuadamente definida por el área a la cual compete el problema?				
3	¿Son claros los diagramas? ¿Pueden ser explícitos sin necesidad de ser descriptos o narrados?				
4	¿Las funciones principales del sistema, están dentro del alcance? ¿Cada una de ellas ha sido adecuadamente definida?				
5	¿Es consistente el comportamiento del sistema con la información que debe procesar y las funciones que debe desarrollar?				
6	¿Las limitaciones del sistema son realistas?				
7	¿Ha sido considerado el riesgo tecnológico del desarrollo?				
8	¿Se han considerado requerimientos alternativos de software?				
9	¿Ha sido detallado el criterio de verificación y validación? ¿Los mismos, son adecuados para determinar el éxito del sistema?				
10	¿Existen inconsistencias, omisiones o redundancias en el modelo de información del sistema?				
11	¿El usuario final ha estado involucrado?				
12	¿El usuario revisó el prototipo y/o manual del usuario?				
13	¿Han sido debidamente planificadas las estimaciones de esfuerzo?				

C.7. Involucrados

C.7.1 .Equipo de Ingeniería

- Gerente de proyecto.
- Grupo de análisis de requerimientos.
- Grupo de análisis funcional.

C.7.2. Equipo de Aseguramiento de la Calidad del Software

- Líder de aseguramiento de la calidad del software.
- Analistas de aseguramiento de la calidad del software Senior y/o Semi-senior.

D. Diseño

D.1. Objetivo

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

D.2. Entradas

- Comprensión del diseño por parte del equipo de aseguramiento de la calidad
- Entregables derivados del diseño:
- Especificaciones de entrada
- Especificaciones de procesamiento
- Especificaciones de archivos
- Especificaciones de salida
- Especificaciones de control
- Diagramas de flujo del sistema
- Requerimientos de software y hardware.
- Especificación de los procedimientos de operación manual.
- Política de resguardo de la información.
- Documento de diseño de alto nivel (Lógico)
- Documento de diseño de detalle (Físico y lógico)

D.3. Proceso

En la figura 8 se detalla gráficamente, el proceso a desarrollar durante este módulo:

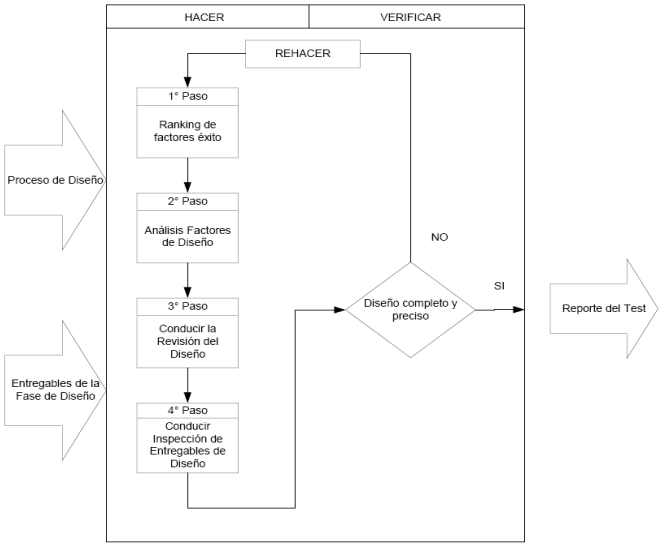

Fig. 8. Proceso del módulo de diseño

D.3.1. Ranking de Factores de Éxito

El ranking ("scoring") es una herramienta predictiva que utiliza experiencias en proyectos anteriores. Los proyectos en curso son analizados para determinar los atributos de los mismos y su correlación con el éxito o el fracaso de proyectos o sistemas en particular. Una vez que los atributos correlacionados al éxito o al fracaso pueden ser identificados, también pueden ser usados para predecir el comportamiento del sistema que está en desarrollo.

El concepto es el mismo que el usado para predecir el resultado de las elecciones. La gente que hace las predicciones procura identificar distritos electorales que muestran una correlación histórica alta en los resultados de elecciones anteriores. El conocimiento del registro de votos de un distrito en especial y los resultados de elecciones previas, nos provee la base para hacer futuras predicciones. Esos distritos con un registro de grandes ganadores pueden servir de muestra, para luego basados en esas muestras, predecir el resultado de una elección antes que los votos sean contados.

Estos tipos de predicciones en sistemas de computación son bien conocidos, pero no han sido bien utilizados hasta que el concepto de ranking fue desarrollado. Por ejemplo, sabemos que hay una correlación positiva entre la participación del usuario y el sistema de computación: cuanto más involucrado este el usuario, mayor probabilidad de que el sistema sea exitoso. También sabemos que habrá problemas con el sistema de computación que empuja las actuales corrientes de tecnología. Por ejemplo, el primer sistema en usar la nueva tecnología puede ser identificada como un sistema de alto riesgo, porque la mayoría de los inconvenientes ocurrirá durante la implementación.

Los criterios que se utilizan bajo el concepto de ranking, se describen en el capítulo de Técnicas y Herramientas

Al concluir el ranking, el resultado puede ser usado en cualquiera de las siguientes formas:

- Estimar la extensión de la verificación: A mayor riesgo, la gerencia del proyecto puede requerir más verificaciones. Sabiendo que la aplicación es de alto riesgo, se alertará al los gerente del proyecto respecto de la necesidad de tomar las medidas correspondientes para reducir el riesgo a un nivel aceptable.
- Identificar módulos a verificar: Dependiendo de la sofisticación del instrumento con el que se obtuvo el ranking, puede identificarse módulos específicos para la verificación. Por ejemplo, si la lógica computacional se sabe que es de alto riesgo, la verificación debería evaluar exhaustivamente la exactitud de este proceso.
- Identificar la composición del equipo de verificación: Los tipos de riesgos asociados con el sistema ayudan a determinar la composición del equipo de test. Por ejemplo, si los riesgos tratan más con la tecnología que con la lógica, el equipo de test debería incluir individuos con conocimientos profundos en esa tecnología.

Al ranking de riesgo se llega a través de un desarrollo matemático, como se muestra en el capítulo de Técnicas y Herramientas.

D.3.2. Análisis de Factores

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

- Desarrollar una solución al problema de negocio planteado.
- Determinar el rol de la tecnología para resolver el problema.
- Desarrollar especificaciones para los segmentos manuales y automatizados del sistema.
- Cumplir con el plan de acción, procedimientos, estándares y regulaciones.
- Definir los controles que reducirán riesgos de la aplicación, llevándolos a un nivel aceptable.

- Completar el proyecto dentro de las restricciones del presupuesto, personal y cronograma establecidos.

Los factores que deben analizarse durante la fase de diseño se describen a continuación, y puede encontrarse más detalle en el capítulo de Técnicas y Herramientas: Los factores a verificar en la Fase de Diseño son:

- Controles de integridad de datos.
- Reglas de autorización.
- Controles de integridad de archivos.
- Pistas de auditoria.
- Plan de contingencia.
- Método para alcanzar el nivel de servicio requerido.
- Procedimientos de acceso.
- Diseño acorde con la metodología establecida.
- Diseño acorde con los requerimientos establecidos.
- Facilidad de uso.
- Mantenibilidad del diseño.
- Portabilidad del diseño.
- Interfaces de diseño.
- Diseño acorde con criterios establecidos.
- Necesidades de operación.

D.3.3. Conducción de la Revisión del Diseño

La revisión de diseño será estructurada usando la misma información básica que la utilizada para llevar a cabo el ranking. El objetivo es identificar de antemano aquellos atributos del diseño que se correlacionan con los problemas del sistema. Entonces durante la revisión del diseño, se investigarán dichos atributos para determinar si el equipo de proyecto los ha tratado apropiadamente.

La revisión del diseño deberá ser conducida por un grupo de individuos con conocimientos en el proceso de diseño. El grupo tendrá la responsabilidad de revisar la integridad y racionabilidad de la aplicación. No es necesario que el grupo conozca la aplicación específicamente pero sí debe conocer la metodología de la organización.

La revisión de diseño normalmente es formal y altamente estructurada. Los miembros del equipo intentan determinar si todas las tareas se han realizado apropiadamente. Al final de la revisión de diseño, el equipo emite un reporte indicando sus hallazgos y recomendaciones acerca del proyecto. El equipo de revisión de diseño comprende los siguientes miembros:

- Personal del proyecto.
- Equipo de revisión independiente.

La siguiente es la lista con los pasos de una revisión de diseño, la descripción detallada está en el capítulo de Técnicas y Herramientas:

- Seleccionar el equipo de revisión.
- Entrenar los miembros del equipo de revisión.
- Notificar al equipo de proyecto.
- Asignar tiempos adecuados.
- Documentar los datos de la revisión.
- Revisar los datos con el equipo de proyecto.
- Desarrollar recomendaciones de revisión.
- Revisar recomendaciones con el equipo de proyecto.
- Preparar un reporte final.

Una o más revisiones pueden llevarse a cabo durante la fase de diseño. La cantidad de revisiones dependerá de la importancia del proyecto y del lapso de tiempo en la fase de diseño. Así podrán llevarse a cabo revisiones del diseño de alto nivel y del diseño detallado. Cada defecto descubierto durante la revisión del diseño de alto nivel debe documentarse. Se confeccionará un reporte de defectos. Los defectos deben documentarse, categorizarse, registrarse, presentarse al equipo de proyecto para corregir, y referenciarlo al documento específico en el cual el defecto fue notado. Un ejemplo del formulario para registrar estos defectos, puede encontrarse en el capítulo de Técnicas y Herramientas.

D.3.4. Conducción de la Inspección de los Entregables de Diseño

La Inspección es un proceso por el cual los productos completos pero no verificados, son evaluados para saber si los cambios especificados fueron implementados correctamente.

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

D.4. Salidas

- Informe de la inspección.
- Resumen de la inspección.

D.5. Lista de Verificación (Checklist)

En la tabla 4.4 se presenta la lista de verificación.

D.6. Métricas

- Eficiencia en la detección de defectos: $\frac{\text{Cantidad de defectos detectados hasta diseño}}{\text{Cantidad total de defectos hasta diseño}}$
- Eficiencia en la remoción de defectos: $\frac{\text{Cantidad de defectos removidos hasta diseño}}{\text{Cantidad total de defectos hasta diseño}}$
- Tasa de efectividad de la verificación: $\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$
- Estabilidad del diseño: $\frac{\text{Cantidad de cambios introducidas en el diseño}}{\text{Cantidad total de cambios en el proyecto}}$

D.7. Involucrados

D.7.1. Equipo de Ingeniería

- Gerente de proyecto.
- Grupo de diseño de alto nivel (o lógico).
- Grupo de diseño de detalle (o físico).
- Grupo de administración de base de datos.
- Grupo de analistas desarrolladores.

D.7.2. Equipo de Aseguramiento de la Calidad del Software

- Líder de Aseguramiento de la Calidad del Software
- Analistas de aseguramiento de la calidad del software Senior y/o Semi-senior.

E. Codificación

E.1. Objetivo

El principal objetivo de esta prueba es asegurar que las especificaciones de diseño hayan sido correctamente implementadas.

E.2. Entradas

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

TABLA V. LISTA DE VERIFICACIÓN DEL MÓDULO DE DISEÑO

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
1	¿El equipo de aseguramiento de la calidad tiene buenos conocimientos acerca del proceso de diseño?				
2	Si se han utilizado herramientas automatizadas en la creación del diseño, ¿las conocen los miembros del equipo de aseguramiento de la calidad?				
3	¿Han recibido los miembros del equipo de aseguramiento de la calidad todos los entregables de la fase necesarios para llevar a cabo las verificaciones correspondientes?				
4	¿Los usuarios están de acuerdo con que el diseño representa la realidad?				
5	¿El equipo de proyecto cree que el diseño representa la realidad?				
6	¿Han identificado los miembros del equipo de aseguramiento de la calidad los factores de éxito, tanto positivos como negativos, que pudieran afectar el éxito del diseño?				
7	¿Han utilizado los miembros del equipo de aseguramiento de la calidad, tales factores en la puntuación de las probabilidades de éxito?				
8	¿Entienden los miembros del equipo de aseguramiento de la calidad los factores relacionados con el diseño?				
9	¿Han analizado los miembros del equipo de aseguramiento de la calidad los factores de test del diseño para evaluar el potencial impacto sobre el éxito del mismo?				
10	¿Ha sido instruido el equipo de Revisión acerca de lo que representa el éxito del Diseño?				
11	¿La Gerencia apoya el uso del proceso de revisión del diseño?				
12	¿El proceso de revisión del diseño fue conducido en un momento apropiado?				
13	¿Son razonables los ítems identificados en el proceso de revisión del diseño?				
14	¿Están de acuerdo los miembros del equipo de aseguramiento de la calidad que los ítems identificados necesitan ser verificados?				
15	¿La Gerencia apoya la ejecución de inspecciones en el proyecto?				
16	¿Se ha provisto del tiempo suficiente en el cronograma del proyecto para realizar inspecciones?				
17	¿Han sido instruidos los responsables del proyecto acerca de la importancia de la participación en las inspecciones?				
18	¿La Gerencia ve las inspecciones como una parte integral del proceso, en lugar de tomarlo como una auditoría al desempeño de los participantes?				
19	¿Han sido planificados los procesos de Inspección?				
20	¿Han sido identificados los inspectores y asignado sus roles específicos?				
21	¿Han sido entrenados los inspectores para cumplir con su rol?				
22	¿Se les ha dado a los inspectores los materiales necesarios para cumplir con la inspección?				
23	¿Se les ha dado a los inspectores tiempo adecuado para realizar las tareas habituales que le permitirán cumplir con la preparación y la reunión para el proceso de inspección?				
24	¿Se han preparado adecuadamente los inspectores para la inspección?				
25	¿Han preparado los inspectores una lista de defectos?				
26	¿Se ha programado la inspección convenientemente para todos los inspectores?				
27	¿Han concurrido todos los inspectores a la reunión de inspección?				
28	¿Estuvieron de acuerdo todos los inspectores con la lista de defectos?				
29	¿Fueron identificados los defectos durante la reunión de revisión registrados y entregados al autor?				
30	¿El autor estuvo de acuerdo acerca de realizar las correcciones necesarias?				
31	¿Se ha desarrollado un proceso razonable para determinar si esos defectos han sido corregidos satisfactoriamente?				
32	¿La certificación del moderador ha sido tomada en cuenta para el producto / entregable inspeccionado?				

- Especificaciones de programas.
- Documentación de programas.
- Listado de código.
- Programas ejecutables.
- Diagramas de flujo de los programas.
- Instrucciones para el operador.

E.3. Proceso

En la figura 9 se detalla gráficamente, el proceso a desarrollar durante este módulo:

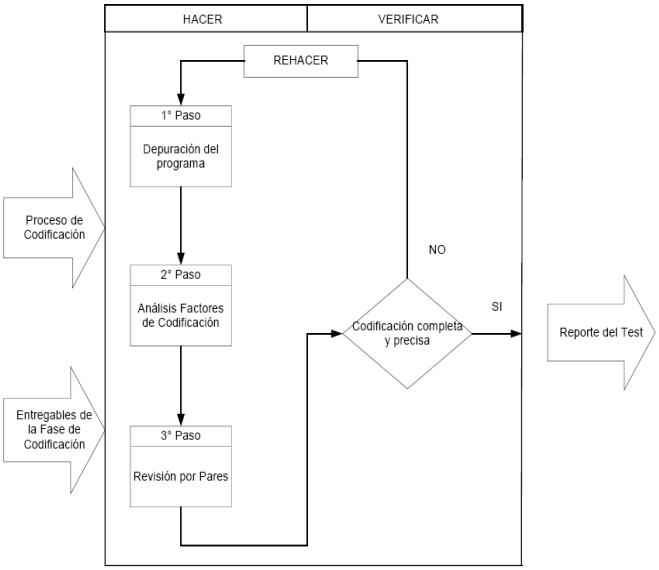

Fig. 9. Proceso del módulo de codificación

E.3.1. Depuración de Programas

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

- Depuración de errores sintácticos.
- Depuración de errores estructurales.
- Depuración de errores funcionales.

El detalle de cada una de ellas está en el capítulo de Técnicas y Herramientas.

E.3.2. Análisis de Factores de Codificación

A continuación, se detallan los factores a analizar. El detalle de cada uno de ellos está en el capítulo de Técnicas y Herramientas:

- Implementación del control de integridad de información.
- Implementación de reglas de autorización.
- Implementación de control de integridad de archivos.
- Implementación de auditorías de rastreo.
- Plan de contingencia escrito.
- Consecución del nivel de servicio deseado para el sistema.
- Implementación de procedimientos de seguridad.
- Cumplimiento del programa con la metodología a adoptar.
- Programa acorde al diseño (Correctitud).
- Programa acorde al diseño (Facilidad de uso).
- Mantenimiento del programa.
- Programa acorde al diseño (Portabilidad).
- Programa acorde al diseño (Acoplamiento).
- Desarrollo de procedimientos operativos.
- Criterio de ejecución del programa (Performance).

E.3.3. Conducción de Revisiones de Pares

Se utiliza la revisión por pares, para evaluar la estructura y funcionalidad del programa. La revisión por pares detecta errores sintácticos, más por observación personal que por el resultado directo del "Walktrough".

E.4. Salidas

Dos son las salidas de este proceso

- La eliminación de todos los errores de codificación utilizando pruebas estáticas.
- Listado de errores encontrados.

E.5. Lista de Verificación (Checklist)

En la tabla VI se presenta la lista de verificación:

TABLA VI. LISTA DE VERIFICACIÓN DEL MÓDULO DE CODIFICACIÓN

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
1	¿Es considerada una responsabilidad del programador la verificación y la validación de los programas?				
2	¿El programador entiende la diferencia entre testing estático y dinámico?				
3	¿El programa podría ser sometido a testing estático como medio primario para remover defectos?				
4	¿El programador entiende el proceso que generará el código del programa?				
5	¿El programador entiende y usa la técnica de "debugging"?				
6	¿El programador entiende los conceptos implicados, y los tendrá presentes en la verificación de la programación?				
7	¿El programador es verificado usando revisión por pares o inspección de código?				
8	¿El programa será sometido a un test completo antes de ingresar a un nivel superior de test?				
9	¿Todos los defectos no cubiertos están registrados en detalle?				
10	¿Todos los defectos no cubiertos fueron corregidos antes de ingresar al siguiente nivel de verificación?				

E.6. Métricas

- Densidad de defectos por componente: $\frac{\text{Cantidad de defectos}}{\text{Tamaño del componente}}$

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

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

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

E.7. Involucrados

E.7.1. Equipo de Ingeniería

- Gerente de proyecto.
- Grupo de programación.

E.7.2 Equipo de Aseguramiento de la Calidad del Software

- Líder de Aseguramiento de la Calidad del Software
- Analistas de aseguramiento de la calidad del software Senior y/o Semi-senior.

F. Verificación del Sistema

F.1. Objetivo

El objetivo es determinar si el software funcionará correctamente cuando se ejecute. Para ello:

- Se verificará que se haya probado la ejecución del software en un ambiente separado (test), siendo aproximadamente el mismo entorno operacional en que será ejecutado luego (producción).
- Las pruebas serán verificadas, una vez terminada su ejecución y aprobación interna.
- Cualquier desviación de los resultados esperados será reportada.

Dependiendo de la severidad y naturaleza de dichos problemas, podrían llegar a realizarse solicitudes de cambios al sistema antes de su puesta en producción.

F.2. Entradas

- Planes de pruebas:
- Pruebas en línea ("On-Line")
- Pruebas por lotes ("Batch")
- Pruebas de interfaces.
- Pruebas de regresión.
- Pruebas de integración.
- Pruebas de "stress".
- Datos de prueba.
- Resultados de las pruebas.

F.3. Proceso

En la figura 10 se detalla gráficamente, el proceso a desarrollar durante este módulo:

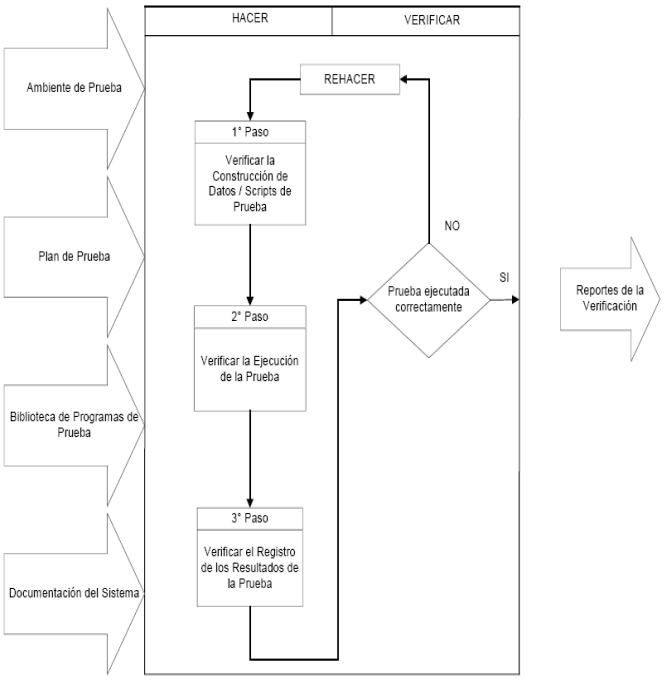

Fig. 10. Proceso del módulo de verificación

F.3.1. Verificar la Construcción de Datos / "Scripts" de Prueba

El concepto de construcción de datos de prueba es, simplemente, la creación de un proceso representativo de la realidad usando transacciones ficticias. De dichos datos, se tomará una muestra representativa para llevar a cabo una verificación de los mismos. La correcta selección de dicha muestra es la clave para el éxito de esta tarea. Dentro de esta tarea debe abarcarse la verificación de los siguientes:

Diseño de archivos de prueba

Estos archivos deben involucrar transacciones normales, transacciones con datos inválidos, y transacciones que puedan hacer incurrir al sistema en alguna situación de excepción.

Ingreso de los datos de prueba

Después que las transacciones de prueba estén definidas, se deben introducir en el sistema todos los datos necesarios para que las mismas puedan ser procesadas.

Análisis de los resultados esperados

Antes de procesar alguna transacción, el equipo de proyecto debe establecer el resultado esperado para cada transacción a probar para poder compararla con el obtenido luego de realizada la misma.

Prueba de transacciones

Para la prueba mencionada se recomiendan los siguientes 9 pasos:

- Identificar los recursos de prueba.
- Identificar las condiciones de prueba.
- Priorizar las condiciones.
- Seleccionar las condiciones.
- Determinar los resultados esperados.
- Crear transacciones de prueba.
- Documentar las condiciones de prueba.
- Ejecutar la prueba
- Verificar y corregir.

Prueba de volumen

El objetivo de dicha prueba es verificar que el sistema pueda realizar adecuadamente las transacciones cuando sus programas internos o sus límites se vean excedidos. Estas pruebas suelen involucrar grandes volúmenes de datos. Los pasos que se recomiendan para esta prueba son:

- Identificar las entradas del sistema.
- Identificar las salidas del sistema.
- Llevar cada dato a su máxima expresión tratando de sobrepasar sus límites.
- Documentar las limitaciones.
- Realizar la prueba de volumen.

Creación de casos de prueba

- Determinar el nivel de la prueba: Unitaria, concurrencia, integración, regresión o "stress".
- Desarrollar los casos de prueba: Un caso de prueba, son todos los pasos que deben seguirse en el sistema para probar dicho caso.
- Ejecutar los casos de prueba: Usar el sistema con los casos desarrollados en el punto anterior.
- Analizar los resultados.
- Mantener los casos de prueba.

F.3.2. Verificar la Ejecución de la Prueba

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

Durante esta fase, el equipo de aseguramiento de la calidad, llevará a cabo tanto la verificación de los planes de prueba, como la verificación de la ejecución de los mismos. Todo esto se realizará, tomando muestras representativas de los planes y sus respectivas ejecuciones.

Se verificará también, que los siguientes tipos de prueba hayan sido llevados a cabo por el equipo de proyecto durante esta fase:

Prueba manual, de regresión y funcional

La prueba manual asegura que las personas que interactúen con el sistema van a poder realizar las operaciones propuestas; la prueba de regresión es la verificación de que lo que se está instalando no afecta a lo que ya estaba instalado; y la prueba funcional apunta a que los requerimientos del sistema puedan ser ejecutados correctamente en diversas circunstancias.

Prueba de autorización

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

Prueba de integridad

Debe ser verificada durante la prueba, los controles sobre la integridad de los archivos de datos.

Prueba de pistas de auditoria

La función de pistas de auditoria, debe ser probada para asegurarse que para cualquier transacción, pueda realizársele un seguimiento desde su inicio hasta su fin, para lograr una correcta reconstrucción de la misma cuando sea necesario.

Prueba de recuperación

Si el procesamiento debe continuar aunque el sistema no esté operativo, puede llegar a tener que provocarse fallas intencionales en el sistema, para poder realizarse la prueba de continuidad, la cual implica lo antedicho.

Prueba de "stress"

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

Prueba de seguridad

En esta prueba lo que se intenta es violar todos los aspectos de seguridad del sistema queriendo obtener como resultado que ninguno de estos falle.

Prueba acorde a la metodología

Todo tipo de prueba, deberá ser llevado a cabo de acuerdo a las políticas, estándares y procedimientos establecidos en la metodología de la organización. Esta última debería especificar el plan de prueba requerido para cada tipo de prueba, las técnicas y herramientas de prueba recomendadas, así como el tipo de documentación requerida a lo largo de la prueba misma. La metodología debería especificar el modo de determinar cuándo una prueba fue exitosa o no.

Prueba funcional

Esta prueba verifica el correcto cumplimiento de los requerimientos del usuario.

Prueba de operación

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

Inspecciones

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

Prueba de desastre

En esta prueba lo que se hace es provocar problemas en el ambiente real de la aplicación ya que es la única manera de ver realmente cómo se comporta el sistema ante situaciones inesperadas.

Prueba funcional y de regresión

Esta fase de la prueba debe verificar la comunicación con los sistemas relacionados. Donde, la prueba funcional verifica la interconexión de la nueva funcionalidad, y la prueba de regresión verifica la continuidad del funcionamiento del resto de los subsistemas asociados preexistentes.

Prueba de performance

Los criterios de performance son establecidos durante la fase de requerimientos e intentan ser medidos en la fase de verificación. De no poder realizarse porque el ambiente necesario para llevarla a cabo sería muy costoso, se dejará para ser medido en producción.

Prueba de facilidad de operación

Esta prueba generalmente es realizada por el equipo de Operaciones que tratará normalmente con el sistema y se intenta que el personal del grupo de proyecto no provea ningún tipo de instrucciones a los usuarios, para que pueda reproducirse el normal desarrollo de las transacciones, tal como sería en el ambiente de producción.

F.3.3. Verificar el Registro de los Resultados de la Prueba

Ya que documentar un problema, es el primer paso para la corrección del mismo, es necesario llevar adelante una verificación del registro de los resultados de la prueba. Esto se llevará a cabo, tomando muestras representativas de tales resultados, para su análisis posterior.

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

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

Para lograr una buena documentación de los problemas, es aconsejable seguir estos tres pasos:

- Documentar el desvío: Esencialmente el usuario compara "lo que es" con "lo que debería ser", y cuando es identificada una desviación en este sentido, corresponde comenzar a desarrollar la declaración del problema.
- Documentar el/ los efecto/s: Aquí debe evaluarse lo que el problema significa para el sistema. La atención que se le dará al mismo, estará directamente relacionada con lo declarado en el/ los efecto/s.
- Documentar la o las causas: La o las causas son la o las razones subyacentes para la condición. En algunos casos la causa puede ser obvia a través de los hechos. Pero en otros, puede ser necesaria una investigación para encontrarla.

F.4. Salidas

- Muestra de planes de prueba verificados.
- Muestra de transacciones de prueba verificadas.
- Muestra de resultados de la ejecución de dichas transacciones verificados.
- Desvío de los resultados esperados documentados.

F.5. Lista de Verificación (Checklist)

En la tabla VII se presenta la lista de verificación:

TABLA VII. LISTA DE VERIFICACIÓN DEL MÓDULO DE VERIFICACIÓN

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
1	¿Todos los pasos fueron realizados como se especificaron?				
2	Si los pasos no fueron realizados como se especificaron. ¿Se afectaron recursos adicionales para resolver defectos durante la etapa de ejecución?				
3	¿Se estableció un ambiente de prueba apropiado para realizar la prueba del software?				
4	¿Se entrenó analistas de calidad en las herramientas de verificación que serán utilizadas durante esta etapa?				
5	¿Se fijó un tiempo adecuado para esta etapa?				
6	¿Se fijaron los recursos adecuados para esta etapa?				
7	¿Los métodos para crear datos de prueba son apropiados para el sistema?				
8	¿Fueron creados los datos de prueba necesarios para probar adecuadamente el software?				
9	¿Fueron programadas todas las técnicas de verificación indicadas en el plan de prueba para ser ejecutadas durante este paso? (Ej. Prueba de regresión)				
10	¿Fueron determinados los resultados esperados de las pruebas?				
11	¿Se ha establecido el proceso por el cual se determina el desvío entre los resultados esperados y los actuales?				
12	¿Se han documentado los resultados esperados y los actuales cuando existe una diferencia entre ellos?				

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
13	¿Se determinó el potencial impacto de una desviación? (Ej. Problema en la verificación)				
14	¿Se ha establecido un procedimiento para asegurar las acciones / resolución apropiada de los defectos?				

F.6. Métricas

- Eficiencia en la detección de defectos: $\frac{\text{Cantidad de defectos detectados hasta verificación}}{\text{Cantidad total de defectos hasta verificación}}$
- Eficiencia en la remoción de defectos: $\frac{\text{Cantidad de defectos removidos hasta verificación}}{\text{Cantidad total de defectos hasta verificación}}$
- Tasa de efectividad de la verificación: $\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$
- Antigüedad media de defectos pendientes: $\frac{\text{Total de antigüedad de defectos pendientes al fin de un período}}{\text{Cantidad total de defectos pendientes al fin de un período}}$
- Antigüedad media de defectos cerrados: $\frac{\text{Total de antigüedad de defectos cerrados al fin de un período}}{\text{Cantidad total de defectos cerrados al fin de un período}}$

F.7. Involucrados

F.7.1. Equipo de Ingeniería

- Gerente de proyecto.
- Representante del grupo de análisis de requerimientos.
- Representante del grupo de análisis funcional.
- Representante del grupo de diseño.
- Representante del grupo de analistas desarrolladores.
- Grupo de testeo.

F.7.2. Equipo de Aseguramiento de la Calidad del Software

- Líder de Aseguramiento de la Calidad del Software
- Analistas de aseguramiento de la calidad del software Senior y/o Semi-senior.

G. Validación del Sistema

G.1. Objetivo

Describir los procedimientos para identificar los criterios de aceptación para el ciclo de vida de los productos y para validarlos. La aceptación final no solo tiene en cuenta que el producto de software resuelva adecuadamente los requerimientos de los usuarios, sino también, reconocer que el proceso de desarrollo fue adecuado. Como parte del proceso de ciclo de vida, la aceptación del software permite:

- Detección temprana de los problemas del software.
- Preparación apropiada de los medios para realizar la prueba.
- Consideración temprana de las necesidades del usuario durante el desarrollo del software.

G.2. Entradas

Las entradas dependen del alcance asignado a la validación del sistema (prueba de aceptación). Estas suelen ser:

- Software probado: Una vez realizada la verificación de las piezas de software se comienza con la prueba de aceptación.
- Plan de Prueba de Aceptación:
- Pruebas en Línea ("On-Line").
- Pruebas por Lotes ("Batch").
- Pruebas de Interfaces.
- Pruebas de Integración.
- Pruebas de "stress".
- Pruebas de conectividad.
- Pruebas de servidores.
- Lista de defectos no resueltos: Quizás no sea prudente esperar hasta que todos los defectos reportados sean corregidos antes de empezar las pruebas de aceptación. En este caso, los encargados de efectuar las pruebas de aceptación reciben una lista no resuelta de defectos, la cual les permitirá conocer los procesos incorrectos y así enfocar la atención en los criterios principales de aceptación antes de la prueba.

G.3. Proceso

En la figura 11 se detalla gráficamente, el proceso a desarrollar durante este módulo.

G.3.1. Verificar los Criterios de Aceptación Definidos

El usuario deberá definir el o los criterios que el software deberá cumplir para ser aceptado.

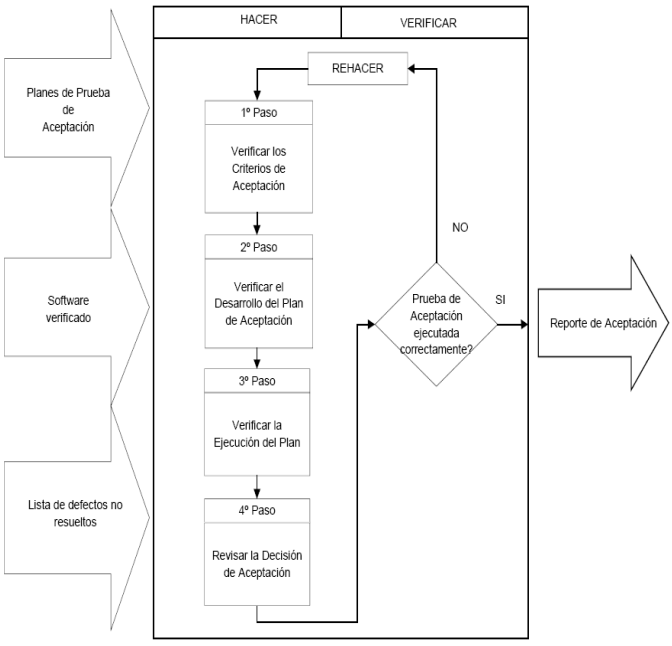

Fig. 11. Proceso del módulo de validación

Los requerimientos de aceptación que el sistema debe cumplir, pueden ser divididos en las siguientes cuatro categorías:

- Requerimientos Funcionales: Referido a las reglas de negocio que el sistema debe ejecutar.

- Requerimientos de Performance: Referido a los tiempos y recursos de operación.
- Requerimientos de Interfaz: Referido a la relación humano / máquina, máquina / módulo.
- Requerimientos globales de Calidad del Software: Referidos a la obtención de los límites para los factores o los atributos tales como la fiabilidad, comprobabilidad, exactitud, y usabilidad.

Estos criterios deberían estar enunciados y documentados durante la fase de requerimientos, para ser refinados y detallados en las fases siguientes. Se procederá a solicitar la recopilación de estos requerimientos, luego el equipo de aseguramiento de la calidad, procederá a verificarlos.

Se debería determinar con el usuario, el grado de criticidad de los requerimientos para los ítems cuyas categorías se presentan en la tabla VIII.

G.3.2. Verificar el Desarrollo del Plan de Aceptación

El primer paso para completar la aceptación del software es el desarrollo simultáneo de un plan de aceptación, un plan de proyecto y de los requerimientos del software que aseguran que las necesidades del usuario están representadas correcta y completamente. Este desarrollo simultáneo proveerá una visión general que asegurará que todo los recursos necesarios, estarán incluidos en el plan de proyecto.

En la tabla IX se detallan los puntos a verificar dentro de un plan de aceptación típico.

TABLA VIII. ÍTEMS DE REQUERIMIENTOS PARA DETERMINAR SU CRITICIDAD

Categoría	Ítems
Funcionalidad	▪ Consistencia interna de documentos, código, y entre las fases. ▪ Trazabilidad (posibilidad de seguimiento) de la funcionalidad. ▪ Verificación adecuada de la lógica. ▪ Preservación de funcionalidad en el ambiente productivo.
Performance	▪ Análisis de factibilidad de requerimientos de performance. ▪ Herramientas de simulación. ▪ Análisis de desempeño en el ambiente productivo.
Interfaz	▪ Documentación de la interfaz. ▪ Complejidad de la interfaz. ▪ Planes de prueba e integración de la interfaz. ▪ Ergonomía de la interfaz. ▪ Prueba de la interfaz en el ambiente productivo.
Calidad global del software	▪ Cuantificación de medidas de calidad. ▪ Criterios para la aceptación de los productos de software. ▪ Suficiencia de documentación y normas de desarrollo de sistemas. ▪ Criterios de Calidad para la Prueba Operacional.

TABLA IX. PUNTOS A VERIFICAR DENTRO DE UN PLAN DE ACEPTACIÓN TÍPICO

Contenido	Descripción
Descripción del proyecto	▪ Tipo de sistema. ▪ Ciclo de vida definido por la metodología de la organización. ▪ Usuarios del sistema. ▪ Principales tareas que debe satisfacer el sistema. ▪ Principales interfaces externas del sistema.
	▪ Uso normal esperado. ▪ Potenciales usos indebidos. ▪ Riesgos. ▪ Restricciones. ▪ Estándares y Normas.
Responsabilidades del usuario	▪ Organización y responsabilidades para las actividades de aceptación del sistema. ▪ Recursos y cronograma de requerimientos. ▪ Requerimientos del recurso. ▪ Requerimientos para soporte automatizado, datos especiales, y entrenamiento. ▪ Normas, estándares y convenciones. ▪ Actualización y revisión de los planes de aceptación.
Procedimientos administrativos	▪ Informes de anomalías. ▪ Control de cambios. ▪ Resguardo de información.
Descripción de aceptación	▪ Objetivos para el proyecto. ▪ Resumen de criterios de aceptación. ▪ Principales actividades de aceptación y revisiones. ▪ Requerimientos de información. ▪ Tipos de decisiones de aceptación. ▪ Responsabilidad de las decisiones de aceptación.

G.3.3. Verificar la Ejecución del Plan de Aceptación

El objetivo de este paso es, verificar el cumplimiento de los criterios de aceptación, en el producto entregado. Esto se puede llevar a cabo a través de revisiones, que involucran verificar productos provisorios y entregables desarrollados parcialmente, a lo largo del proceso de desarrollo.

El criterio de aceptación del sistema, debería estar especificado formalmente en el plan del proyecto. El plan identifica los productos a ser verificados, el criterio específico de aprobación/rechazo, las revisiones, y los tipos de verificación que se harán durante el ciclo de vida completo.

Las decisiones de aceptación necesitan una estructura sobre la cual operar, algunos componentes son: contratos, criterios de aceptación y mecanismos formales. La aceptación del software deberá referirse a un criterio específico que los productos deben tener para ser aceptados. El principal medio para alcanzar la aceptación en el desarrollo de los sistemas de software, es mantener una revisión periódica de la documentación y productos de software.

Cuando la decisión de aceptación requiere cambios, se hace necesaria otra revisión para asegurarse de que los cambios solicitados hayan sido configurados e implementados en forma correcta, y que el efecto en cualquier otra parte sea aceptable.

Las actividades de aceptación, pueden incluir la verificación de las piezas de software. La verificación de aceptación del software, se vuelve formal, cuando durante el ciclo de vida del desarrollo, el usuario acepta o rechaza el mismo. Esto es como un requerimiento contractual entre el usuario y el equipo del proyecto. El rechazo, generalmente, significa que se deberá realizar un trabajo adicional en el sistema a fin de hacerlo aceptable para el usuario. La prueba de aceptación final, es la última oportunidad que tiene el usuario para examinar la funcionalidad, interfaz, performance y aspectos de calidad antes de la revisión final de aceptación. En ese momento, el sistema deberá incluir el software a entregar, toda la documentación de usuario y las versiones finales de otros entregables del sistema.

G.3.4. Revisar la Decisión de Aceptación

La aceptación de un sistema generalmente significa que el proyecto se ha completado, con la excepción de alguna contingencia.

Las decisiones de aceptación pueden involucrar alguna de las siguientes situaciones:

- Los cambios solicitados son aceptados antes de pasar a desarrollar la próxima actividad.
- Algunos cambios, deben ser aceptados y llevados a cabo antes de continuar con el desarrollo del producto, y otros, pueden ser postergados hasta la próxima revisión del producto.
- El desarrollo puede continuar y los cambios podrán ser aceptados en la próxima revisión.
- No hay cambios solicitados y el desarrollo puede continuar.

Cada una de estas situaciones, será verificada por el equipo de aseguramiento de la calidad, teniendo en cuenta los defectos no resueltos, el cronograma del proyecto, el estado actual del mismo, los recursos involucrados, etc.

G.4. Salidas

Informe de la decisión de Aceptación.

G.5. Lista de Verificación (Checklist)

En la tabla X se presenta la lista de verificación.

TABLA X. LISTA DE VERIFICACIÓN DEL MÓDULO DE VALIDACIÓN

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
1	¿Se incorporó la prueba de aceptación al plan general de pruebas?				
2	¿La prueba de aceptación está enfocada como un proceso del proyecto en lugar de un paso aislado al final del testing?				
3	¿Fueron seleccionados los usuarios correctos para determinar los criterios de aceptación para el software y/o componentes de hardware?				
4	¿El grupo que define los criterios de aceptación, es representativo de los usos del sistema a ser probado?				
5	¿Esos individuos aceptan la responsabilidad de identificar los criterios de aceptación?				
6	¿Los criterios de aceptación fueron identificados tempranamente como para lograr influir en el planteo e implementación de la solución?				
7	¿Se ha desarrollado y escrito el plan de aceptación?				
8	¿El plan de aceptación es consistente con los criterios de aceptación?				
9	¿Se están verificando los productos intermedios por parte de los testeadores, antes de ser utilizados para la siguiente tarea de implementación?				
10	¿Se han seleccionado las técnicas de prueba apropiadas para el test de aceptación?				
11	¿Los testeadores tienen el nivel de conocimiento necesario para realizar dicha tarea?				
12	¿Se afectaron los recursos necesarios para la realización de la prueba de aceptación?				
13	¿Se destinó el tiempo necesario para la realización de la prueba de aceptación?				
14	¿Se han publicado las opiniones internas de la prueba de aceptación?				
15	¿Fue buena la reacción del equipo del proyecto en lo que concierne a los testeadores en la prueba de aceptación?				
16	¿Se ha tomado la decisión final de aceptación del sistema?				
17	¿Se han identificado los criterios de aceptación críticos?				
18	¿Los requerimientos fueron escritos con el detalle suficiente, de forma tal de poder obtener a partir de ellos las interfaces?				
19	¿Están los responsables de cada interfaz, descritos en el diagrama?				
20	¿Se ha definido un caso de prueba para cada uno de los límites que posee el sistema?				
21	¿Los usuarios del sistema están de acuerdo con los casos definidos? ¿Los consideran completos?				
22	¿Se definieron tanto las condiciones normales como las fallidas a probar?				
23	¿Los usuarios están de acuerdo con que los casos de prueba cubren todos los probables escenarios?				

G.6. Métricas

- Eficiencia en la detección de defectos: $\frac{\text{Cantidad de defectos detectados hasta validación}}{\text{Cantidad total de defectos hasta validación}}$
- Eficiencia en la remoción de defectos: $\frac{\text{Cantidad de defectos removidos hasta validación}}{\text{Cantidad total de defectos hasta validación}}$
- Tasa de efectividad de la verificación: $\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$
- Estabilidad de las criterios de aceptación: $\frac{\text{Cantidad de criterios iniciales}}{\text{Cantidad total de criterios}}$
- Estabilidad de la aceptación: $\frac{\text{Cantidad de cambios durante la aceptación}}{\text{Cantidad total de cambios}}$

G.7. Involucrados

G.7.1. Equipo de Ingeniería

- Gerente de proyecto.
- Representante del grupo de análisis de requerimientos.
- Representante del grupo de análisis funcional.
- Representante del grupo de diseño.
- Representante del grupo de analistas desarrolladores.
- Grupo de testeo.
- Usuario final.

G.7.2. Equipo de Aseguramiento de la Calidad del Software

- Líder de Aseguramiento de la Calidad del Software
- Analistas de aseguramiento de la calidad del software Senior y/o Semi-senior.

H. Instalación

H.1. Objetivo

Realizar una verificación completa de la fase de instalación para nuevos sistemas y/o cambio de versiones. La verificación, incluye, para cada criterio de instalación identificado, una evaluación recomendada y las técnicas y herramientas sugeridas para llevarla a cabo.

H.2. Entradas

- Plan de instalación.
- Diagrama de flujo de la Instalación.
- Documentos y listados de instalación (sólo si es requerida una instalación especial).
- Resultados de la verificación de instalaciones especiales.
- Documentación de pasaje del sistema a producción.
- Instrucciones para los nuevos operadores.
- Instrucciones y procedimientos para los nuevos usuarios.
- Resultado del proceso de instalación.

H.3. Proceso

En la figura 12 se detalla gráficamente, el proceso a desarrollar durante este módulo:

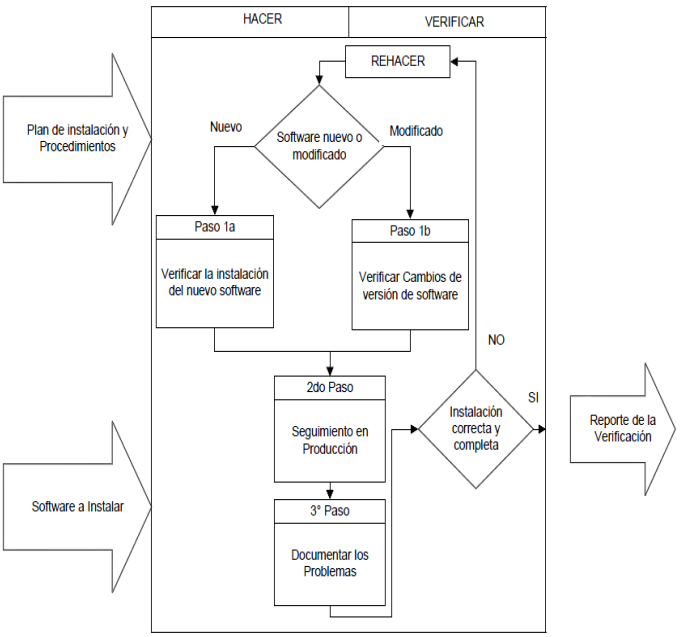

Fig. 12. Proceso del módulo de instalación

H.3.1. Verificación de la Instalación de Nuevo Software

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

Es importante que los resultados de la verificación estén disponibles antes de la instalación completa del sistema. El objetivo de la verificación es determinar si la instalación fue exitosa, por lo tanto los resultados deben estar disponibles lo antes posible. Esto significa que los resultados a obtener de la verificación deben estar explicitados antes de que la verificación comience. Los 15 factores a tener en cuenta en la verificación de la instalación son:

- Integridad y exactitud de la instalación.
- Prohibición de modificación de datos durante la instalación.
- Integridad de archivos de producción.
- Registro de pistas de auditoría de instalación.
- Integridad del sistema/versión anterior asegurado (continuidad de procesamiento).
- Recuperación ante fallas de la instalación.
- Control de acceso durante la instalación.
- Instalación acorde a la metodología.
- Instalación en producción de los programas indicados en la fecha asignada.
- Puesta a disposición de instrucciones de instalación.
- Documentación completa (Mantenibilidad).
- Documentación completa (Portabilidad).
- Interfaces.
- Monitoreo de la performance.
- Procedimientos operativos.

H.3.2. Verificación de la Instalación de Cambios de Software

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

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

Son tres las sub-tareas a verificar en la instalación de cambios de software.

a) Verificar si es adecuado el plan de recuperación ante fallas

Existen varios aspectos de los cambios que impactan en el proceso de recuperación ante fallas:

- Agregado de una nueva función.
- Cambio en un JCL (Job Control Language).
- Uso adicional de programas utilitarios.
- Cambios en períodos de resguardo.
- Cambios en los programas.
- Introducción de un formulario nuevo o revisado.

Los analistas de calidad deben evaluar el impacto en el plan de recuperación, cambio por cambio. Si determinan que el cambio afecta a la recuperación, se debe actualizar el plan.

b) Verificar que el cambio correcto, fue puesto en producción

El pasaje de la modificación, desde del ambiente de prueba a producción lo debe hacer el dueño del software, previa aprobación del usuario.

El ambiente de producción debería ser capaz de controlar los programas de acuerdo con la fecha de puesta en producción. Cada versión del programa en producción debería ser identificada (etiquetada) de acuerdo a si entra o sale de producción. Debe estar disponible para cada programa un historial de los cambios, y así poder proveer pistas de auditoría.

Para verificar que se introdujo el cambio correcto en producción se verificará lo siguiente:

- Que los cambios en los programas hayan sido documentados: El objetivo es contar con un historial, para proveer pistas de auditoría que indiquen cuándo se ha realizado un cambio y por otro impedir cambios no autorizados.
- Que se cuente con un procedimiento para pasajes de versiones del ambiente de prueba al ambiente de producción.
- El dueño del software decide cuándo una versión va a ser pasada a producción. Esta aprobación da la orden a operaciones para dar aviso que el cambio va a ser instalado.
- Que el procedimiento citado en el punto anterior se cumpla

c) Verificar que las versiones innecesarias hayan sido eliminadas

Los programas con versiones innecesarias (viejas) deben ser borrados. Esto solo puede hacerse con el debido nivel de autorización. Se debe elaborar un formulario que autorice la eliminación de las versiones antiguas. Este formulario lo debe completar el Gerente del proyecto de mantenimiento de software y enviarlo al departamento de operaciones.

El departamento de Operaciones debe tener un proceso para eliminar versiones innecesarias, después de recibir la debida autorización.

H.3.3. Seguimiento en Producción

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

El personal asignado al mantenimiento de software y el usuario deberán proveer pistas, acerca de dónde ellos creen que podrían ocurrir problemas. El tipo de indicios incluye:

- Investigación de transacciones.
- Clientes.
- Reportes.
- Archivos.
- Performance.

Estos indicios deben ser documentados mediante el uso de un formulario.

H.3.4. Documentar los Problemas

Los problemas detectados durante el monitoreo de los cambios, deben ser documentados. Además se debe evaluar el riesgo asociado a ese problema.

El reporte de un problema causado por un cambio en el sistema, permite asociar tal problema a lo específico del cambio. Este reporte debe ser enviado al analista de mantenimiento de software para su análisis y posterior corrección.

H.4. Salidas

- Reporte de recuperación ante fallas.
- Reporte de historia de cambios al software.
- Reporte de puestas en producción.
- Reporte de autorización de eliminación de versiones innecesarias.
- Reporte de notificación de monitoreo de cambios de software.
- Reporte de problemas causados por cambios de Software.

H.5. Lista de Verificación (Checklist)

En la tabla XI se presenta la lista de verificación.

H.6. Métricas

- Eficiencia en la detección de defectos:

$$\frac{\text{Cantidad de defectos detectados hasta instalación}}{\text{Cantidad total de defectos hasta instalación}}$$

- Eficiencia en la remoción de defectos:

$$\frac{\text{Cantidad de defectos removidos hasta instalación}}{\text{Cantidad total de defectos hasta instalación}}$$

- Tasa de efectividad de la verificación:

$$\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$$

H.7. Involucrados

H.7.1. Equipo de Ingeniería

- Gerente de proyecto.
- Grupo de mantenimiento de software.
- Grupo de instalación de software.
- Grupo de operaciones.

H.7.2. Equipo de Aseguramiento de la Calidad del Software

- Líder de Aseguramiento de la Calidad del Software

TABLA XI. LISTA DE VERIFICACIÓN DEL MÓDULO DE INSTALACIÓN

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
1	¿Cada cambio es revisado para cada impacto sobre el plan de reinicio/recuperación?				
2	¿Si el cambio impacta a la recuperación, es calculado nuevamente el tiempo de no servicio?				
3	¿Si el cambio impacta a la recuperación, es estimado nuevamente el riesgo del tiempo de no servicio?				
4	¿Los cambios necesarios para el proceso de recuperación son documentados?				
5	¿La notificación de los cambios de la versión de producción fueron documentados?				
6	¿Los cambios de los sistemas de aplicación de control de cambio de aplicación son controlados por una numeración?				
7	¿Existen procedimientos para eliminar versiones antiguas de las bibliotecas de objetos?				
8	¿Existen solicitudes de eliminación para que producción sea autorizado para eliminar programas?				
9	¿Están establecidos procedimientos para asegurar que la versión de los programas sea pasada al ambiente de producción en la fecha correcta?				
10	Si esto afecta a procedimientos de operación, ¿Están notificados los operadores del día en que la nueva versión se pase a producción?				
11	¿Están establecidos los procedimientos para monitorear los cambios de los sistemas de aplicación?				
12	¿Las personas que monitorean reciben notificación de que el sistema de aplicación fue cambiado?				
13	¿Las personas que monitorean los cambios reciben indicios de las áreas consideradas impactadas por el probable problema?				
14	¿Las personas que monitorean el cambio del sistema de aplicación reciben una guía de que acciones tomar si el problema ocurre?				
15	¿Los problemas detectados, inmediatamente después de que el cambio del sistema ocurrió, son documentados en un formulario especial para poder vincularlos a un cambio en particular?				
16	¿Se solicita, a las personas que documentan problemas, que documenten el impacto a nivel organizacional que puede implicar el problema?				
17	¿La información acerca de la instalación de cambios de software es recopilada y documentada?				
18	¿El servicio de la dirección realiza revisiones y utiliza la retroalimentación de los datos?				
19	¿El servicio de la dirección periódicamente revisa la efectividad de la instalación de cambios de software?				

- Analistas de aseguramiento de la calidad del software Senior y/o Semi-senior.

I. Metodología, Estándares y Procedimientos

I.1. Objetivo

La verificación de productos, deberá ajustarse a la organización. Esto incluye ajustar el vocabulario y los términos utilizados en la verificación para que sean los mismos que los utilizados en la organización para describir los componentes del sistema, documentos y productos.

I.2 Entradas

- Metodología de desarrollo de la organización.
- Estándares de la organización.
- Procedimientos de la organización.
- Productos que ingresan a cada una de las fases definidas en el presente modelo.

I.3 Proceso

En la figura 14 se detalla gráficamente, el proceso a desarrollar durante este módulo.

I.3.1. Verificar el Cumplimiento de los Estándares de Documentación

La formalidad, extensión y nivel de detalle de la documentación a preparar dependerán de las prácticas de la organización y del tamaño, complejidad y riesgo del proyecto.

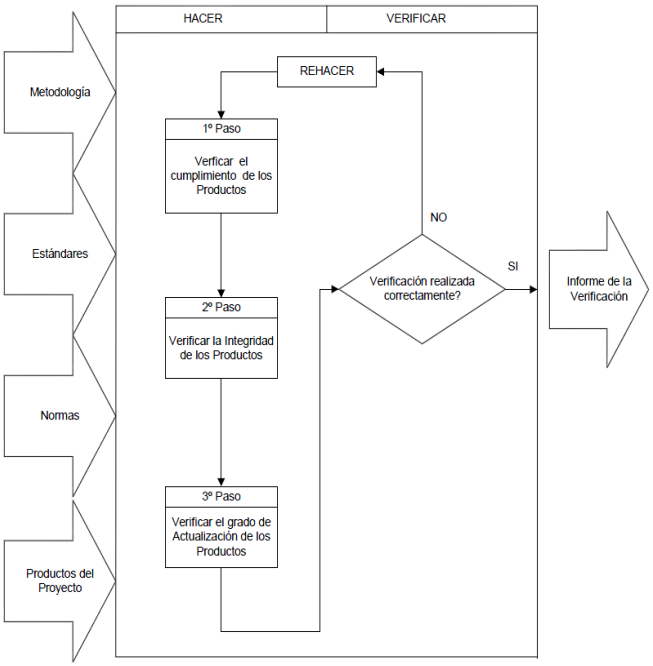

Fig. 13. Proceso del módulo de metodología, estándares y procedimientos

Lo que es adecuado para un proyecto puede ser inadecuado para otro. Para verificar la documentación, se procederá a comprobar si está completa, es adecuada y cumple con las normas y estándares corporativos.

Esta tarea intenta evaluar el apego a los procedimientos y estándares definidos por la metodología de la organización, evaluando los criterios así establecidos y determinando si el nivel y el alcance de la documentación cumplen con lo requerido.

En el caso que la metodología de la organización, establezca pautadamente la documentación necesaria para cada tipo de proyecto según su criticidad, se procederá a verificar si la documentación cumple con cada una de esas pautas.

I.3.2. Verificar la Integridad de la Documentación

Para evaluar la integridad de la documentación, se usarán los criterios definidos al respecto, en la metodología de la organización. El equipo de aseguramiento de la calidad de software deberá determinar si cada documento cumple con esos criterios. Si la documentación no alcanza a cubrir un criterio, deberá identificarse la deficiencia y documentarse.

Otros criterios adicionales, que pueden ser utilizados para evaluar la integridad de cada documento, son discutidos en el capítulo de Técnicas y Herramientas, nombrándose a continuación cada uno de ellos:

- Contenido de la documentación.
- Usuarios del documento.
- Redundancia.
- Flexibilidad.
- Tamaño del documento.
- Combinación de diferentes tipos de documentos.
- Formato.
- Secuencia de contenido.
- Títulos de secciones/párrafos.
- Expansión de los párrafos.
- Diagramas de flujo y tablas de decisión.
- Formularios.

Es aconsejable llevar a cabo una o más verificaciones de adecuación de la documentación a los estándares de la organización. Tales verificaciones requieren que una persona capaz, no asociada al proyecto, haga una simulación de cambio al sistema basado en la documentación actual y el requerimiento de cambio (no deben cambiarse ni la documentación real ni los programas). Después que la persona haga el cambio, alguien familiarizado con el proyecto debe evaluar si se ha hecho correctamente. Esta verificación revelará si:

- La documentación es entendible para las personas ajenas al proyecto
- Una persona ajena puede utilizar la documentación para hacer un cambio correctamente, y lo puede hacer de manera eficiente y efectiva.

I.3.3. Verificar el grado de actualización de la Documentación

La documentación que no esté actualizada no será útil. Si la metodología de la organización ya establece algún método para verificar el grado de actualización de la documentación, dicho método será usado por el grupo de verificación durante la misma. Caso contrario, se propone que el equipo de verificación de la documentación utilice cualesquiera de las siguientes cuatro técnicas de verificación (que se detallarán se detallan en el capítulo de Técnicas y Herramientas) para validar la actualización de la documentación:

- Utilizar la documentación existente para llevar a cabo un cambio en la aplicación
- Comparar el código fuente de los programas con la documentación
- Verificar que la documentación esté vigente
- Verificar la actualización de los documentos con el usuario final

Las anteriores técnicas podrán aplicarse sobre los documentos completos o sobre partes de los mismos o sobre una muestra de la documentación del proyecto.

I.4. Salidas

- Reporte de documentación verificada

I.5 Lista de Verificación (Checklist)

En la tabla XII se presenta la lista de verificación.

I.6. Métricas

- Documentación vigente:

$$\frac{\text{Cantidad de documentación vigente no actualizada}}{\text{Cantidad total de documentación vigente}}$$

- Revisión de documentos:

$$\frac{\text{Cantidad de documentos revisados}}{\text{Cantidad de documentos a revisar}}$$

I.7. Involucrados

I.7.1. Equipo de Ingeniería

- Gerente de proyecto
- Responsables de elaboración de documentos
- Grupo de Metodología, Estándares y Procedimientos

TABLA XII. LISTA DE VERIFICACIÓN DEL MÓDULO DE METODOLOGÍA, ESTÁNDARES Y PROCEDIMIENTOS

#	Ítem	Respuesta			Comentarios
		SI	NO	N/A	
1	¿Existen estándares para la documentación del sistema?				
2	¿Los miembros del equipo de prueba están informados sobre las intenciones y contenidos de esos estándares?				
3	¿Los estándares son adaptables a sistemas de varios tamaños, de manera que el tamaño del proyecto no sea directamente proporcional con al tamaño del documento?				
4	¿Se les entregó al equipo de aseguramiento de la calidad una copia completa de la documentación del sistema correspondiente a esta verificación?				
5	¿El equipo de aseguramiento de la Calidad ha medido las necesidades de la documentación de acuerdo a los criterios establecidos por la metodología de la organización?				
6	¿El equipo de aseguramiento de la calidad ha determinado qué documentos deben revisarse?				
7	¿Las personas del proyecto están de acuerdo con las sugerencias del equipo de aseguramiento de la calidad en cuanto a las revisiones de la documentación?				
8	¿El equipo de aseguramiento de la calidad determinó la completitud de los documentos usando los criterios detallados en la metodología de la organización?				
9	¿ El equipo de aseguramiento de la calidad ha utilizado el proceso de Inspección para determinar la completitud de la documentación del sistema?				
10	¿ El equipo de Aseguramiento de la Calidad de Software ha determinado el grado de actualización de la documentación del proyecto?				
11	¿ El equipo de aseguramiento de la calidad ha desarrollado un documento detallando las deficiencias de la documentación respecto de los estándares de la organización?				
12	¿Se ha asegurado el equipo de aseguramiento de la calidad de que las deficiencias descritas están documentadas?				

I.7.2. Equipo de Aseguramiento de la Calidad del Software

- Líder de Aseguramiento de la Calidad del Software
- Analistas de aseguramiento de la calidad del software Senior y/o Semi-senior.

V. TÉCNICAS Y HERRAMIENTAS

En el presente capítulo se presentan y describen brevemente las principales Técnicas y Herramientas identificadas en los módulos del modelo general de aseguramiento de la calidad.

También se presenta, sólo a modo de ejemplo, un conjunto de productos entregables que se obtienen al aplicar éstas técnicas y herramientas. Lo que se pretende mostrar con estos productos, son los datos que se deberían registrar, más allá del formato, el soporte electrónico o detalles de implementación.

A. Estimación del Proyecto

A.1. Verificar la Validez de la Estimación de los Costos de Software

Una estimación inapropiada de costos puede dañar más a la calidad del Proyecto de Software que a cualquier otro factor, por eso, la verificación puede aumentar la validez de la estimación. La estimación del software es un proceso de tres partes como se describe a continuación:

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

A.1.1. Validar el Modelo de Estimación

La organización necesita usar un modelo para realizar las estimaciones. El objetivo de este punto es validar la sensatez del modelo de estimación. Esta verificación deberá desafiar el modelo usando las siguientes 14 características deseables para la estimación de costos.

Estas características buscan determinar los puntos débiles del modelo:

- El alcance del modelo debe estar bien definido.
- El modelo debería ser ampliamente aplicado.
- El modelo debe ser fácil de usar.
- El modelo debe ser capaz de usar información actual, cuando esté disponible.
- El modelo debe permitir el uso de datos históricos y ajustarlos para una organización en particular y un tipo de software.
- El modelo debe ser verificado contra un número histórico razonable de proyectos.
- El modelo debe requerir sólo entradas basadas en las propiedades del proyecto, las cuales están bien definidas y pueden ser establecidas con un grado de certeza razonable al momento en que la estimación es llevada a cabo.
- El modelo debe proveer entradas basadas en criterios objetivos.
- El modelo no debe ser hipersensible para criterios de entrada subjetivos.
- El modelo debe ser sensible a todos los parámetros del proyecto que fueron establecidos teniendo un marcado efecto en el costo y no debe requerir entrada de parámetros no correlacionados con costos.
- El modelo debe incluir estimaciones de cómo y cuándo van a ser necesarios los recursos.
- El modelo debe producir un rango de valores por la cantidad que ha sido estimada.
- El modelo debe incluir la posibilidad de realizar análisis de sensibilidad, para que el responsable de la estimación pueda ver la variación de los parámetros seleccionados.
- El modelo debe incluir alguna estimación del riesgo para completar las estimaciones de tiempo o costo.

A.1.2. Validar que el Modelo Incluya Todos los Factores Necesarios

Los factores que influencian al costo del proyecto de software deben estar divididos en los que contribuyen en el desarrollo y mantenimiento de la organización y en aquellos inherentes al proyecto de software.

Es importante que los modelos sean usados correctamente y que todos los factores que influencian los costos de software sean imputados correctamente. Los modelos pueden producir resultados incorrectos basados en esa influencia de factores. En primer lugar el factor puede ser excluido del modelo como resultado de una incorrecta estimación. En segundo lugar se puede introducir un factor incorrecto o incompleto al modelo, causando estimaciones incorrectas de costos de software.

Si un factor no fue incluido, el analista de calidad debe determinar dónde afecta el factor significativamente en el costo actual de construir el software. A continuación se describen los factores influyentes en el costo del software:

- Tamaño del software.
- Porcentaje de diseño y/o código nuevo.
- Complejidad del sistema de software.
- Dificultad de diseño y codificación.
- Calidad.
- Lenguajes a usar.
- Clasificación de nivel de seguridad del proyecto.
- Tipo de máquina a utilizar (tecnología).
- Utilización de hardware.
- Volatilidad del requerimiento
- A continuación se describen los factores influyentes, que dependen de la organización:
- Cronograma del proyecto.
- Personal.
- Ambiente de desarrollo.
- Recursos no atribuibles directamente a aspectos técnicos del proyecto.
- Recursos de computación.
- Honorarios.
- Inflación.

A.1.3. Verificar la Exactitud del Modelo de Estimación de Costos

Se proponen las siguientes cuatro verificaciones para validar las estimaciones producidas por el modelo de estimación de costos de software:

1) Recalcular lo estimado: El analista de calidad puede validar el proceso de estimación ejecutando el modelo de estimación. El propósito de esto es:

- Validar que los datos de entrada fueron ingresados correctamente.
- Validar que los datos de entrada fueron razonables.
- Validar que la ecuación matemática fue calculada correctamente.

2) Comparar estimaciones realizadas con proyectos típicos: El analista de calidad puede determinar cuánto tiempo lleva desarrollar proyectos del mismo tamaño y complejidad.

El cálculo realizado por el sistema de estimación, es luego comparado con costos actuales de proyectos típicos. Si existiera una diferencia el analista de calidad puede evaluar más en detalle, la validación de la estimación.

3) Razonabilidad de la estimación: Esta verificación es similar a la del punto anterior, se utiliza la experiencia pasada. El analista de calidad documenta los factores que influyen sobre la estimación de costos, documenta el cálculo producido por el sistema de estimación y luego valida la razonabilidad de esa estimación teniendo en cuenta experiencias anteriores. Es recomendable un mínimo de tres experiencias consultadas a los Gerentes de proyecto. En caso en que uno o más no sienta que es razonable la estimación, la misma deberá ser rechazada.

4) Redundancia en estimaciones de costo de software: El Analista de calidad debe recalcular la estimación usando otro modelo de estimación de costo. Si la diferencia es pequeña, la estimación será confiable. Caso contrario se tendría que realizar una investigación adicional.

A continuación, se detallan las fuentes de los modelos de estimación de software:

- Desarrollos internos (propios) de la organización.
- Modelos de estimación incluidos en metodologías de desarrollo de sistemas.
- Paquetes de software para desarrollo de estimaciones de software.
- Uso de puntos de función para la estimación de costos de software

B. Estado del Proyecto: Sistema de Acumulación de Puntos

La creciente complejidad de los sistemas, combinado con los requerimientos para diseños estructurados y modulares, ha aumentado la cantidad de elementos de software desarrollados y entregados. El aumento de elementos, más los tradicionales hitos usados para medir el progreso han hecho subjetivo y a menudo poco confiable el seguimiento del desarrollo del software y la predicción del consumo progresivo del tiempo.

Se ha utilizado exitosamente un sistema para seguir el progreso del desarrollo de software a través de un esquema de puntos ganados. Los puntos están asignados en cada paso en el ciclo de desarrollo del software de la organización. Los pasos son hitos en los cuales un producto generado es aceptado. Al aceptar los productos se ganan los puntos que poseen asociados. La proporción de puntos ganados sobre el total de puntos posibles se recopila y así se determina el progreso logrado. Luego, se generan informes, tabulando la información en una variedad de reportes gerenciales.

El sistema así implementado es flexible y altamente automatizado. Los puntos acumulados son rápidamente verificados, objetivos, y basados en el estado actual del desarrollo del proyecto. Cálculos simples o comparaciones de los puntos acumulados proveen una medida precisa del progreso, del desvío en el cronograma y la predicción de progresos futuros.

B.1. Cómo Utilizar el Sistema de Puntos

El sistema de acumulación de puntos, propuesto por W. Perry [5], es una extensión del sistema de "hitos". En su forma más simple se asume que cada ítem o componente del software pasa por procesos de desarrollo similares y hay una cantidad de hitos claramente identificables dentro del proceso.

A modo de ilustración, se desarrollarán 5 componentes y 4 hitos definirán el proceso de desarrollo. Los hitos pueden representar diseños revisados y aceptados, código revisado y completo, resultados de prueba verificados, y componentes liberados. En un caso simple, cada hito para cada ítem de software vale 1 punto. En este caso, pueden ganarse 20 puntos. Como parte de cada revisión de diseño, inspección de código, prueba de verificación, o entrega, se cumple el hito y se ganan los puntos correspondientes. Al poner en un archivo una lista con los componentes e hitos logrados (puntos ganados) y crear simples generadores de reportes, puede adquirirse una medida objetiva, precisa y oportuna de los resultados. En la tabla XIII se muestra cómo es un reporte simple de estado.

Este esquema simplificado funciona bien para un conjunto homogéneo de componentes donde todos tienen similar complejidad y cada hito representa una cantidad de trabajo similar. A través de la introducción de "pesos" en los factores, pueden manejarse fácilmente, los componentes de complejidad diversa o los hitos que representan esfuerzos disímiles para completarlos.

TABLA XIII. REPORTE SIMPLE DE ESTADO

REPORTE DEL ESTADO DEL SISTEMA					
ÍTEM	DISEÑO	CODIFICACIÓN	VERIFICACIÓN	ENTREGA	PUNTOS GANADOS
Componente A	1	1			2
Componente B	1				1
Componente C	1				1
Componente D	1	1	1		3
Componente E	1	1			2
TOTALES	5	3	1	0	9
PORCENTAJE 9/20 = 45= %					

El motor del sistema es un archivo de datos y algunos reportes simples. El archivo de datos es simplemente una colección de registros, uno por cada ítem que debe seguirse, que contiene campos para indicar si se ha alcanzado algún hito. Usualmente es ventajoso incluir campos para la descripción de los ítems, analista responsable, identificación del trabajo, entre otros.

El mantenimiento o actualización del archivo puede ser tan efectivo como modificar registros con un editor de línea o tan complejo como crear un programa interactivo para ese propósito. Deben usarse algunos medios de acceso limitado para restringir modificaciones no autorizadas al archivo. Una vez actualizado el mismo, para incluir una entrada de datos al componente en desarrollo se actualizan los campos de estado de los hitos a medida que se van alcanzando tales hitos. En algunos casos, esto puede ser un proceso manual; una vez que el evento ocurrió y se alcanzó el hito, una persona (autorizada) actualiza el estado en el archivo. En otras circunstancias, en sistemas más sofisticados, un programa puede determinar que ha ocurrido un hito (compilación sin errores o prueba exitosa) y automáticamente actualiza el estado del hito.

Una vez armado el archivo, se escriben los programas generadores de reportes para imprimir el estado. Para proyectos menores, puede ser suficiente un programa que simplemente imprime cada registro, suma los puntos ganados y definidos, y calcula el porcentaje de puntos ganados. Los proyectos más grandes pueden necesitar varios reportes para subsistemas diferentes o reportes resumen que enfaticen los cambios ocurridos.

B.2. Utilización del Sistema de Puntos como Método de Prueba

El método de puntos para seguir el progreso del software puede ser utilizado por el Analista de calidad en alguna de las tres formas siguientes:

B.2.1. Validar el Progreso Informado.

El uso del sistema de puntos por el equipo de aseguramiento de la calidad, elabora una evaluación del progreso que puede compararse con los reportes de progreso del Gerente del proyecto. Si el seguimiento del progreso es aproximadamente el mismo utilizando los dos resultados, el examinador puede validar el sentido de los reportes del proyecto producidos por el equipo del proyecto. Esto es un anexo de lo que la prueba normalmente hace, pero puede ser muy valiosa desde la perspectiva de la alta gerencia. Nótese que si hay una diferencia significativa en la estimación del progreso del proyecto, la diferencia puede estar en el sistema de puntos o en el sistema que usa el Gerente del proyecto. El objetivo de hacer ambas estimaciones es proveer al Gerente del proyecto y a la alta gerencia de mayor seguridad respecto de la validez de los resultados del reporte de progreso.

B.2.2. Planeamiento de la Prueba

El método de puntos para el seguimiento del progreso indica cuándo ocurrirá la prueba. Al tener un sistema que permita al equipo de proyecto, seguir el progreso, puede asistirlo para planificar el uso de los recursos de prueba. El sistema de puntos no requiere muchos recursos; no obstante está destinado a asistir al equipo de proyecto, indicándole cuando los programas/subsistemas estarán disponibles para verificar.

B.2.3. Reportar el Estado de la Prueba

El sistema de puntos también indica cuándo está hecha la prueba y cuando se liberan los módulos a producción. La información contenida en el sistema de puntos es la misma que necesitará el equipo de proyecto para reportar los resultados de la prueba.

C. Matriz de Riesgos

La matriz de riesgos es una herramienta diseñada para identificar y evaluar riesgos y determinar qué es lo que el sistema debe hacer con cada uno de ellos.

A continuación se describe como usar la matriz de riesgos. En forma ideal la matriz de riesgos comienza en la fase de requerimientos y se desarrolla y termina en la fase de diseño. La implementación de esta herramienta es un proceso de cinco pasos. Los pasos deben ser ejecutados en la siguiente secuencia:

C.1 Identificación del Equipo de Evaluación de Riesgos

La clave para el éxito de la matriz de riesgos es establecer un equipo evaluador de riesgos, cuya responsabilidad será completar la matriz. El objetivo de completar la matriz, es llevar a cabo un control adecuado de requerimientos y diseño para reducir los riesgos a un nivel mínimo aceptable.

El grupo de riesgo, puede ser parte del equipo que realiza la toma de requerimientos o parte del grupo de test, o puede ser un equipo seleccionado específicamente con el propósito de confeccionar la matriz de riesgo. El grupo debería contar con entre 3 y 6 miembros, y al menos poseer las siguientes habilidades:

- Conocimiento sobre el uso de la aplicación.
- Entender el concepto de riesgo.
- Habilidad para identificar controles.
- Estar familiarizado con ambos, riesgos de la aplicación y de los servicios de información.
- Entender el concepto de servicio de información y sistema de diseño.
- Los candidatos en el grupo de riesgo deberán al menos incluir a alguien del área usuaria, y alguno o todos de las siguientes áreas:
- Auditor interno.
- Consultor de riesgo.
- Representante del área de procesamiento de datos.
- Representante del área de seguridad.
- Representante del área de operaciones.

C.2. Identificación de Riesgos

El primer objetivo del grupo evaluador de riesgos, es identificar los riesgos enfocándose en la aplicación, pero no en

los riesgos ambientales. El equipo evaluador de riesgo, puede utilizar uno de los dos métodos siguientes para la identificación del riesgo:

C.2.1. Análisis de Escenarios de Riesgos

En este método el equipo de riesgo delibera sobre los riegos potenciales de la aplicación usando sus experiencias, juicio y conocimiento del área de aplicación. Es importante tener sinergia, así los miembros del grupo pueden cuestionarse uno a otro para desarrollar una lista completa de riesgos que son compatibles a la aplicación.

C.2.2. Lista de Verificación de Riesgos

Se provee al equipo de riesgo de una lista con los riesgos más comunes que ocurren en aplicaciones automáticas. El equipo selecciona de la lista aquellos riesgos aplicables a la aplicación. En este método, el equipo necesita pocas habilidades porque la lista de riesgos provee el estímulo para el proceso, y el objetivo del equipo es determinar cuales de los riesgos de la lista son aplicables a la aplicación. He aquí una lista de riesgos para su identificación:

- Acceso no controlado al sistema.
- Prácticas de seguridad no eficaces para la aplicación.
- Errores de procedimiento en los servicios de información:
- Procedimientos y controles.
- Manipulación de medios de almacenamiento.
- Errores del programa.
- Defectos del sistema operativo.
- Fallas en el Sistema de Comunicación:
- Fallas accidentales.
- Actas accidentales.

C.3. Establecer Objetivos de Control

Durante la fase de requerimientos, deben establecerse los objetivos de control para cada riesgo. Estos objetivos definen el nivel de aceptación del perjuicio de cada riesgo identificado. Otra forma de manifestar el nivel de daño aceptable, es el objetivo mensurable de control.

La adecuación del control no puede chequearse hasta que no esté definido el nivel de perjuicio de cada riesgo. Por lo tanto, mientras que la definición de los objetivos de control es responsabilidad del usuario y del proyecto, puede llevar a la formación de un grupo de riesgos para definirlos. Una vez definidos los objetivos de control, se pueden chequear los requerimientos para determinar si son factibles.

En la tabla XIV se muestra un ejemplo de matriz de riesgo al final de la fase de requerimientos para un típico sistema de Facturación y Distribución. Esta matriz muestra cuatro riesgos para el sistema de Facturación y Distribución, y objetivos de control para cada uno de aquellos riesgos. Por ejemplo, uno de los riesgos es que el producto sea enviado pero no facturado. En esta instancia, el objetivo de control es asegurar que todos los envíos sean facturados. En otras palabras, el nivel de perjuicio aceptable de este riesgo es cero, y el equipo del proyecto debe instalar un sistema que asegure que para cada envío que se despacha se prepara una factura. Sin embargo, nótese que el siguiente riesgo es que el producto se facturará con un precio o cantidad errónea y que los controles tienen establecido un nivel de defecto mayor a cero, como los otros dos riesgos.

C.4. Identificar Controles en Cada Sistema

Durante la fase de diseño, el equipo de riesgo identificará los controles en cada fase del sistema de aplicación para cada riesgo identificado. Los segmentos de sistemas más comunes son:

- Origen: La creación del documento fuente más la autorización asociada con aquella transacción original.

TABLA XIV. EJEMPLO DE MATRIZ DE RIESGO AL FINAL DE LA FASE DE REQUERIMIENTOS

RIESGO	OBJETIVOS DE CONTROL
Despachado pero no facturado	Asegurar que todos los envíos estén facturados
Facturado con precio o cantidad errónea	Facturar al precio actual el 99% de los ítems básicos y error permitido menor al 10%
Facturado al cliente equivocado	Reducir facturas perdidas a menos del 1%
Despacho del producto o cantidad equivocada	Despachar los productos y cantidades correctas al 99% de los ítems de base

- Ingreso de Datos: Transferencia de información a un medio legible por la máquina.
- Comunicación: El movimiento de datos desde un punto del sistema a otro. El movimiento puede ser manual o electrónico.
- Procesamiento: Aplicación del sistema lógico a los datos.
- Almacenamiento: La retención de datos, por largos o cortos períodos de tiempo.
- Salida: La traducción de la información de computadora a los medios entendibles y utilizables.
- Uso: Satisfacción de la necesidad del negocio a través de los resultados del sistema de procesamiento.

El equipo de evaluación de riesgos, determinará qué controles son aplicables a qué riesgos y los registrará en el segmento correcto del sistema. Al término del desarrollo de la matriz de riesgo, el equipo de riesgo hará una estimación para verificar si los controles son los adecuados para reducir el riesgo hasta el nivel de aceptación identificado en el objetivo de control. Esto verificará la efectividad de los controles al finalizar el proceso de diseño. En la tabla XV se muestra un ejemplo de matriz de riesgo para sistemas de facturación y distribución al final de la fase de diseño.

Los mismos cuatro riesgos identificados durante la fase de Requerimientos (tabla XIV) también se muestran en esta matriz, por lo que los controles están asociados a cada riesgo.

En este ejemplo, el riesgo de despachar sin facturar muestra que los controles 1, 2 y 3 ayudarán a disminuir ese riesgo (para una matriz real estos controles deben describirse). La matriz muestra en qué segmento del sistema de aplicación residen aquellos controles.

Después de identificar y registrar los controles, el equipo de riesgo debe determinar si aquellos tres controles y los segmentos a los cuales pertenecen son los adecuados para reducir el riesgo de despachar sin facturar.

C.5. Determinar si los Controles son adecuados

La verificación termina cuando el equipo de riesgo determina si los controles son los adecuados para reducir cada uno de los riesgos identificados al nivel aceptable.

TABLA XV. EJEMPLO DE MATRIZ DE RIESGO AL FINAL DE LA FASE DE DISEÑO

RIESGO DEL SEGMENTO DEL SISTEMA	ORIGEN	INGRESO DATOS	COMUNIC.	PROCES.	ALMAC.	SALIDA	USO
Despacho sin facturación	#1			#2			**#6**
Facturado con precio o calidad errónea		#6		#7 #8 #9	#10	#11	
Facturado al cliente equivocado				#12 #3	#14	#15	**#16**
Despacho de producto o cantidad errónea	#17	#18		#19 #20		#21	**#22**

D. Análisis de Factores (Módulo de Requerimientos)

Se llevará a cabo un proceso para estimar las incumbencias asociadas a la fase de requerimientos del desarrollo del sistema. Debe incluirse un programa de verificación para cada ítem. Hay 15 ítems a considerar, cubriendo cada fase del proceso de desarrollo. Para cada ítem hay un programa de verificación que tiene en cuenta ciertas consideraciones las cuales se encuentran detalladas más abajo. El programa de verificación enumera aquellos criterios, que aseguran al equipo de aseguramiento de la calidad, que la magnitud de esa preocupación es mínima. A estos criterios debe responder el equipo de aseguramiento de la calidad. También se debe realizar una verificación suficiente para evaluar si el equipo de proyecto ha manejado adecuadamente cada factor de verificación.

A continuación, se detallarán brevemente cada uno de los factores a tener en cuenta:

D.1. Requerimientos Compatibles con la Metodología (Factor: Metodología)

El procedimiento utilizado para definir y documentar requerimientos, debe estar presente durante la fase de requerimientos. Cuanto más formales sean estos procedimientos, se facilita el proceso de su verificación. El proceso de requerimientos es un proceso de, análisis, toma de decisiones y registro de requerimientos en una forma predefinida para poder utilizarse luego en el diseño.

D.2. Funcionalidad de las Especificaciones (Factor: Precisión)

La satisfacción del usuario solo puede asegurarse, cuando se han cumplido los objetivos del sistema. El cumplimiento de estos objetivos solo pueden medirse cuando éstos sean mensurables. Por ejemplo, los objetivos cualitativos, como la mejora de servicio, no son mensurables, mientras que sí lo es el pedido de procesamiento en cuatro horas de usuario.

D.3. Usabilidad de las Especificaciones (Factor: Facilidad de Uso)

La cantidad de esfuerzo requerido para usar el sistema y la habilidad necesaria, deben definirse durante la fase de requerimientos. La experiencia muestra que las aplicaciones difíciles de usar finalmente no se utilizan, en cambio los sistemas funcionales fáciles de usar son altamente utilizados. Entonces los desarrolladores, deberán crear especificaciones fáciles de usar, incrementando así la facilidad del uso del sistema en sí mismo.

D.4. Mantenimiento de las Especificaciones (Factor: Mantenibilidad)

Debe definirse el grado de mantenimiento esperado, así como también las áreas donde los cambios son muy probables. Las especificaciones determinarán los métodos de mantenimiento (Ej.: cambio de parámetros introducido por el usuario) y el intervalo de tiempo en el cual se necesitan implementar dichos cambios.

D.5. Necesidades de Portabilidad (Factor: Portabilidad)

La capacidad para operar el sistema en diferentes tipos de hardware, o migrarlo a otra versión, deberá enunciarse como parte del requerimiento. La necesidad de tener la aplicación desarrollada como portable, puede afectar significativamente la implementación de requerimientos.

D.6. Interfaces del Sistema (Factor: Acoplamiento)

Se debe definir en forma precisa, la información esperada como entrada desde otros sistemas computadorizados y las salidas a entregar a otros sistemas. Esta definición no solo incluye los tipos de información intercambiada, sino también, el momento en el cual debe estar disponible cada interfaz y el procesamiento que se espera como resultado de cada una de ellas. Otros factores a tener en cuenta al definir las interfaces son: privacidad, seguridad y resguardo de la información.

D.7. Criterios de Performance (Factor: Performance)

Debe quedar establecido: la eficacia, economía y eficiencia esperadas del sistema. Estos objetivos son una parte integral del proceso de diseño. Cuando no son alcanzados, la insatisfacción del usuario está garantizada. Como producto final de la fase de requerimientos, se realizará el análisis del costo-beneficio del factor performance, calculado para la aplicación.

D.8. Necesidades operativas (Factor: Facilidad de Operación)

Las consideraciones operativas deben definirse durante la fase de requerimientos. Esto es especialmente importante en sistemas de aplicación manejados por el usuario. Los procesos a seguir para operar el sistema, en otras palabras, los procedimientos necesarios para procesar transacciones, deben ser lo más simple posible. También deben considerarse procedimientos de operación por excepción y monitoreo centralizado.

D.9. Tolerancia (Factor: Fiabilidad)

Debe definirse la confiabilidad esperada de los controles del sistema. Por ejemplo, la fase de requerimientos determinará los requerimientos de control para la precisión de la facturación, el porcentaje de órdenes que necesitan procesarse dentro de las 24 horas, etc.

La tolerancia de facturación puede afirmar que se procesarán las facturas con tolerancia ± 1% de los precios de los productos actuales.

Si no se establecen estas tolerancias, no hay base para asignar y medir la confiabilidad del sistema por período de tiempo. Si no se define el nivel de defectos esperados, normalmente se espera cero defectos. Los controles para lograr cero defectos son antieconómicos.

Usualmente es más económico, que surja una cantidad mínima de defectos en el proceso pero que sean detectados y mensurables.

D.10. Reglas de Autorización Definidas (Factor: Autorización)

Deben especificarse los métodos de autorización (niveles de seguridad) para asegurar que las transacciones se procesen de

acuerdo con la intención que tiene la organización, respecto del sistema.

D.11. Requerimientos de Integridad de Archivos (Factor: Integridad de Archivos)

Se necesita especificar los métodos para asegurar la integridad de los archivos. Esto normalmente incluye el conjunto de controles que deben mantenerse dentro del archivo e independientemente de la aplicación. Los controles deben asegurar que los registros de detalle sean consistentes con los totales de control para cada archivo.

D.12. Recuperación ante Fallas (Factor: Auditoria)

La recuperación abarca los procedimientos de recomposición ante la identificación de un problema. Se debe definir para cada aplicación, las necesidades de recomposición de procesos e información. Si la recomposición es necesaria, se necesita enunciar el momento para su ejecución. Este momento, puede variar según la hora del día y el día de la semana.
Estos requerimientos de recomposición afectan el tipo y disponibilidad de los datos.

D.13. Impacto de Fallas (Factor: Continuidad de Procesamiento)

La necesidad para asegurar la continuidad de procesamiento depende del impacto de las fallas. Si la falla causa solo problemas mínimos, puede ser innecesario asegurarse el procesamiento continuo. Por el contrario, cuando la continuidad de la operación es esencial, es necesario duplicar los equipos y centros de cómputos, para que se pueda continuar procesando.

D.14. Nivel de Servicio Deseado (Factor: Nivel de Servicio)

El nivel de servicio implica tiempo de respuesta basado en los requerimientos. El nivel de servicio requerido variará sobre la base de los requerimientos. Cada nivel de servicio deseado necesita estar enunciado; por ejemplo, hay un nivel de servicio para corregir un error de programación, otro para instalar un cambio y otro para responder a una consulta, etc.

D.15. Permisos y Accesos (Factor: Seguridad)

Los requerimientos de seguridad deben desarrollarse mostrando la relación entre recursos de sistema y humanos. Los requerimientos deben enunciar todos los recursos del sistema disponibles sujetos a control, y luego indicar quién debe tener acceso a aquellos recursos y con qué propósitos. Al final del módulo, el equipo de aseguramiento de la calidad, puede emitir juicio acerca de la adecuación de cada criterio. El equipo debe emitir uno de los siguientes cuatro juicios acerca de cada criterio:

- Muy adecuado: El equipo de proyecto ha hecho más de lo normalmente esperado para el criterio.
- Evaluación adecuada: El equipo de proyecto ha hecho el trabajo suficiente para asegurar el control por encima del criterio.
- Evaluación inadecuada: El equipo de proyecto no ha hecho el trabajo suficiente, y deben trabajar más en este criterio.
- No aplicable (N/A): Debido al tipo de aplicación o diseño filosófico del sistema por parte de la organización, la implementación de este criterio no es aplicable al sistema que se está revisando.

E. Inspecciones

Las inspecciones y las recorridas "walkthrough" apuntan a asistir a los productores para mejorar su trabajo.
Las inspecciones intentan examinar técnicamente el trabajo y proveer a los productores, una evaluación independiente de aquellas áreas productivas en donde las mejoras son necesarias. Las recorridas son generalmente menos formales y están conducidos en un formato más educacional. Las inspecciones, por el contrario, generalmente tienen un formato formal, la asistencia es específica, y la información se refleja en los resultados.
Hay tantos tipos de inspecciones como tipos de productos. Es de mucha ayuda inspeccionar los requerimientos antes de comenzar el diseño de alto nivel, el diseño de alto nivel antes de comenzar el diseño detallado, el diseño detallado antes de comenzar la implementación, y la implementación antes de comenzar la verificación. Esto no significa que todos los diseños de alto nivel deben ser inspeccionados antes del comienzo de cualquier diseño detallado, pero el diseño específico a realizar debe basarse en el trabajo que ha sido inspeccionado. También es de ayuda inspeccionar los casos verificados y la documentación.
Se recomiendan las inspecciones para diseño, codificación, casos de prueba, y documentación. Caso contrario, es adecuado un proceso de recorrida menos formal.

E.1. Proceso

El proceso de inspección sigue ciertos principios básicos:

- La inspección es un proceso formal, estructurado a través de un sistema de listas de verificación ("checklists") y roles definidos para los participantes. Provee un instrumento ordenado para implementar un estándar de excelencia de ingeniería del software o de conocimiento en toda la organización.
- Los estándares y listas de verificación genéricas son desarrollados para cada tipo de inspección y, cuando sea apropiado, se ajustan a las necesidades de un proyecto específico. Estas listas cubren el planeamiento de la inspección, la preparación, la conducción, la salida y reportes de normas.
- Los inspectores están preparados de antemano y tienen identificados sus tareas y cuestiones antes de que comiencen la inspección.
- El foco de la inspección estará en identificar problemas, no en resolverlos. Este foco, junto a lo mencionado en el punto anterior, asegura que la inspección pueda manejarse con una mínima pérdida de tiempo.
- Una inspección es conducida por técnicos para técnicos. Los directivos no se involucrarán, aunque serán informados de los hallazgos y las fechas en las cuales se resolverán los problemas identificados.
- La información de la inspección deberá ser ingresada a una base de datos y se la utilizará para monitorear la efectividad de la inspección, seguimiento y conducción de la calidad del producto.

Mientras mucha gente pueda estar interesada en los resultados de la inspección, el propósito de la inspección es asistir a los productores para mejorar su trabajo. Esto puede hacerse mejor limitándose a cinco o seis los inspectores. El punto clave de esta limitación es la atención técnica.

El moderador no es el director del trabajo que se está inspeccionando, ni tampoco ninguno de los otros participantes. Los moderadores necesitan una base completa de los principios y métodos de inspección antes de que puedan hacer un trabajo competente. Esto les dará las herramientas básicas y los ayudará a tener mayor confianza en sí mismos, necesaria para liderar tal actividad.
Como las inspecciones bien realizadas requieren de una intensa concentración de todos los participantes, éstos pueden quedar exhaustos. Por esto, las sesiones de inspección generalmente no deben exceder las dos horas.
También es de gran ayuda asignar algunos inspectores en áreas productivas específicas durante el proyecto.

E.2. Participantes

Los participantes de la inspección, se detallan a continuación:

- El moderador (líder de inspección): Persona responsable para liderar la inspección pronta y eficientemente a una conclusión satisfactoria.
- Los productores: Persona/s responsable/s de hacer que se inspeccione el trabajo.
- Los examinadores (inspectores): Generalmente es la gente directamente involucrada y conocedora del trabajo que está siendo inspeccionado.
- El registrador (quien anota): Alguien que registra los resultados significativos de la inspección.

E.3. Salidas

Las salidas a obtener luego de realizarse una inspección, se detallan a continuación:

- Informe de Inspección (Figura 14)
- Resumen de la Inspección (Figura 15)

INFORME DE LA INSPECCIÓN							
PROYECTO				FECHA			
SISTEMA				UNIDAD			
MODERADOR				OFICINA			
TIPO DE REUNION	VISIÓN GENERAL		REINSPECCION	REQUERIMIENTOS		DISEÑO	
	CODIFICACIÓN		VERIFICACIÓN	VALIDACIÓN		OTRA	
# INPECCIONES				DURACIÓN			
# REVISIONES				TIEMPO PREPARACIÓN			
ESTADO	ACEPTADA		CONDICIONAL	REINSPECCIÓN		FECHA	
INSPECTORES							
PRODUCTORES							
REGISTRADOR							
CERTIFICACIÓN DEL MODERADOR						FECHA	
COMENTARIOS							

Fig. 14. Informe de inspección

F. Ranking de Factores de Éxito (Módulo de Diseño)

El ranking ("scoring") es una herramienta predictiva que utiliza experiencias en sistemas anteriores. Se analizan los sistemas existentes para determinar los atributos o características de los mismos y su correlación con el éxito o el fracaso de cada aplicación en particular. Una vez que los atributos correlacionados al éxito o al fracaso pueden ser identificados, también pueden ser usados para predecir el comportamiento del sistema que está en desarrollo.
Los lineamientos que se utilizan bajo el concepto de ranking, se describen a continuación:

RESUMEN DE LA INSPECCIÓN								
PROYECTO					FECHA			
SISTEMA					UNIDAD			
MODERADOR					OFICINA			
TIPO DE REUNION	VISIÓN GENERAL		REINSPECCION		REQUERIMIENTOS		DISEÑO	
	CODIFICACIÓN		VERIFICACIÓN		VALIDACIÓN		OTRA	
TIPO	DEFECTOS IMPORTANTES				DEFECTOS MENORES			
	FALTANTE	INCORRECTO	NO REQUERIDO	TOTAL	FALTANTE	INCORRECTO	NO REQUERIDO	TOTAL
FUNCION								
INTERFAZ								
DATOS								
LOGICA								
ENTRADA/SALIDA								
RENDIMIENTO								
MANTENIMIENTO								
ESTANDARES								
DOCUMENTACIÓN								
FACTORES HUMANOS								
SINTAXIS								
OTROS								
TOTALES								

Fig. 15. Resumen de la inspección

- Muestreo: Los atributos que se utilizarán corresponderán a una muestra de todos los atributos abarcados en la implementación de un sistema.
- Alta correlación positiva. Los atributos elegidos tendrán que mostrar una alta correlación positiva en el pasado con cualquier éxito o fracaso durante la automatización de una aplicación. Estos atributos no deberían ser intuitivos sino, que deberán ser atributos para los cuales pueda demostrarse que la ausencia o presencia de los mismos muestre una alta correlación con respecto al resultado final del proyecto.
- Facilidad de uso. Para ser efectivo, el proceso de ranking debe ser lo más simple posible.
- Desarrollar el ranking de riesgo. El ranking para cada atributo debe determinarse en un formato mensurable de modo que el ranking de riesgo total pueda ser desarrollado para cada aplicación. Esto indicaría el grado de riesgo, el área de riesgo, y también una comparación de riesgos entre las diferentes aplicaciones.

Al ranking de riesgo se llega a través de un desarrollo matemático, como se muestra a continuación.

TABLA XVI. DESARROLLO MATEMÁTICO PARA OBTENER UN RANKING DE RIESGO

RIESGO DE LA CARACTERÍSTICA	# DE CARACTERÍSTICAS EVALUADAS	FACTOR DE MULTIPLICACIÓN	FACTOR PARA EL RANKING
Alta	3	3	9
Media	9	2	18
Baja	12	1	12
Total			39

Para usar esta tabla, el número de atributos o características calificadas con alto, medio y bajo, deberá ser totalizado, y los totales colocados en la columna "# (cantidad) de Características Evaluadas".
Luego a cada número se lo multiplica por el factor de multiplicación y así obtener el ranking de riesgo. Por ejemplo, el número total de características definidas con alto riesgo debe multiplicarse por tres. Luego los tres números se suman para llegar a totalizar el riesgo total, el cual puede utilizarse para comparar entre los diferentes sistemas, o bien, respecto de un estándar definido por la organización. cuanto más alto es el puntaje, mayor será el riesgo, y a menor puntaje menor riesgo.

G. Análisis de Factores (Módulo de Diseño)

Los factores que deben analizarse durante el módulo de diseño se describen a continuación:

G.1. Controles de Integridad de Datos

La integridad de datos va desde la identificación de riesgos, hasta la decisión gerencial de aceptar esos riesgos, establecida en términos de la cantidad de pérdidas aceptables. Los controles de integridad de datos se diseñan a partir de la tolerancia a tales riesgos, los cuales se encuentran especificados.

G.2. Reglas de Autorización

Las autorizaciones en los sistemas pueden ser manuales y/o automáticas. Los procedimientos para autorización manual deben especificarse durante la fase de diseño. Los métodos de autorización automática deben especificarse más detalladamente que los manuales, por el hecho de que no se pueden dejar librados a criterios personales según la situación que se presente.

G.3. Controles de Integridad de Archivos

La integridad de los archivos estará asegurada por los métodos de identificación de archivos, controles automáticos y controles de integridad de archivos mantenidos independientemente. Las especificaciones para este proceso integrador de tres partes debe determinarse durante la fase de diseño.

G.4. Pistas de Auditoria

Estas pistas proveen la capacidad para rastrear transacciones desde su origen hasta los controles. Además, las pistas de auditoria se usan para consolidar el procesamiento de transacciones individuales y recuperar la integridad de operaciones luego de su pérdida.
Frecuentemente los entes gubernamentales especifican las pistas de auditoria que necesitan retener para cada tipo de información que manipulan. Estas necesidades de información, deben definirse durante la fase de diseño.

G.5. Plan de Contingencias

El plan de contingencias esquematiza las acciones a tomar ante la aparición de problemas.
Este plan deberá incluir los métodos manuales a seguir cuando las aplicaciones automatizadas no estuvieran en operación, así como también, las consideraciones de lugar físico. Las especificaciones del plan de contingencia deben llevarse a cabo durante la fase de diseño.

G.6. Método para Alcanzar el Nivel de Servicio Requerido

La fase de requerimientos define el nivel de servicio a obtener durante la operación de la aplicación. El método para alcanzar ese nivel de servicio es desarrollado durante la fase de diseño. Esto involucra el desempeño del sistema y su habilidad para satisfacer las necesidades de los usuarios a lo largo del tiempo.

G.7. Procedimientos de Acceso

La seguridad en los sistemas automatizados es llevada a cabo predefiniendo quién puede tener acceso a qué recursos y para hacer qué. Esto estará indicado por los perfiles de seguridad definidos. Durante la fase de diseño, deberán desarrollarse los procedimientos, las herramientas y las técnicas necesarias para crear e implementar los perfiles de seguridad necesarios.

G.8. Diseño Acorde con la Metodología

El proceso de diseño de sistemas debe ejecutarse y documentarse de acuerdo con la metodología establecida por la organización. Los procedimientos estándares de diseño aseguran la facilidad de comprensión por todos los miembros del equipo capacitados y entrenados en esa metodología y al mismo tiempo ayuda a asegurar que se cumpla en forma completa el proceso de diseño.

G.9. Diseño Acorde con los Requerimientos

El diseño del sistema es una transformación de los requerimientos de los usuarios en especificaciones más detalladas. Durante cualquier transformación, pueden ocurrir malos entendidos o malas interpretaciones. Entonces, deben tomarse las medidas necesarias para asegurar que el diseño cumpla sus objetivos y esté focalizado a cumplir con los requerimientos definidos.

G.10. Facilidad de Uso

Cuanto el producto final sea más fácil de usar, habrá mayor probabilidad que el usuario lo utilice y que las transacciones sean procesadas correctamente. Deben considerarse las habilidades necesarias para cada puesto de trabajo y la motivación en el mismo, de todas las personas usuarias del sistema.

G.11. Mantenibilidad del Diseño

El costo de mantener una aplicación normalmente excede el costo de desarrollo. Identificar aquellos aspectos del sistema que son los más factibles de cambiar y construir aquellas partes del sistema para facilidad del mantenimiento es un aspecto importante del proceso de diseño. La mantenibilidad del diseño puede cambiar significativamente dependiendo de la frecuencia esperada de los cambios introducidos.

G.12. Portabilidad del Diseño

Si los requerimientos indican que el sistema debe poder ser transferido desde un ambiente de hardware a otro, o una versión de software de base a otra, el diseño debe tener en cuenta e incorporar estos aspectos de portabilidad. Cuando el hardware y software futuros son inciertos, el diseño debe ser lo más general posible y no intentar sacar ventaja de los aspectos o facilidades que brindan el hardware y software existentes.

G.13. Interfaces de Diseño

Deben ser identificadas y especificadas y documentadas las interfaces con otras aplicaciones. Las especificaciones de las interfaces deben también considerar usos alternativos de la información de la aplicación.

G.14. Diseño Acorde con Criterios Establecidos

El estudio de costo/beneficio realizado durante la fase de requerimientos no provee una estimación de gran precisión. La estimación de la fase de requerimientos puede estar afectada en más o menos el 50%. Durante la fase de diseño, la estimación del desempeño puede ser definida mejor por lo que se puede hacer una mejor predicción y así lograr el cumplimiento del criterio de desempeño establecido. Una guía útil a ser utilizada, es que la precisión de la estimación del criterio de desempeño al final de la fase de diseño puede variar en ±10%.

G.15. Necesidades Operacionales

El sector de operaciones deberá identificar los requerimientos de procesamientos futuros para preparar el manejo de tales requerimientos cuando el sistema se implemente. Cuanto más grandes sean los requerimientos de procesamiento, mayor será la necesidad de involucrar al sector de operaciones en la consideración de las alternativas de diseño.

H. Revisión del Diseño

El objetivo es identificar aquellos atributos del diseño que se correlacionan con los problemas del sistema. Entonces durante la revisión de diseño, se investigarán dichos atributos para determinar si el equipo de proyecto los ha tratado apropiadamente.

El equipo de revisión de diseño comprende los siguientes miembros:

- Personal del proyecto: El personal del proyecto puede conducir su propia revisión del diseño. Normalmente la persona asignada con la responsabilidad de la revisión del proyecto no es la misma que realmente diseñó el sistema; sin embargo el que revisa puede tener responsabilidad parcial en el diseño. Esto requiere que las personas durante el proceso de revisión acepten roles y responsabilidades diferentes a los que han tenido en el proceso de diseño. Debido a probables vínculos con el diseño real del sistema, tener una lista de verificación de revisión de diseño como herramienta evaluadora, normalmente cumple una valiosa función para el/los que revisan.
- Equipo de revisión independiente: Los miembros de este equipo de revisión no son miembros del proyecto que está siendo revisado. Pueden ser de otros proyectos o del equipo de aseguramiento de la calidad. Esta manera de operar provee un mayor grado de independencia para conducir la revisión ya que no habrá conflicto de intereses entre los roles de diseñador y del que revisa. Por otro lado es frecuentemente difícil para los inspectores ser críticos unos con otros, especialmente en situaciones donde el que revisa pueda eventualmente trabajar para la persona que está siendo revisada.
- La siguiente es una guía en la conducción de una revisión de diseño:
- Seleccionar el equipo de revisión: Los miembros del equipo de revisión deben seleccionarse antes del proceso de revisión propiamente dicho.
- Entrenar los miembros del equipo de revisión: Las personas que conducirán la revisión deben estar entrenadas sobre cómo dirigir la revisión. Mínimamente significa revisar el checklist y explicar el objetivo e intención de cada asunto. También es aconsejable entrenar a las personas involucradas en relaciones interpersonales y así realizar la revisión en un ambiente armonioso.
- Notificar al equipo del proyecto: El equipo del proyecto debe estar notificado de la revisión con varios días de antelación cuando comienza la revisión y su responsabilidad durante la misma. Es importante organizar formalmente la revisión para que estén todos presentes.
- Asignar tiempos adecuados: La revisión debe conducirse formalmente y tan eficientemente como sea posible. Aún cuando la misma gente que diseñó la revisión, la dirija, las relaciones interpersonales y los efectos de la sinergia de la revisión pueden producir muchos efectos positivos si se asigna el tiempo suficiente para permitir una interacción apropiada.
- Documentar los datos de la revisión: Deben registrarse todos los hallazgos realizados en la revisión. Normalmente esto puede llevarse a cabo en el checklist, a menos que los comentarios sean muy extensos. En cualquier caso, los datos deben hacer referencia a preguntas específicas del checklist que cubrían.
- Revisar los datos con el equipo de proyecto: La precisión de los datos debe ser comprobada por todas las personas involucradas, y la revisión no debe continuar si esto no está hecho. Es mejor hacerlo al final de la revisión que durante el proceso mismo.
- Desarrollar recomendaciones de revisión: Basándose en los datos, el equipo de revisión hará sus recomendaciones para corregir cualquier situación de conflicto. Estas recomendaciones son parte importante del proceso de revisión.
- Revisar recomendaciones con el equipo de proyecto: El equipo de proyecto debe ser el primero en recibir las recomendaciones y tener la oportunidad de aceptar, modificar o rechazar las recomendaciones.
- Preparar un reporte: Se debe preparar un reporte para documentar los hallazgos, y las acciones tomadas o a tomar acerca de las recomendaciones. Este reporte puede o no enviarse a altos niveles gerenciales, dependiendo de las reglas establecidas en la revisión por la organización. Sin embargo es importante tener un registro formal del proceso de revisión, qué se encontró y las acciones tomadas respecto de las recomendaciones.

La cantidad de revisiones dependerá de la importancia del proyecto y del lapso de tiempo en la fase de diseño. Así podrán llevarse a cabo revisiones del diseño de alto nivel y del diseño detallado.

H.1. Revisión del Diseño de Alto Nivel

Cada defecto descubierto durante la revisión del diseño lógico o de alto nivel debe documentarse. Se confecciona un reporte de defectos para llevar registro de los mismos, incluyendo tipo y categoría de los defectos. La descripción de cada defecto se registra bajo una columna de Faltante, Erróneo o Adicional. Al final de la revisión del diseño lógico, los defectos se resumen y totalizan. En la figura 16 se muestra un formulario de registro de defectos en la fase de diseño de alto nivel:

REGISTRO DE DEFECTOS EN LA FASE DE DISEÑO DE ALTO NIVEL				
CATEGORÍA	DEFECTO			
	FALTANTE	ERRÓNEO	ADICIONAL	TOTAL
LA INFORMACIÓN NO HA SIDO DEFINIDA ADECUADAMENTE				
DEFINICIÓN DE ENTIDAD INCOMPLETA				
ENTIDAD INCORRECTA				
ATRIBUTOS DE ENTIDAD INCORRECTOS				
VIOLACIÓN DE NORMALIZACIÓN				
CLAVE PRIMARIA INCORRECTA				
CLAVE FORÁNEA INCORRECTA				
SUBTIPO DE ENDIDAD INCORRECTO				
PROCESO NO DEFINIDO ADECUADAMENTE				
OTROS				
TOTALES				

Fig. 16. Formulario de registro de defectos en la fase de diseño de alto nivel

H.2. Revisión del Diseño Detallado

Cada defecto descubierto durante la revisión del diseño físico o detallado debe documentarse. Se confecciona un reporte de defectos para llevar registro de los mismos, incluyendo tipo y categoría de los defectos. La descripción de cada defecto se registra bajo una columna de Faltante, Erróneo o Adicional. Al final de la revisión del diseño físico, los defectos se resumen y totalizan. En la figura 17 se muestra un formulario de registro de defectos en la fase de diseño detallado:

REGISTRO DE DEFECTOS EN LA FASE DE DISEÑO DETALLADO				
CATEGORÍA	DEFECTO			
	FALTANTE	ERRÓNEO	ADICIONAL	TOTAL
SECUENCIA O LÓGICA ERRÓNEA				
PROCESAMIENTO INCORRECTO				
LAS ENTRADAS DEL MÓDULO SON INCORRECTAS				
LAS SALIDAS DEL MÓDULO SON INCORRECTAS				
EL MÓDULO NO ACEPTA LOS RANGOS DE DATOS ESPECIFICADOS				
LOS PROCEDIMIENTOS DE RECUPERACIÓN NO ESTÁN / SON INCOMPLETOS / NO SON ADECUADOS				
EL PROCESAMIENTO EN EL MÓDULO DIFIERE DE LO ESPECIFICADO				
VALORES ACEPTADOS ERRÓNEOS / AMBIGUOS				
ALMACENAMIENTO DE DATOS ERRÓNEO / INADECUADO				
VARIABLES FALTANTES O ERRÓNEAS				
OTROS				
TOTALES				

Fig. 17. Formulario de registro de defectos en la fase de diseño detallado

I. Depuración de Programas

La depuración ("debugging") permite al programador evaluar la integridad y precisión del programa previo a la conducción de revisiones y verificaciones menos económicas. La depuración, incluye el diseño detallado y la codificación.
Esta operación puede ser tan extensa o tan reducida como se quiera. La cantidad de depuraciones a realizar dependerá de:

- El tiempo de espera hasta que se reciba el siguiente programa.
- Cronograma de implementación.
- Recursos de prueba
- Eficiencia de herramientas de test.
- Políticas del área.

La depuración puede ser sintáctica, estructural, o funcional.

I.1. Depuración Sintáctica

Las especificaciones y expresiones del programa deben desarrollarse de acuerdo con la metodología definida por la organización y los requerimientos recopilados. El programador puede chequear las expresiones y sintaxis para asegurarse que ha escrito el programa de acuerdo con las reglas establecidas. Al chequear la sintaxis se hacen preguntas como:

- ¿La función a realizar, está claramente identificada?
- ¿Las sentencias del programa están identificadas correctamente?
- ¿Las sentencias del programa están construidas utilizando la estructura apropiada?
- ¿Los elementos de datos están apropiadamente identificados?
- ¿Las estructuras de datos son las adecuadas para colocar los valores que serán utilizados en dichas estructuras?

I.2. Depuración Estructural

Los problemas de estructura crean un significante número de defectos en la mayoría de las aplicaciones. Estos defectos ocultan defectos funcionales por lo que su detección se torna difícil. Las preguntas típicas durante la depuración estructural son:

- ¿Se colocaron todas las sentencias necesarias?
- ¿Se utilizan todas las definiciones de datos en las sentencias definidas?
- ¿Se utilizan todos los elementos de datos definidos?
- ¿Las tablas internas y valores límite están usados de manera que cuando se excede el límite se puede continuar procesando?
- 5.9.3 Depuración Funcional
- Las funciones son los requerimientos que el programa ejecutará. Las preguntas cuando se depura la funcionalidad son:
- ¿El programa ejecutará la función específica en la forma indicada?
- ¿Algunas de las funciones son mutuamente excluyentes?
- ¿El sistema detectará datos incorrectos o ilógicos?
- ¿Se acumulará apropiadamente la información entre diferentes ejecuciones del programa?

J. Análisis de Factores (Módulo de Codificación)

La profundidad de la verificación en el módulo de codificación, depende de cómo se adecua el sistema a las necesidades del usuario, al final del módulo de diseño. A mayor confianza del equipo de verificación en la adecuación de la aplicación al final del módulo de diseño, tendrán menos inquietudes durante el módulo de codificación.
En el módulo de codificación, el equipo de aseguramiento de la calidad debe identificar las inquietudes que sean de mayor interés, y luego desarrollar el proceso de verificación para hacer frente a dichas inquietudes. Al identificarlas, el equipo de aseguramiento de la calidad, debe tener en cuenta los cambios que han ocurrido en las especificaciones del sistema desde que se produjo la última verificación. Los objetivos que los miembros del equipo de aseguramiento de la calidad deben considerar en el módulo de codificación, son:

- ¿Los sistemas son mantenibles?
- ¿Se han implementado correctamente las especificaciones del sistema?
- ¿Los programas son compatibles con los estándares y procedimientos?
- ¿Existe un plan de verificación capaz de evaluar los ejecutables?
- ¿Los programas están adecuadamente documentados?
- Las inquietudes ("concerns") a considerar durante esta tarea se describen a continuación:
- Implementación del control de integridad de información: Es necesario tener implementados controles específicos de manera de lograr la precisión en el procesamiento deseado. Los controles implementados en forma impropia, no alcanzarán el nivel de tolerancia aceptado, y por la mala comprensión del propósito de los controles, serán implementadas soluciones simplistas cuando en realidad se

requieren controles complejos para alcanzar los objetivos de control establecidos previamente.

- Implementación de reglas de autorización: Es necesario verificar la implementación de reglas de autorización de manera de dificultar su evasión. Además, las reglas de autorización no deben sólo considerar la ejecución de las reglas si no también tener en cuenta los métodos más comunes de evadirlas.
- Implementación de controles de integridad de archivos: Los controles de integridad de archivos deben implementarse de manera que minimicen la probabilidad de pérdida la integridad, debiendo además prevenirla y detectarla oportunamente.
- Implementación de auditorías de rastreo: Es necesario implementar una verificación de las auditorias para facilitar la recuperación de información de las mismas (rastreo). Si la verificación de la auditoria contiene información costosa o demanda mucho tiempo, su valor disminuye significativamente. Las consideraciones de la implementación incluyen la cantidad de información a recuperar seguida de la facilidad de recuperación, referencias cruzadas de información para la recuperación y el tiempo que la información de la auditoria necesita ser almacenada.
- Plan de contingencia escrito: El plan de contingencia (serie procedimientos detallados perfilando aquellas tareas a ejecutar ante la ocurrencia de problemas) debe describir las tareas preparatorias para que los datos necesarios y otros recursos estén disponibles cuando sea necesario activarse. Abordar el diseño de la contingencia es de poco valor si no se documenta o no estará en manos de la gente que lo utilizará.
- Consecución del nivel de servicio deseado para el sistema: El nivel de servicio deseado puede solo hacerse realidad cuando los procedimientos y métodos estén implementados. Uno de los procedimientos que debe establecerse, es el monitoreo del nivel de servicio para asegurarse que cumple con las especificaciones. La inclusión de la rutina de monitoreo provee seguridad en el logro del nivel de servicio a largo plazo, caso contrario, se detectará a tiempo para tomar medidas correctivas.
- Implementación de procedimientos de seguridad: La seguridad es la combinación de previsión y entrenamiento, más herramientas y técnicas de seguridad. Los procedimientos que aseguran que tanto las herramientas como las técnicas de seguridad estén disponibles y que trabajen juntas, deben desarrollarse durante la fase de codificación.
- Cumplimiento del programa con la metodología a adoptar: Los procedimientos a implementar deben asegurar conformidad con estándares, políticas, procedimientos y métodos definidos por la organización. Si se detecta la no conformidad, se deberían tomar las medidas necesarias para modificar el diseño y así lograr la conformidad.
- Programas acordes al diseño (Correctitud): El cambio continuo de condiciones puede provocar que varios miembros del personal del proyecto, ignoren los objetivos del mismo durante la fase de codificación. El argumento es que siempre hay cambios, de manera que el ajuste a los objetivos del sistema ya definidos, ahora ya no es muy significativa. El equipo de aseguramiento de la calidad, debe desalentar esta forma de pensar y continuamente monitorear la implementación de dichos objetivos. Si no se han alcanzado los mismos, o deben cambiarse, o bien, debe cambiarse el sistema para tratar de ajustarlos a las especificaciones funcionales ya realizadas de la aplicación.
- Programas acordes al diseño (Facilidad de uso): La implementación de especificaciones del sistema puede invalidar algunas de los aspectos del diseño respecto de la facilidad de uso, a menos que los mismos se encuentren definidos específicamente. La programación es la traducción de especificaciones de diseño a lenguaje ejecutable por la máquina. Esto puede entorpecer el logro de la facilidad de uso. La codificación debe lograr la facilidad de uso, de igual forma en que se intenta cumplir con el resto de las especificaciones funcionales.
- Mantenibilidad de los programas: El método de codificación puede significar más para el mantenimiento que las especificaciones de diseño mismas. Las reglas de mantenimiento deben definirse en parte por los estándares y en parte por las especificaciones del sistema. Además, el programador debe utilizar su juicio y experiencia para desarrollar código altamente mantenible.
- Programas acordes al diseño (Portabilidad): La portabilidad de los programas depende del lenguaje seleccionado y de cómo se usa ese lenguaje. Las especificaciones deben indicar lo que se hace y no se hace en la programación para lograr su portabilidad, y la codificación debe apegarse a esas especificaciones de diseño. Si la portabilidad es una preocupación importante y las especificaciones del programa fallan al definir adecuadamente la portabilidad de la codificación, la misma quedará en manos del programador.
- Programas acordes al diseño (Acoplamiento): Las especificaciones de diseño deben indicar parámetros a pasar desde y hacia otros módulos y aplicaciones del sistema.
- Normalmente es una buena práctica del programador, verificar que las especificaciones del sistema estén actualizadas antes que las funciones sean codificadas. Esto no solo asegura que el programa se ajusta al diseño, sino también, que las especificaciones de interconexión entre aplicaciones no se han modificado desde que se documentó el diseño.
- Desarrollo de procedimientos operativos: Los procedimientos deben desarrollarse durante la fase de programación, de forma de poder operar la aplicación. Durante la fase siguiente, los programas ejecutables serán operados. Los procedimientos operativos deben ser consistentes con los requerimientos operacionales definidos para la aplicación.
- Alcance de los criterios de performance por parte de los programas: La creación del programa provee la primer oportunidad para los usuarios de evaluar si el sistema puede lograr el nivel rendimiento deseado. Una evaluación temprana del rendimiento, brindará una buena oportunidad para hacer ajustes si es necesario.

K. Revisiones por Pares

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

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

La revisión por pares también puede ser formal. Las formales son tareas integrales en el proceso de programación, mientras que las informales son solicitadas a discreción del líder de programación. Además puede aplicarse a otros productos, no sólo al código de un programa.
El equipo de revisión por pares debe tener entre tres y seis miembros. Es importante tener por lo menos tres miembros para obtener variedad de opiniones. Las personas a considerar para este equipo serán:

- Programadores (por lo menos dos).
- Especialistas en JCL ("Job Control Language") o similar.
- Operadores.
- Líderes de programación.

El programa de revisiones por pares se realiza ejecutando las siguientes tareas:

K.1. Establecer Reglas Básicas para la Revisión

Esto no es necesario para cada revisión pero es importante tener buenas reglas de base. Entre dichas reglas, están:

- Áreas incluidas y excluidas de la revisión por pares. Por ejemplo: Ver si la eficiencia de los programas será incluida.
- Cuándo se usarán reportes.
- Método para seleccionar el líder de la revisión por pares.
- Ubicación de la conducción de la revisión.
- Método para seleccionar productos a ser revisados por pares.

K.2. Seleccionar al Equipo de Revisión

Los integrantes del equipo deben ser seleccionados con la suficiente antelación, de manera que puedan organizar su tiempo y estar entrenados para la revisión.

K.3. Entrenar a los Miembros del Equipo

Si una persona del equipo no ha participado previamente en el programa de revisión por pares, se la debe entrenar en el proceso. El entrenamiento incluye el entendimiento de las reglas básicas de esta revisión, preferentemente algún entrenamiento en relaciones interpersonales acerca de cómo entrevistar y trabajar con gente en el proceso de la revisión, y entrenamiento en los estándares y metodología de programación.

K.4. Seleccionar el Método de Revisión

El líder del equipo debe seleccionar el método de revisión. La revisión en sí misma consiste en dos partes. La primera es una explicación general de los objetivos y funcionamiento del programa. La segunda parte es la revisión de los programas utilizando el método seleccionado. Los métodos que pueden ser utilizados para conducir la revisión por pares son cuatro:

- Diagrama de flujo: El programa se explica desde un diagrama de flujo. Esto es más efectivo cuando el diagrama de flujo es producido directamente desde el código fuente.
- Código fuente: La revisión examina cada línea del código fuente para entender el programa.
- Muestra de transacciones: El líder de programación explica los programas exponiendo los procesamientos típicos que ocurren a partir de una muestra representativa de transacciones.
- Especificaciones de programas: Las especificaciones de programas se revisan como un medio para entender el programa.

K.5. Conducir la Revisión

El líder de programación del proyecto normalmente inspecciona la revisión por pares. La revisión comienza habiendo hecho el líder de programación, una revisión rápida de las reglas básicas, explicado los objetivos, y luego conduce al equipo a revisar el procesamiento del programa. El equipo de revisión será libre de preguntar y comentar sobre cualquier aspecto de la explicación del programador y de hacer recomendaciones y sugerencias acerca del programa. Generalmente, la revisión por pares es dirigida en forma democrática.
El rol del líder del equipo es asegurar que las preguntas y comentarios del equipo sean ordenados, asegurar los derechos de hacer preguntas, recomendaciones o parar en un punto específico si, en la opinión del líder de inspección, no tiene sentido continuar discutiendo.

K.6. Conclusiones

Al final de la revisión por pares, el líder de programación indicará cuándo no tiene más comentarios que hacer y lo deja en manos del líder del equipo de revisión por pares. El líder del equipo de revisión por pares, toma el control y resume la información bosquejada de la revisión y presenta las recomendaciones del equipo de revisión. Idealmente, esto se hace como actividad grupal, pero algunos equipos de revisión por pares, especialmente cuando el proceso se formaliza, puede querer algún tiempo a solas para discutir entre ellos lo que han escuchado y lo que van a recomendar. Las conclusiones y recomendaciones luego se presentan al equipo del proyecto para su consideración.

K.7. Reportes

En el proceso de la revisión por pares, se prepara el reporte documentando los resultados. Sin embargo, esto es opcional y no es una parte esencial del proceso de revisión por pares.
El formato de los informes típicos será similar al que ya se han detallado con anterioridad:

- Informe de Inspección
- Resumen de la Inspección

L. Verificación de Documentación

L.1. Integridad de la Documentación

Los criterios principales para la verificación de la integridad de la documentación, surgirán de la metodología y estándares de la organización. A continuación, se detallan criterios adicionales para verificar la integridad de la documentación:

- Contenido de la documentación: Al índice de cada documento le debería seguir una breve descripción de cada elemento dentro del documento. Cada documento debe tener un índice de contenidos mínimos obligatorios, para poder determinar si los documentos contienen toda la información necesaria. Si faltan elementos, hay un defecto potencial en la integridad de la documentación, la cual debería ser notificada.
- Usuarios del documento: Cada tipo de documento estará escrito dirigido a un usuario en particular, el cual podrá ser una persona o grupo, los cuales usarán el documento para ejecutar una función. (Por ejemplo, operación, mantenimiento, diseño y programación).

- La información deberá ser presentada con la terminología y el nivel de detalles apropiados para el tipo de usuario al cual está dirigido el documento.
- Redundancia: Los documentos típicos dentro del proyecto (Ej. Especificación de Requerimientos, Análisis de Factibilidad, etc.) tendrán alguna redundancia. Se deberá incluir material introductorio en cada tipo de documento para proveer al lector de un marco de referencia, facilitando la interpretación del documento con la menor cantidad de referencias cruzadas hacia otros documentos. La información que debería ser incluida en cada tipo de documento diferirá en su contexto y a veces en la terminología y nivel de detalle, porque intenta ser leído por diferentes usuarios en diferentes puntos del ciclo de vida del software
- Flexibilidad. La flexibilidad en el uso de los documentos depende de los lineamientos vigentes en la organización.
- Tamaño del documento: Cada tipo de documento puede tener un tamaño que va de unas pocas a varias decenas de hojas. La longitud depende del tamaño y complejidad del proyecto y del juicio del Gerente de proyecto así como el nivel de detalles necesario para el ambiente en el cual el software deberá desarrollarse y ejecutarse.
- Combinación de diferentes tipos de documentos: En ocasiones, es necesario combinar algunos tipos de documentos bajo una sola portada o producir algunos volúmenes con los mismos tipos de documentos. Los tipos de documentos que pueden ser combinados son manuales para el usuario, operaciones, y mantenimiento del programa. Para sistemas de gran envergadura, puede ser preparado un documento para cada módulo. Algunas veces el tamaño del documento puede requerir ser repartido en varios volúmenes para hacer más fácil su uso. En esos casos, deberán estar separados por secciones, sub-secciones, etc.
- Formato: El formato obligatorio debe ser verificado, y debe recomendarse el uso del mismo.
- Secuencia de contenido: En general, el orden de las secciones y párrafos de un tipo de documento debe ser el mismo que el que se muestra en el índice de contenidos obligatorios. El orden puede cambiarse si enriquece la presentación, caso contrario, debería ajustarse a los estándares ya definidos por la organización.
- Títulos de secciones/párrafos. Estos títulos generalmente son los mismos que los mostrados en la índice de contenidos. Pueden modificarse para reflejar la terminología del software a documentar si el significado le agrega valor a la presentación. Se pueden agregar o eliminar secciones o párrafos según sea necesario.
- Expansión de los párrafos: La mayoría de los tipos de documentos definidos por la metodología, estándares o lineamientos de la organización tienen párrafos con un título general y una lista de puntos que pueden discutirse dentro de ese párrafo. La intención de esa lista no es establecer la discusión de cada uno de estos puntos, sino sugerir la consideración de los mismos al escribir el párrafo. Este y todos los otros párrafos pueden expandirse y subdividirse para llevar a cabo una mejor presentación de la información.
- Diagramas de flujo y tablas de decisión: Los gráficos de las soluciones a problemas en forma de diagramas de flujo o tablas de decisión, pueden estar incluidos o ser un apéndice del documento.
- Formularios: El uso de formularios específicos depende de las prácticas de la organización. Parte de la información especificada en los párrafos del índice de contenidos puede registrarse en tales formularios. Si es así, el formulario puede referenciarse desde el párrafo apropiado. Se requiere usar los formularios que conforman los estándares de la organización.

L.2. Grado de actualización de la Documentación

El equipo de verificación de la documentación puede usar cualquiera de las siguientes cuatro verificaciones propuestas para validar la actualización de la documentación. Los tipos de verificación posibles se enumeran a continuación:

L.2.1. Utilizar la Documentación Vigente

La actualización puede validarse utilizando la documentación vigente para realizar algún cambio en la aplicación. Esto permite al analista de calidad buscar y confirmar la consistencia entre varios documentos (por ejemplo, que las especificaciones en el documento de diseño del programa, sean las mismas que están en el código actual) y determinar si la documentación respalda el funcionamiento del sistema.

L.2.2. Comparar el Código Fuente con la Documentación

Esta verificación usa la versión actual del código fuente de uno o más programas como base de la documentación. Esta verificación es realizada sobre la base de una muestra elegida al azar de los programas o parte de los mismos y se las verifica contra la documentación. El objetivo es determinar si el código es representativo de la documentación que se posee del mismo.

L.2.3. Verificar la Vigencia de la Documentación

Las personas que elaboran la documentación deben verificar que esté vigente. Deben hacerse preguntas específicas, como éstas:

- ¿La documentación es 100% representativa de la aplicación en operación?
- ¿La documentación cambia cada vez que se hace un cambio en el sistema?
- ¿Los individuos que cambian el sistema confían en que la documentación es la correcta?

L.2.4. Verificar la Actualización de la Documentación con el Usuario

Los usuarios finales deben preguntarse si la documentación para el sistema está actualizada. Si los usuarios finales no están familiarizados con la documentación, necesitarán que se seleccione una muestra pidiendo piezas específicas de la documentación. La documentación seleccionada debe ser familiar para el usuario final de manera que se les pueda dar partes representativas del documento y pedir que validen si están actualizadas o no.

VI. Nivel De Servicio Esperado

En el presente capítulo se describe el Nivel de Servicio Esperado, a través de la cuantificación de los resultados esperados.

A. Generalidades

A.1. Necesidad de cuantificación

Toda aplicación de un modelo o servicio, requiere de ciertos niveles de servicio o indicadores que permitan obtener una

cuantificación de los resultados de la aplicación del mismo y/o establecer las expectativas para aplicaciones futuras. Para ello, se describen cuatro indicadores.

Los indicadores aquí presentados, tanto su definición como el valor objetivo sugerido, constituyen sólo una muestra no estática del tipo de indicadores que habría que considerar; es decir se pueden agregar nuevos y/o modificar los aquí presentados, conforme a la capacidad de la organización y a la experiencia en la aplicación del modelo.

A.2. Definición de términos

Con el objeto de clarificar diferentes términos que se suelen confundir o usar indistintamente y para precisar los acuerdos de nivel de servicio, se presenta a continuación las siguientes definiciones, propuesta en [6]:

- Errores: Estos son equivocaciones humanas como los errores tipográficos o sintácticos.
- Defectos: Estos son condiciones impropias de un programa que generalmente son el resultado de un error. No todos los errores producen defectos en el programa, como comentarios incorrectos o algunos errores de documentación. Por el contrario, un defecto puede producirse por causas ajenas al programador, como problemas con las herramientas.
- Bugs: Son defectos del programa que se encuentran operando el mismo, ya sea durante la prueba o durante su uso. Los bugs provienen de los defectos, pero no todos los defectos causan bugs (algunas están latentes pero nunca se encuentran). Los defectos pueden encontrarse muchas veces, dando como resultado múltiples bugs por defecto.
- Fallas: Es un mal funcionamiento en una instalación de usuario. Puede ser provocado por un bug, una instalación incorrecta, una falla del hardware, etc.
- Problemas: Son dificultades encontradas por los usuarios. Pueden provenir de fallas, mal uso o mal entendimiento. Los problemas son eventos relacionados con los humanos, contrariamente a las fallas que son eventos relacionados con el sistema.
- Entradas Unitarias: Corresponde a cada uno de los elementos del conjunto que componen la entrada a cada uno de los módulos del modelo de Aseguramiento de Calidad de Software propuesto. Ejemplos de estos elementos o entradas unitarias son documentos, especificaciones, planes, diagramas, etc.

En la tabla XVII se resumen algunas de las definiciones establecidas:

TABLA XVII. DEFINICIÓN DE TÉRMINOS

Categoría	Ítems medidos	Causas
Errores	Acciones humanas	Equivocaciones del programador
Defectos	Propiedades del programa	Errores
Bugs	Mal funcionamiento del programa	Defectos del programa
Fallas	Mal funcionamiento del sistema	Bugs y otros mal funcionamientos
Problemas	Percepciones humanas	Fallas, errores humanos, incomprensión humana

En la figura 18 se muestran las relaciones entre las categorías de la tabla anterior. Este diagrama tiene una doble utilidad:

- Comenzando con el error del programador, se puede rastrear en el diagrama hasta llegar al problema.
- Comenzando con el problema se pueden rastrear las posibles causas.

B. Indicadores

B.1. Indicador A

El indicador A establece el nivel tiempo promedio en que debe poder adaptarse el plan general de aseguramiento de la calidad a un proyecto en particular. El esquema correspondiente al indicador A se muestra en la figura 19.

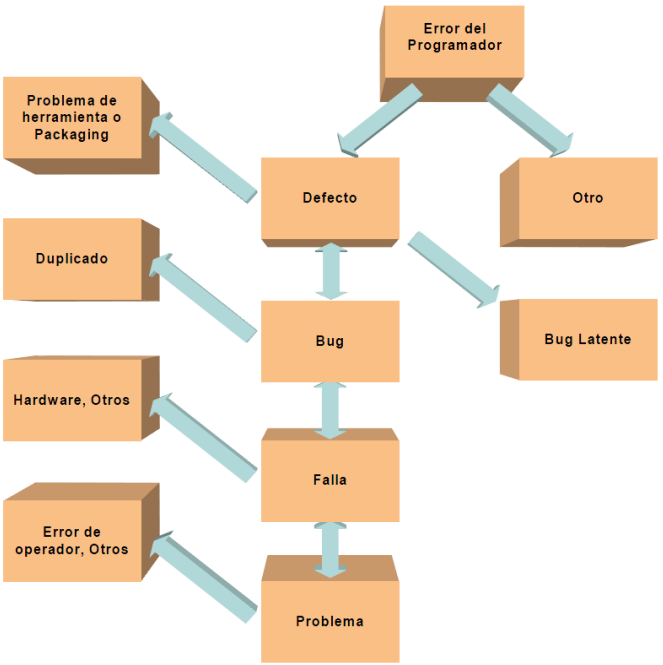

Fig. 18. Proceso del módulo de metodología, estándares y procedimientos

- Descripción: Tiempo promedio insumido en la generación del plan específico de aseguramiento de la calidad para un proyecto dado.
- Valor objetivo sugerido: Menor o igual a 1 semana.

Fig. 19. Tiempo promedio en que se debe adaptar el plan general al proyecto en particular

B.2. Indicador B

El indicador B establece el nivel de la capacidad de trabajo concurrente del equipo de SQA en su conjunto. El esquema correspondiente al indicador B se muestra en la figura 20.

- Descripción: Capacidad máxima de entradas unitarias en proceso de verificación concurrente.
- Valor objetivo sugerido: Hasta 10 entradas unitarias.

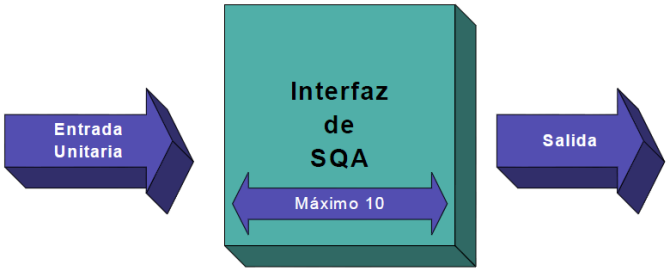

Fig. 20. Nivel de capacidad de trabajo concurrente de SQA

B.3. Indicador C

El indicador C establece el nivel de la rapidez de trabajo del equipo de SQA. El esquema correspondiente al indicador C se muestra en la figura 21.

- Descripción: Tiempo máximo promedio de verificación para una entrada unitaria.
- Valor objetivo sugerido: Menor o igual a 1 semana.

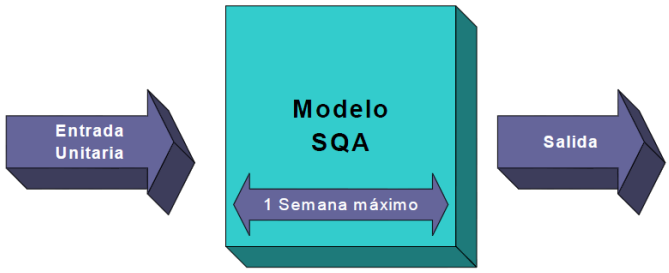

Fig. 21. Rapidez de trabajo de SQA

B.4. Indicador D

El indicador D establece el nivel de eficacia del trabajo del equipo de SQA. El esquema correspondiente al indicador D se muestra en la figura 22.

- Descripción: Porcentaje de defectos detectados en el proceso de aseguramiento de la calidad del software (Considerando el total de defectos de un sistema desde el inicio del proyecto hasta 6 meses después de su puesta en producción).
- Valor objetivo sugerido: 75% de los defectos

Fig. 22. Nivel de eficacia del trabajo de SQA

VII. Equipo De SQA

En el presente capítulo se detalla el Equipo de SQA sugerido, indicando su estructura y responsabilidades.

A. Estructura del equipo aseguramiento de la calidad

El equipo de trabajo de SQA puede tener variadas estructuras y dependerá de la propia realidad y características particulares de la organización en la cual el equipo se desempeña.

En el presente trabajo se presentará una estructura en tres niveles, muy común en varias organizaciones. En la figura 23 se esquematiza esta estructura:

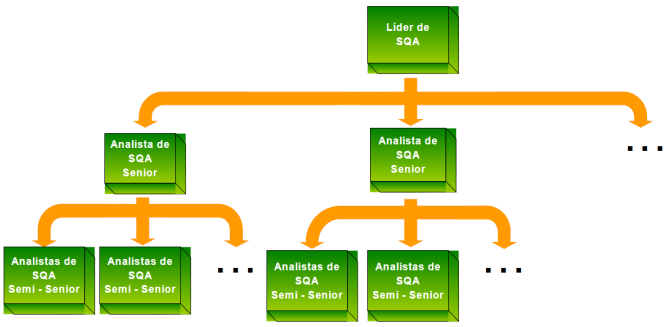

Fig. 23. Estructura en tres niveles del equipo de aseguramiento de la calidad

Cada sub-equipo, confirmado por un analista de SQA Senior y sus Analistas de SQA Semisenior, podría ser asignado a un proyecto en particular, mientras que, el Líder de SQA de la organización, tiene el control sobre los sub-equipos y mantiene una visión general que comprende a todos los proyectos de dicha organización.

B. Responsabilidades del equipo

A continuación de describe, sólo a modo de ilustrativo, las principales responsabilidades de cada uno de los perfiles establecidos en la estructura de la figura 23.

B.1. Líder de SQA

- Mantenimiento del dialogo primario con los referentes del área de desarrollo.
- Administración del presupuesto de los proyectos para la función de SQA.
- Responsable de asegurar al equipo de trabajo los elementos que éste o cualquiera de sus integrantes necesite.
- Coordinación de las necesidades políticas de los proyectos con las capacidades operativas disponibles.
- Coordinación de responsabilidades entre los analistas de SQA de los distintos proyectos.
- Mantenimiento de la conducción de todo el equipo de SQA.
- Determinación de los objetivos de SQA, para etapa de un proyecto y control de su cumplimiento.
- Elaboración periódica de informes de avance y seguimiento.
- Participación en reuniones técnicas, siempre que resulte necesario.
- Participación en los módulos que se indica en el modelo de aseguramiento de la calidad propuesto.
- Supervisión (directa o indirecta) de los analistas de SQA.
- Evaluación de cada integrante del equipo.
- Responsable de la disciplina y cumplimiento del equipo.

B.2. Analista de SQA Senior

- Aseguramiento del cumplimiento de su presupuesto asignado.
- Aplicación de la metodología de aseguramiento de la calidad para los proyectos asignados.
- Aseguramiento del cumplimiento de los objetivos fijados por el líder de SQA.
- Aseguramiento del cumplimiento de la metodología de trabajo fijada.

- Participación en los módulos que se indica en el modelo de aseguramiento de la calidad propuesto.
- Conducción y administración del equipo asignado a su cargo.
- Asignación de tareas a los analistas de SQA Semi-senior.
- Supervisión de las tareas de los analistas de SQA Semi-senior que le correspondan en cada proyecto.
- Elaboración periódica de informes de gestión de su equipo.
- Elaboración de informes por excepción cada vez que resulte necesario.
- Comunicación inmediata en situaciones que excedan su capacidad de control o decisión.
- Responsable primario de la disciplina y cumplimiento de su equipo.
- Colaboración técnica con el Líder de SQA en las reuniones o consultas que éste requiera.
- Asesoramiento al Líder de SQA en toda cuestión técnica vinculada a su área de influencia.

B.3. Analista de SQA Semi-senior

- Participa en los módulos que se indica en el modelo de aseguramiento de la calidad propuesto.
- Cumplimiento detallado con la metodología de trabajo impuesta y con las pautas que le fijen sus supervisores.
- Elaboración de informes detallados de cada unidad de análisis.
- Elaboración de informes por excepción cada vez que resulte necesario.
- Comunicación inmediata en situaciones que excedan su capacidad de control o decisión.
- Colaboración técnica con el Analista de SQA Senior en las reuniones o consultas que éste requiera.

VIII. CONCLUSIONES

En el presente capítulo se establecen las Conclusiones obtenidas al concluir el presente trabajo, señalando los beneficios y problemas esperados en la aplicación de la función de SQA en una organización. También se establecen las bases para un análisis costo-beneficio.

A. Beneficios y problemas

Aunque pocos profesionales pondrían en duda la necesidad de calidad en el software, muchos no están interesados en establecer funciones de SQA formales. Las principales razones de esta aparente contradicción son las siguientes:

- Los responsables del desarrollo se resisten a hacer frente a los costos extras inmediatos y les cuesta ver los beneficios a largo plazo.
- Por desconocimiento, muchos profesionales creen que ya están haciendo todo lo que hay que hacer con respecto al aseguramiento de la calidad.
- Pocos saben dónde situar esa función dentro de la organización.
- Todos quieren evitar cierto nivel de burocracia que la función de SQA tiende a introducir en el proceso de ingeniería del software o de conocimiento.

En el presente trabajo, se ha presentado un enfoque práctico para la aplicación de la función de aseguramiento de la calidad del software en una organización, que intenta echar luz sobre las razones ya enunciadas que impiden la aplicación de la dicha función.

Aunque instanciada en la metodología IDEAL, este enfoque es independiente de la metodología de desarrollo que esa organización utilice. La independencia del enfoque con respecto a metodología utilizada por la organización, se debe a qué el enfoque presentado es general, aplicable a fases comunes de cualquier metodología y a que no obliga a su aplicación rígida, sino que permite su adaptación a esa metodología y a la propia realidad de la organización.

Asimismo, el enfoque presentado constituye un complemento no disruptivo de la metodología de desarrollo que ya utiliza una organización y no obliga a profundos cambios en la misma. Por el contrario, si la organización no utiliza una metodología, incentiva la utilización de una.

Algunos de los principales beneficios esperados de la aplicación constante y rigurosa de la función de SQA, mediante el enfoque presentado, son los siguientes:

- El software tendrá menos defectos latentes, como consecuencia, se reducirá el esfuerzo y el tiempo durante las etapas de prueba y mantenimiento.
- Se dará una mayor fiabilidad y, por tanto, una mayor satisfacción del cliente.
- Se podrán reducir los costos de mantenimiento (un porcentaje sustancial de los costos totales del software).
- El tiempo y el costo total del ciclo de vida del software disminuirá.
- Por el lado negativo, la función de SQA podría resultar problemática por las siguientes razones:
- Es difícil institucionalizar en organizaciones pequeñas, en las que no están disponibles los recursos necesarios para llevar a cabo esas actividades.
- Representa un cambio cultural, y el cambio nunca es fácil.
- Requiere un gasto que, de otro modo, nunca se hubiera destinado explícitamente a la ingeniería del software o de conocimiento o al aseguramiento de la calidad.

B. Evaluación costo-beneficio

A la hora de establecer la función de aseguramiento de la calidad del software en una organización, es razonable preguntarse si valdrá la pena, si el costo de su establecimiento y aplicación continua se verá justificado por los beneficios alcanzados.

En el marco de un análisis básico de costo-beneficio, se puede afirmar que la función de SQA en una organización será efectiva si se cumple con la siguiente: $A > B + C$ siendo:

A: Costo de las fallas que aparecen sin la aplicación de la función de SQA. Esto incluye, entre otros, las actividades de reparación y re-trabajo, resolución de quejas del cliente, retorno y reemplazo del producto, soporte y ayuda al cliente, y el pago de penalidades contractuales.

B: Costo de la propia aplicación de la función de SQA. Esto incluye, entre otros, los salarios del equipo de SQA y las actividades de planificación, revisiones y auditorias.

C: Costo de las fallas que no se encuentran con la aplicación de la función de SQA. Esto incluye los mismos puntos que A, pero deberían ser dramáticamente inferiores.

Sin embargo, es importante resaltar que en un análisis más minucioso habría que considerar también otros aspectos, tales como:

- Reducciones en los costos de las pruebas y de la integración.

- Reducción del número de cambios en las primeras versiones.
- Reducción en los costos de mantenimiento.
- y otros de no tan fácil cuantificación, tales como:
- Mejora en satisfacción del cliente.
- Mejora en la imagen de la imagen externa de la organización.
- Mejora en la imagen interna del equipo de ingeniería del software o de conocimiento.

Finalmente, también es necesario tener en cuenta al momento de hacer un análisis costo-beneficio, que la función de SQA no necesariamente está circunscripta al ámbito de la ingeniería del software o de conocimiento, sino que evoluciona como parte de un esfuerzo general de gestión en la organización, dirigido a mejorar la calidad. A ese esfuerzo general de gestión dentro de la organización se lo suele denominar "gestión de la calidad total".

IX. Bibliografía

[1] Roger S. Pressman (2001). Software Engineering: A Practitioner's Approach. McGraw-Hill.

[2] J. Mc Call, P. Richards y G. Walters (1977). Factors in Software Quality. Rome Air Development Center, United States Air Force.

[3] R. Grady y D. Caswell (1987). Software Metrics: Establishing a Company-Wide Program. Prentice Hall.

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

[5] William E. Perry (2000). Effective Methods for Software Testing. Wiley Computer Publishing.

[6] Watts S. Humphrey (1989). Managing the Software Process. Addison Wesley.

X. Financiamiento

Las investigaciones cuyos resultados se presentan en este trabajo fueron parcialmente financiados por subsidio del proyecto UNLa 33B102.

Eduardo Diez. Es Profesor Asociado del Area de Ingeniería de Software en la Licenciatura en Sistemas (Res. CONEAU 1089/12) del Departamento de Desarrollo Productivo y Tecnológico de la Universidad Nacional de Lanús y Profesor Adjunto Regular del Área de Ingeniería de Software en la Carrera de Ingeniería Informática de la Facultad de Ingeniería de la Universidad de Buenos Aires. Es Profesor de la Maestría en Ingeniería de Software (Acreditada por Resolución CONEAU nro 239/04) en el marco del Convenio Universidad Politécnica de Madrid - ITBA y Profesor de Posgrado en la Maestría en Ingenieria de Sistemas de Información en la Escuela de Posgrado de la Facultad Regional Buenos Aires de la Universidad Tecnológica Nacional. Es Investigador del Grupo de Investigación en Sistemas de Información y dirige del Laboratorio de Investigación y Desarrollo en Aseguramiento de Calidad de Software de la Licenciatura en Sistemas y de la Maestria en Sistemas de Información del Departamento de Desarrollo Productivo y Tecnológico de la Universidad Nacional de Lanús. Su areas de interes en investigación son Calidad de Software y Modelos de Madurez de Procesos de Software. Es Director del Proyecto UNLa 33B102 Aseguramiento de la Calidad para Proyectos de Explotación de Información. Es Analista de Sistemas y Licenciado en Sistemas por la Universidad de Belgrano. Es Especialista en Ingeniería de Sistemas Expertos y Magister en Ingeniería de Software por el Instituto Tecnológico de Buenos Aires y por la Facultad de Informática de la Universidad Politécnica de Madrid.